A Report on QR-Based Testing

Harald Brandl and **Gordon Fraser** and **Franz Wotawa** *
Institute for Software Technology
Graz University of Technology
8010 Graz, Austria
{brandl,fraser,wotawa}@ist.tugraz.at

Abstract

As reactive and embedded systems continuously interact with their environment, it is important to test as many as possible interactions. The use of qualitative models of the environment and hardware has the potential to provide test cases that might not be considered with traditional testing methods. We present an approach that derives abstract test cases from such models using qualitative reasoning, which is a well known artificial intelligence technique to represent and reason about physical behavior. For this purpose we introduce the underlying concepts of qualitative reasoning, show the test case generation process, and provide the results of a case study.

Introduction

This paper extends the ideas proposed by Franz Wotawa (Wotawa 2007) and presents first results for test case generation from QR-models.

The growing demand for smarter products with increased functionality leads to a steady increase of the complexity of systems and software. As an example, current cars typically have more than 40 control units with many sensors and actuators on board; this number will further increase. Verification and validation are therefore very important, and lead to many issues related to the automation of test case generation and execution. In the context of reactive and embedded systems, the difficulty of automated testing is further increased because there is a high degree of interaction with the surrounding environment. Because of this, correctness cannot be guaranteed solely by testing the implemented functionality, but also requires testing of the reactions to general stimuli originating from the environment. Although the behavior upon certain wrong inputs is sometimes explicitly specified in the case of reactive and embedded systems, it is unlikely that all important cases are considered at design time. Consequently, systems might fail in some cases when interacting with the physical world.

In order to overcome this problem it is necessary to consider the behavior of the physical world and its interaction with the system. Qualitative reasoning (QR), which originates from the field of Artificial Intelligence (AI), provides us with means for deriving all possible behaviors. QR was originally developed to model commonsense reasoning within the physical world, e.g., to allow for generating answers to questions like "If I throw a stone upwards, what will happen?" A QR simulator will not only provide us with one answer like the stone will move up to a certain point and fall down afterwards, but with all possible answers, including that the stone might go up and up if thrown fast enough.

In general, model based testing techniques use some kind of formal specification describing a system's expected behavior in order to determine whether an implemented system is correct. The specification is used to derive test cases and also serves as oracle, which decides if an executed test case detected an error. For more details on test based on formal specifications we refer to existing surveys on the topic; e.g., (Hierons et al. 2008). The semantic model of most of these formal languages can be interpreted as a kind of transition system representing a system's time variant behavior. Unlike traditional formal specifications, QR techniques allow to describe a system in a declarative manner by a set of qualitative differential equations (QDEs). From this compact representation a QR engine derives a transition system which represents all possible behaviors that can evolve over time starting from an initial scenario.

QR modeling tools like Garp3 (Bredeweg et al. 2006) are well suited when specifying physical systems on an abstract behavioral level. Especially when there is already a mathematical description of the system available (set of differential equations), the mapping can be done in a straightforward way. Therefore, a focus of the presented approach lies on control systems. The continuous domain of control systems (or integer domain for digital controllers) can be transferred via qualitative abstraction and simulation to a transition system (TS).

In this paper, we briefly introduce the basic concepts of QR. As a running example we use a system that sends GPS data to a base station via GSM on a regular basis (Figure 1); such

*Authors are listed in alphabetical order

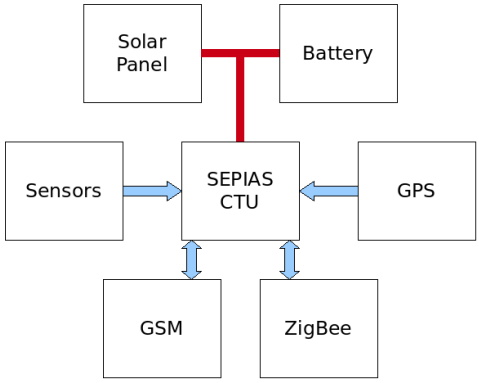


Figure 1: Block Diagram of a container tracking unit.

Figure 2: The implemented prototype of the container tracking unit.

systems can for example be found in the logistics domain. To increase mobility the system has a solar panel which charges a battery. The system monitors the state of the battery and the solar panel and acquires other environmental parameters like the temperature via its sensors. Once new data has been collected, the system decides upon the next actions or if it should go to a power saving mode. A near field communication module allows data exchange with other devices within range.

The running example will be used to introduce the basic concepts of generating test cases from QR models, and as a case study to evaluate the feasibility of the presented approach. The possible behaviors inferable from QR models are represented as labeled transition systems (LTS). Manually specified test purposes are used to derive test cases from the LTS. For example, from the requirement that the system should not run out of battery we create a test purpose that expresses that we do not allow a battery to completely discharge. Traditional LTS conformance testing techniques are adapted to be applicable to QR based models, allowing derivation of test cases from the synchronous product of the system LTS and the test purpose. As the obtained test cases are abstract they have to be refined in order to be executable on an implementation under test (IUT). This is realized by applying abstraction/refinement relations to the information exchanged between a test case and an IUT.

The main contributions of this paper are as follows:

- Models that are close to hardware are used for test case generation.
- The underlying models can include the system hardware, the part of the software which interacts with the environment, and the environment itself.
- The use of environmental models ensures that all possible behaviors are covered, while during manual test case generation some details may be missed.
- We adapt the input-output conformance testing theory (ioco)(Jard and Jeron 2004) to transition systems derived from QR models.

The remainder of the paper is organized as follows. First, we introduce the basic concepts of QR and the modeling of our running example. Then we present our approach of test case generation from QR models in detail and present the first results of our case study. Finally, we discuss related research and conclude the paper.

QR Modeling with Garp3

Garp3 provides every means to build and inspect QR models. A detailed description of the functions can be found in the user manual (Bouwer, Liem, and Bredeweg 2005), and there is an elaborate user guide (Bredeweg et al. 2005) for building QR models.

A Garp3 model consists of a set of model fragments, which are the basic units that describe behavior. Two main types of fragments can be distinguished: static and process model fragments. A third fragment type called agent can be used to model exogenous influences on the system. Static fragments represent behavior that is invariant with regard to time, such as proportional relations between quantities, like, for example, “the amount of water in a vessel is proportional to the water level”. A dynamic fragment introduces changes via influences between quantities, for example “a positive flow rate into a vessel will increase the amount of liquid and hence the liquid level over time”.

Usually, systems consist of several model fragments that are activated when certain boundary conditions are met. Garp3 uses colors to represent information like, e.g., the identification of conditional elements in a model. Garp3 adds the consequences of a model fragment as new facts to the knowledge base, unless they contradict existing facts. Model fragments enable the designer to partition the system domain into qualitative equivalence classes that capture certain behavior. During simulation the set of fragments collected in a library changes between being active and inactive as the system evolves over time.

Within a model fragment the main modeling primitives are entities, quantities, proportionalities, and a set of ordinal re-

lations. In dynamic model fragments there are additional influences. Entities are the components of the system that have certain properties expressed through associated quantities. For example, the entity battery has the quantities "voltage", "current", and "charge".

Proportionalities establish a mathematical relation between two quantities in the form of a monotonic increasing or decreasing function. The notation P+ (Q1;Q2) expresses that a change of Q2 causes a change of Q1 in the same direction. A proportionality with a minus sign states that a change of the cause quantity induces a change in the opposite direction of the effected quantity.

Ordinal relations called inequalities provide means to constrain possible behavior. Influences cause dynamic changes of the system and provide means for integration. For instance, I+ (Q1,Q2) means that the value of Q2 determines the change of direction of Q1. If Q1 is positive Q2 increases, if Q1 is zero Q2 does not change, and if Q2 is negative Q1 decreases. The graphical notation used in Garp3 states relations with arrows between quantities.

The initial state of the system is captured with scenarios. This initial state and the model fragments serve as input to the simulation engine. Simulation is used to generate the behavior of a QR system. The simulation engine derives everything that does not contradict the boundary conditions, i.e, inequalities between quantities. QR models can only describe systems with continuously changing quantities, as stated by the continuity law (Forbus 1984). In general, model creation is an iterative process: One has to find the right level of abstraction and check if the simulation output satisfies the requirements. If there are discrepancies, the model has to be adapted and simulated again.

We use Garp3 models to describe hybrid systems, i.e., systems that combine discrete and continuous behavior, in a qualitative fashion. Control modes of hybrid systems can be captured with sets of model fragments that define a certain control mode when applied. Whenever a model fragment is activated or deactivated during simulation the system switches to a different mode. Garp3 models represent a restricted form of hybrid systems, i.e., they require that changes of variables between control modes follow the continuity law.

Example Model

Our example of a container tracking unit (see Figure 1) consists of a solar panel, a battery, and the controller board drawing some load current. In addition, there are external quantities such as temperature and light conditions.

Figure 3 shows the model fragment for the solar panel. It is a static fragment that defines the main proportionalities (P-arrows) and correspondences (arrows with V for value-, and

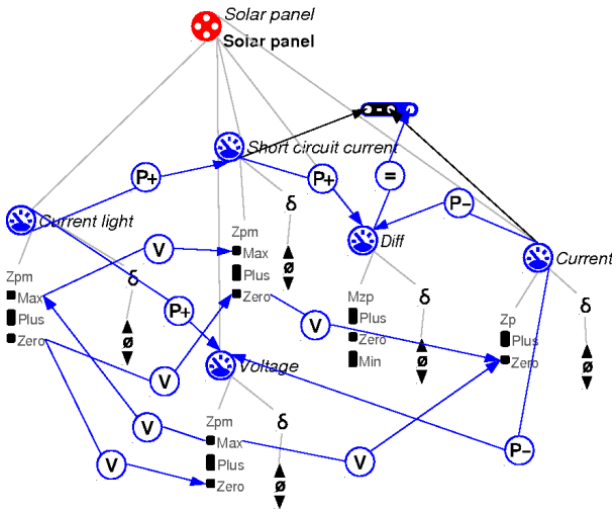


Figure 3: Solar Panel.

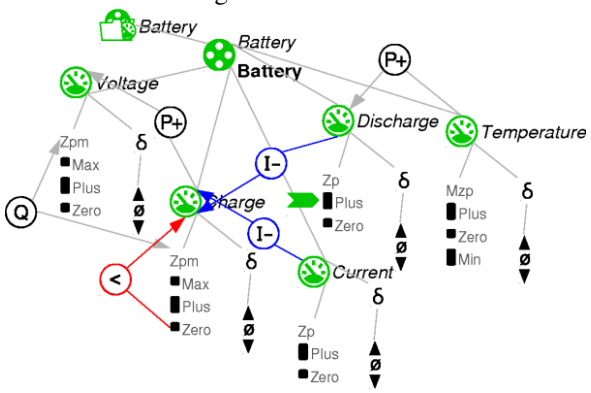


Figure 4: Battery.

Q for quantity space-correspondences) between its quantities. A second fragment of the solar panel not shown here is a conditional one that becomes active when the current drawn from the panel is greater than the generated current. If this condition is met, the fragment introduces a negative feedback (P-) from the calculated *Diff* quantity to the drawing current. This limits the maximum current that the panel can deliver due to the current light conditions. A further property the model reflects is that the voltage decreases with increasing load current (P-). In the case of a short circuit the voltage changes to zero while the maximum load current is provided.

Certain corresponding boundary values are stated with directed or undirected *value correspondences*. A directed value correspondence (V) from *Current light = zero* to *Voltage = zero* indicates that the voltage is zero if the current light is zero. This restricts the state space to allowed behavior. The model fragment has two interface quantities *Current light* and *Current* that can be referenced by other model fragments or exogenous quantities.

The model fragment in Figure 4 depicts the discharging behavior of a battery. The fragment is activated if the charge level of the battery is greater than zero. Furthermore, a discharge rate greater than zero is defined (the arrow on the plus value) which is directly proportional to the temperature. The discharge rate and the load current decrease the battery's charge level as stated by the negative influences. The correspondence (Q arrow) and proportionality (P arrow) between charge and voltage causes *Voltage* to take on the same value as *Charge*.

Figure 5 shows the compound model fragment of our system comprising the solar panel, the battery, and the container tracking unit represented as current drawing load. The fragment is active when the voltage of the solar panel is greater than the battery's voltage, causing the flow of a charge current from the panel to the battery. The arrow on the solar panel's current quantity requires the charge current to be greater than zero. Note that the system software has to capture the behavior of the physical world as far as the specification is concerned.

Test Case Generation

When testing reactive systems it is important that the system model includes the behavior of the environment. To see why this is so, consider an analogy from control theory: The combination of the control process with the controller forms the whole system. These two components influence each other, and many important conclusions like stability of the closed loop system cannot be drawn when looking at them separately. This also applies to testing, where both the system state and the environmental conditions have to be considered. This section shows a method for deriving test cases from QR models and a technique to minimize them.

Deriving Test Models from Garp3 Models

We classify systems based on how they interact with the environment. The first type comprises conventional control systems interacting in both directions with sensors that perceive the environment and actuators that change it. The second type are systems that only perceive their environment without changing it. The gathered information is used to adapt the internal behavior, which is not directly observable from the outside world. In this context the term environment comprises anything but the system software. As software is not well suited for modeling with Garp3 we draw this line between software and the physical world.

Garp3 is suitable to model systems that can be completely specified by their observable external behavior. If there are additional requirements on the internal system state (e.g., "the CPU has to operate in low power mode"), it is necessary to monitor the execution of the software. We propose to annotate the system software with assertions that can check whether certain conditions are fulfilled. A test case can only lead to a pass verdict if there was no assertion violation.

During test case execution the system software may influence the environment and hence any trace in the environmental model that serves as test case. Therefore, we define interaction points as a subset of the quantities in the Garp3 model. The quantities Q are partitioned into three sets, viewed from the system side: input, output, and hidden. Input quantities are refined to concrete sensor inputs for the system, and system outputs are abstracted to output quantities influencing the environment. Consequently, the set of interaction points is the set of all but the hidden quantities.

The simulation output of a QR model is a state space representation of all possible behaviors that may evolve over time, starting from an initial scenario. We define this output as a QR transition system (QR TS) $M = (S, T, s_0, Q, qs, QS, v, \delta)$, where S is the set of states, T is the transition relation $T \subseteq S \times S$, and $s_0 \in S$ is the initial state.

Every quantity in the set of quantities Q of the simulation output has an associated quantity space. QS denotes the domain of quantity spaces, and the function $qs : Q \rightarrow QS$ maps each quantity to its associated quantity space. Each state in the state space binds all quantities to a distinct value and delta. The value v for quantities in Q and states in S is defined as $v : S \times Q \rightarrow qs(Q)$, and the delta δ is defined as $\delta : S \times Q \rightarrow \{min, zero, plus\}$. The $delta$ of a value stands for its direction of change over time, $\delta \sim \frac{\partial value}{\partial t}$.

If there is a transition between two states, then either a value or a delta for some quantity changes. The continuity rule proposed by De Kleer and Brown (Kleer and Brown 1984) states that a qualitative value as a function of time cannot jump over values in its domain but has to change continuously. As all state transitions computed by the QR engine adhere to this rule, it follows that on every transition from a state to a successor state at least one quantity changes its value or direction; otherwise, the two states are qualitatively equivalent.

Test Purposes for QR Models

In order to generate test cases from QR TS derived from QR models, we adapt techniques from LTS testing. An LTS extends a transition system with a set of labels P and a function L mapping transitions to labels: $L : T \rightarrow 2^P$, where P is a set of atomic propositions. Garp3 is able to enumerate the complete state space because of qualitative abstraction as is possible for LTS, but in contrast to LTS, QR TS comprise states with simultaneously changing inputs and outputs. Consequently, the behavior of interest is not only specified via the occurrence of some state sequences. In addition we are interested in the relations between quantities.

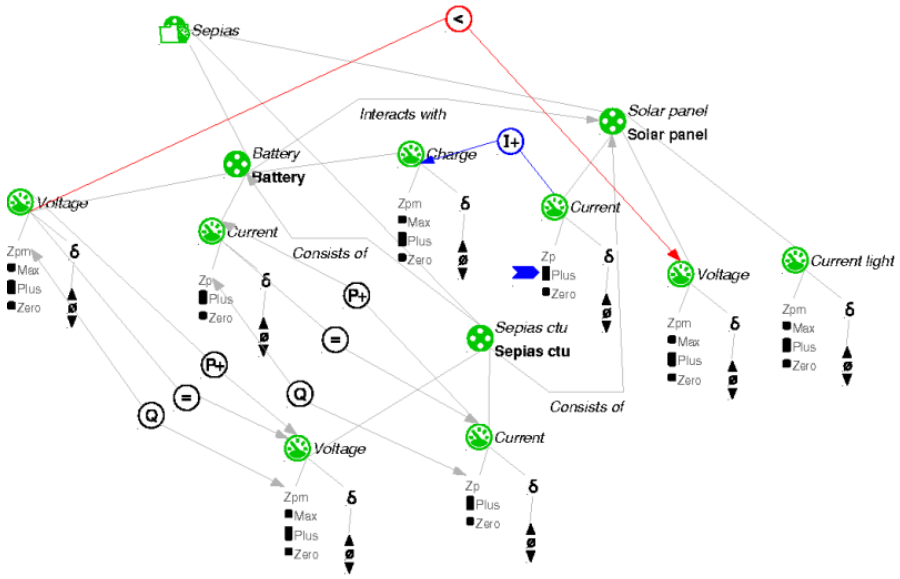


Figure 5: Container Tracking Unit.

Therefore, we define properties on the quantities, and use these properties to define test purposes. The idea is to formally specify both test purpose and test model as LTS, and then to derive test cases by computing the synchronous product of the specification and a test purpose, as initially proposed by Jard and Jeron(Jard and Jeron 2004). A test purpose describes some aspect of the specification that is of interest for testing. It is defined as a regular expression over symbols that represent properties of model quantities. A property set represents the conjunction of several properties. To remove redundancy one can define global properties, which can optionally be added to property sets. The set of all such property sets represents the set P of possible transition labels.

As an example, consider the property of a battery that its charge is greater than zero: $battery : charge > zero$, where $battery$ is the entity name and $charge$ the name of the quantity. This property denotes a value relation where *zero* is a value in the quantity's space. A second type of relation considers the δ of quantities using the operator dx. As an example, the property $battery : charge\ dx = min$ states that the battery charge decreases.

The regular expression that completely specifies the test purpose consists of the defined symbols and operators allowed in regular expressions. The equivalent deterministic automaton accepts all symbol sequences that lead to an accept state. This automaton represents an LTS with labels corresponding to properties of the test purpose. Suppose we are interested in the the cyclic occurrence of a property a, e.g., for three times and thereafter a path leading to property b. The regular expression $([\hat{}a] * a)\{3\}. * b$ describes such a test purpose. Although theoretically possible, our current implementation does not make use of reject states, which are used in LTS testing to consider only parts of the state space. As QR models have discrete, commonly only small value domains the models' state spaces usually are not very big, and reject states might not be necessary at all. Avoiding reject states in test purposes has the effect that test case execution cannot be inconclusive.

Once a test purpose is defined, we use the properties defined in this test purpose as symbolic labels for the transitions of our transition system. We annotate all transitions in the QR TS with labels, where the properties represented by the labels of a transition have to be satisfied in the target state of the transition. This augmentation of the QR TS is necessary for computing the synchronous product with a test purpose. Algorithm 1 describes the annotation process. The LTS is created simply by adding labels to the existing QR TS. The conversion from QR TS to LTS is done by iterating through the set of states. For each outgoing transition all symbols of a given test purpose are considered. Each symbol represents a set of conjunctively combined properties. If all properties in the set are satisfied, the current symbol is added as transition label. In addition, a property set can have a $global$ flag, which requires that all global properties have to be satisfied in addition to the properties of the set. Consequently, if the global properties are not satisfied and the $global$ flag is set, the algorithm rejects the currently considered symbol for the current transition. If the global properties are satisfied, the reserved symbol y is added as a transition label. The algorithm always terminates because the number of states in a QR transition system is finite (The domains of the QR variables are finite).

As an example, Figure 6 shows a QR TS consisting of three QR states. There are three symbols a, b, and c denoting three different properties on the state variables. The QR TS is la-

Algorithm 1 QRSTATEGRAPH2LTS()

```
1: for all states in QR TS do
2:     for all outgoing edges do
3:         for all symbols of the test purpose do
4:             get property set PS corresponding to
               current symbol
5:             get state S pointed to by current outgoing
               edge
6:             if S satisfies global_properties then
7:                 add label 'y' to current edge
8:             else if PS has global flag set to true
               then
9:                 continue with next symbol
10:            end if
11:            if S satisfies PS then
12:                add current symbol to current edge
13:            end if
14:        end for
15:    end for
16: end for
```

beled using these properties. As state $s1$ satisfies property a but not properties b and c, the transition from $s0$ to $s1$ is only labeled with a. State $s2$ satisfies all properties, therefore the transition from $s0$ to $s2$ is labeled with all symbols. If there is more than one symbol in a label, this is interpreted as the disjunction of the represented properties. Finally, $s3$ satisfies a and b and the transition from $s0$ to $s3$ is labeled accordingly.

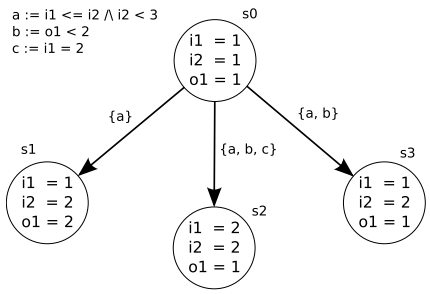


Figure 6: Labeled QR TS.

Test Case Generation with Test Purposes

With the converted LTS and a given test purpose we have two LTS with the same alphabet, and hence the synchronous product (Jard and Jeron 2004) can be computed; this is done with a Depth First Search (DFS) algorithm. Starting from the initial states of both LTS we match common labels on all outgoing edges between the current states both LTS are in. For every match found we get a state tuple by following both edges to states. If this tuple is a new state in the product LTS we add it with an edge. In addition the new state is pushed onto a stack. The algorithm terminates when the stack gets empty. In worst case this happens after all state tuples $Q^{M_1} \times Q^{M_2}$ have been visited. On average this algorithm performs better than the simple approach of considering all state tuples.

The synchronous product is a new LTS, to which Tarjan's algorithm as a framework (Thierry Jeron 2004) for determining the set of states leading to an accepting state is applied. It computes the set of strongly connected components while updating reachability information for the visited states. A state can reach an accepting state if itself or another state in the same SCC can reach an accepting state. A strongly connected component is defined as a subset of graph states, inside which every pair of states can reach each other via transitions to states inside the set. A directed graph with possible cycles partitioned in its SCCs is a directed acyclic graph (DAG) with the set of SCCs as nodes. The computed subgraph is called Complete Test Graph (CTG). For each state of the CTG, quantities can be mapped to actual values with the v and δ functions (see Section), to serve as test data and expected output.

Although input and output information is not contained in labels but in states, it is possible to adapt existing conformance relations to QR models. As an example, in order to determine the conformance of a system to a QR model we adapt the input/output conformance relation (ioco) (Jard and Jeron 2004):

$$i \; ioco \; s \leftrightarrow \forall \sigma \in traces(s) : out(i \; \textit{after} \; \sigma) \subseteq out(s \; \textit{after} \; \sigma)$$

where i is an IUT, s is the specification (the LTS derived from the QR model), $traces(s) = \{\sigma = \langle t_0, t_1, \ldots \rangle \mid t_0 = s_0 \wedge \forall i \geq 0 : t_i \in S \wedge (t_i, t_{i+1}) \in T\}$, and the set $s \; \textit{after} \; \sigma = \{s' \in S \mid \sigma = \langle t_0, ..., t_n \rangle \wedge t_i \in S, \; (t_n, s') \in T\}$. In this definition, $out(s)$ describes the state $s \in S$ with its output quantities $output(Q) \subset Q$, and their values and deltas:

$$out(s) =_{def} \bigcup \{(q, v(s,q), \delta(s,q)) \mid q \in output(Q)\}$$

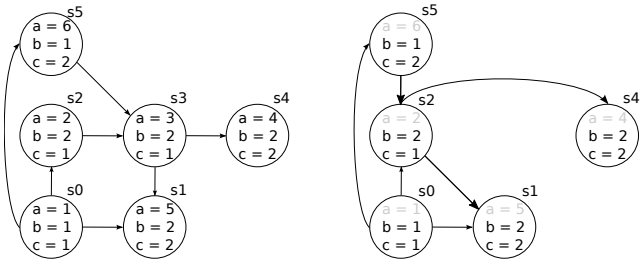
This relation considers all traces of the implementation that are also contained in the specification (QR TS). The values and deltas of all quantities of the implementation after a trace σ have to be a subset of the ones contained in the specification after the same trace. During test case execution when the implementation changes to a state not defined in the specification, meaning that after some trace the conformance relation is violated, we get a fail verdict.

State Space Minimization

As hidden quantities cannot be observed during test case execution they are not relevant in the context of test case generation. When we find two connected states that only differ in their non-relevant quantities we can merge them and update the unconnected edges of the removed state. Two states s_1 and s_2 of a TS $M = (S, T, s_0)$ are equivalent, if the following condition holds, where $rel(Q) \subset Q$ denotes the set of non-hidden quantities:

$$(s_1, s_2) \in T \wedge \forall q \in rel(Q) : \\ v(s_1, q) = v(s_2, q) \wedge \delta(s_1, q) = \delta(s_2, q)$$

Figure 7(a) shows an example state space with three states and three quantities. Assume that quantity a is hidden and thus not relevant. Consequently, states $s2$ and $s3$ are equivalent and can be merged. Figure 7(b) shows the result of this merge, with the updated transitions. Now a problem of nondeterminism arises in state $s2$, as the successor states $s1$ and $s4$ are equivalent but cannot be merged because there is no transition between them.



(a) Original TS. (b) Minimized TS.

Figure 7: State Space Minimization.

If all hidden quantities are constant the minimized CTG remains deterministic. This is because constant quantities cannot discriminate two states. Otherwise the CTG in terms of relevant quantities may become non-deterministic. At the end of the minimization we convert the possibly nondeterministic CTG to its equivalent deterministic CTG using the standard finite automaton technique.

Dealing with Controllability

Test cases have to be controllable. This means that on every node in the CTG where a decision between different inputs for the implementation is possible every branch taken leads to a new test case. As most systems like ours are not controlled by their environment, the implementation can be non-deterministic. Consequently, test cases are not linear sequences but transition systems that can handle alternative outputs. The set of input quantities splits a state's outgoing transitions into partitions where the input quantities all have same values and deltas. The implementation has the possibility to react with the according output quantity assignments in that partition. In the example of Figure 6 the set of input quantities $\{i1, i2\}$ splits the outgoing transitions of state $s0$ into the partitions $\{(s0, s2)\}$ and $\{(s0, s1), (s0, s3)\}$. For the second partition the implementation side decides which branch is taken next. The states of a test case have only outgoing transitions of one of its partitions which ensures controllability.

To achieve controllability we extract test cases from the CTG as follows: For every uncovered transition in the CTG we create a complete path by searching backwards to the start state and forward to an accept state. Then we traverse the path and add all parts of the CTG that are reachable considering the implementation's nondeterministic outputs to the test case TS. This approach returns test cases until all transitions in the CTG are covered.

Test Case Execution

A test case is a QR TS. All information needed for its execution is stored in its states. A QR state comprises a set of quantities with their current values and deltas. This abstract information has to be mapped to concrete quantity values, e.g., $voltage$ with $value = plus$ and $delta = positive$ is mapped to a time variant function like $voltage = start(voltage) + k \cdot t$. When the state is entered the concrete voltage value is updated regularly on some time step Δt as long as all state quantities fulfill the state's conditions. The $start(voltage)$ denotes the starting value of voltage when the state is entered. In this way we proceed with all input quantities and simultaneously observe the conditions of the output quantities. The behavior of the output quantities decides in which state of a branch we are in. If there is no matching state the implementation fails the test case. A state is left when it gets inconsistent with the observed output quantities. The current values of the output quantities have to be transferred to the successor state so that the according functions have initial values to compute further output values. When an accepting state is reached we get a pass verdict. The next section contains an example test case to illustrate this.

Demonstration and Results

For demonstration purposes we simplify the model of our system by replacing the solar panel with an exogenous sine quantity, which emulates a charging current changing with light conditions. With the load also emulated as a sine changing current, our model reduces to the battery model fragment shown in Figure 4. The temperature is defined as constant exogenous quantity. Simulation of the QR model results in a QR TS with 70 states and 228 transitions.

Assume we are interested in the behavior that leads to an empty battery. For this, we define the two property symbols *a* for $battery : charge > zero$ and *b* for $battery : charge = zero \wedge battery : charge\ dx = zero$. The regular expression describing the test purpose is *a*b*. This test purpose is fulfilled by paths containing any number of states where the battery charge is greater than zero (*a*), followed by a state where the battery charge is zero and the battery is not charging (*b*).

Next we define input- and output-quantities for our QR TS. The battery's current drawn by the system is an output. The system acquires the battery's voltage as indicator for the actual charge level as input. These two quantities are relevant for test case generation. From the test case view inputs and outputs of the specification are reversed. In a first step we augment the QR TS with labels and subsequently compute

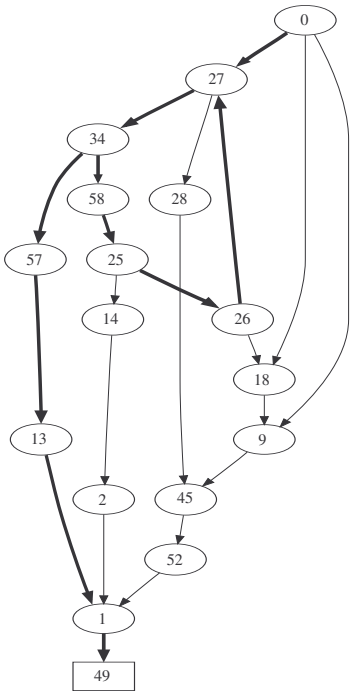


Figure 8: Test Case.

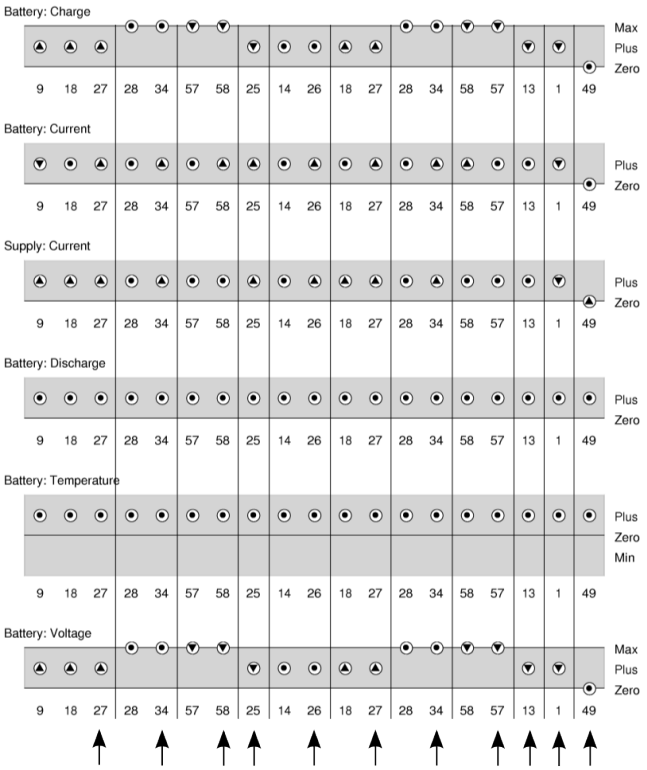


Figure 9: Value History: Test Case leading to an empty Battery.

the synchronous product with the test purpose. After that Tarjan's algorithm for computing reachability information is applied to the product LTS which reduces search space for later test case extraction to 59 states and 207 transitions. We minimize the obtained graph with regard to relevant quantities and ensure determinism. Now we have the CTG ready for test case extraction comprising 26 states and 72 transitions. As the state space is not big the computation time for all this steps is about 2 seconds.

The simplified model results in 16 test cases which cover 71 transitions in the CTG. One transition cannot be covered because when chosen during resolving a controllability conflict it never leads to an accepting state. In total, the 16 test cases cover 31% of the transitions of the QR TS.

Figure 8 shows an example test case for this test purpose. States are annotated with their state IDs as used in the QR TS. The test case has cyclic behavior in states {27, 34, 58, 25, 26}. Figure 9 shows one possible execution sequence through the test case leading to an empty battery. The vertical separation lines denote sets of outgoing transitions starting with the initial state on the left side. The arrows mark which transition has been taken during the execution.

Table 1: Characteristics of example test purposes.

Test Purpose	CTG	# TCs	tr coverage
battery is empty	55	55	82%
battery is full	53	38	65%
	Σ	93	90%

Table 1 depicts some characteristics for two test purposes considering $Battery : Charge$ and $Supply : Current$ as input and $Battery : Current$ as output, applied to the full example model. The CTG column specifies the number of states in the CTG, and $\#TC$ the number of test cases extracted from the CTG. The last column denotes the transition coverage for the test suites measured on the QR TS.

Related research

Tretmans (Tretmans 1996) described test case generation for Labeled Transition Systems (LTS). The paper focused on Input-Output-LTS (IOLTS) and introduced conformance relations for them. The proposed testing theory also deals with states where quiescence is allowed. Jard and Jeron (Jard and Jeron 2004) presented a tool for automatic conformance test generation from formal specifications. They used IOLTS as formal models and defined the *ioco* conformance relation for weak input enabled systems. Test cases are generated using defined test purposes.

Auguston et al. (Auguston, Michael, and Shing 2005) introduced the use of attributed event grammars for generating test-cases from environment models for reactive systems. In the paper the authors use the grammar for representing an event-based model. Possible execution traces of the model

form the test-cases. Insofar the underlying idea for test-case generation as described in this paper is very similar, but can be distinguished with respect to the underlying modeling language. Whereas Auguston et al. are using attributed event grammars, in this paper we are proposing the use of qualitative models for test-case generation.

Conclusions

In this paper we introduced an automated test case generation approach which relies on qualitative models. Qualitative models are well suited for modeling in the embedded systems domain. Especially when such systems strongly interact with their physical envirnoment this approach is a good choice.

Qualitative models represent all possible physical behaviors of systems and their environments, and can be used to find test cases which might not be considered when only using a system's specification. Similar to previous approaches we make use of test purposes in order to generate tests.

The following steps lead from a QR model to a test suite: (1) simulation of the QR model → QR TS, (2) conversion of the TS into a LTS according to a test purpose, (3) product of the system LTS with the test purpose, (4) minimization and ensuring determinism → CTG, and finally (5) extraction of controllable test cases from the CTG.

The first results indicate that useful test cases can be automatically generated from QR models. However, a more in-depth analysis is still required. In general, the approach is well suited when a physical model is available, e.g., in the embedded systems area. We are currently in the process of evaluating the approach on models derived from Matlab Simulink models via qualitative abstraction. First experiments indicate a sound test case execution, and resulting test cases exercise the interactions of a system under test with its environment. Future work will include application of the presented methods to larger models. This represents new challenges for the QR simulation tools, as for example in the case of weakly constrained Garp3 models. Here, the simulation output may become quite big (several thousand states) with many transitions, which leads to very long computation times (several days) and memory problems with current versions of Garp3.

Acknowledgements This work has been supported by the FIT-IT research project *Self Properties in Autonomous Systems(SEPIAS)* which is funded by BMVIT and the FFG, and the EU project ICT-216679, Model-based Generation of Tests for Dependable Embedded Systems (MOGENTES). The research herein is partially conducted within the competence network Softnet Austria (www.soft-net.at) and funded by the Austrian Federal Ministry of Economics (bm:wa), the province of Styria, the Steirische Wirtschaftsförderungsgesellschaft mbH. (SFG), and the city of Vienna in terms of the center for innovation and technology (ZIT).

References

Auguston, M.; Michael, J. B.; and Shing, M.-T. 2005. Environment behavior models for scenario generation and testing automation. In *International Workshop on Advances in Model Based Testing (A-MOST 2005)*. St. Louis, Missouri, USA: ACM.

Bouwer, A.; Liem, J.; and Bredeweg, B. 2005. *User Manual for Single-User Version of QR Workbench*. Naturnet-Redime, STREP project co-funded by the European Commission within the Sixth Framework Programme (2002-2006). Project no. 004074. Project deliverable D4.2.1.

Bredeweg, B.; Liem, J.; Bouwer, A.; and Salles, P. 2005. *Curriculum for learning about QR modelling*. Naturnet-Redime, STREP project co-funded by the European Commission within the Sixth Framework Programme (2002-2006). Project no. 004074. Project deliverable D6.9.1.

Bredeweg, B.; Bouwer, A.; Jellema, J.; Bertels, D.; Linnebank, F. F.; and Liem, J. 2006. Garp3 - a new workbench for qualitative reasoning and modelling. In *Proceedings of 20th International Workshop on Qualitative Reasoning (QR-06)*, 21–28.

Forbus, K. D. 1984. Qualitative process theory. *Artif. Intell.* 24(1-3):85–168.

Hierons, R. M.; Bogdanov, K.; Bowen, J. P.; Cleaveland, R.; Derrick, J.; Dick, J.; Gheorghe, M.; Harman, M.; Kapoor, K.; Krause, P.; Luettgen, G.; Simons, A. J. H.; Vilkomir, S.; Woodward, M. R.; and Zedan, H. 2008. Using formal specifications to support testing. *ACM Computing Surveys*.

Jard, C., and Jeron, T. 2004. TGV: theory, principles and algorithms. *International Journal on Software Tools for Technology Transfer (STTT)* 7(4):297–315.

Kleer, J. D., and Brown, J. S. 1984. A qualitative physics based on confluences. *Artif. Intell.* 24(1-3):7–83.

Thierry Jeron, P. M. 2004. Test generation derived from model-checking. *Computer Aided Verification: 11th International Conference, CAV'99. Trento, Italy, July 1999. Proceedings* 1633/1999(4):682.

Tretmans, J. 1996. Test generation with inputs, outputs, and quiescence. In *TACAS '96: Proceedings of the Second International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, 127–146. Springer-Verlag.

Wotawa, F. 2007. Generating test-cases from qualitative knowledge – preliminary report. In *Proceedings of the 21st Annual Workshop on Qualitative Reasoning*.

Order of Magnitude Reasoning in Modeling Moral Decision-Making

Morteza Dehghani Emmett Tomai Ken Forbus Matthew Klenk
Qualitative Reasoning Group, Northwestern University
2133 Sheridan Road, Evanston, IL 60201 USA
{morteza, etomai, forbus, m-klenk}@northwestern.edu

Abstract

We present a cognitive model of moral decision-making, MoralDM, which models psychological findings about utilitarian and deontological modes of reasoning. Current theories of moral decision-making extend beyond pure utilitarian models by relying strongly on contextual factors that vary with culture. In MoralDM, the impacts of secular versus sacred values are modeled via qualitative reasoning, using an order of magnitude representation. We present a simplified version of ROM(R) (Dague, 1993) and discuss how it can be used to capture people's degree of quantity sensitivity. MoralDM uses a combination of first-principles reasoning and analogical reasoning to determine consequences and utilities of moral judgments. A natural language system is used to produce formal representations for the system from psychological stimuli, to reduce tailorability. We compare MoralDM against psychological results in moral decision-making tasks and show that its performance improves with experience.

Introduction

While traditional models of decision-making in AI have focused on utilitarian theories, there is considerable psychological evidence that these theories fail to capture the full spectrum of human decision-making. In particular, research on moral reasoning has uncovered a conflict between normative outcomes and intuitive judgments. This has led some researchers to propose the existence of deontological moral rules, which could block utilitarian motives. Consider the starvation scenario (from Ritov & Baron 1999) below:

> A convoy of food trucks is on its way to a refugee camp during a famine in Africa. (Airplanes cannot be used.) You find that a second camp has even more refugees. If you tell the convoy to go to the second camp instead of the first, you will save 1000 people from death, but 100 people in the first camp will die as a result.
>
> Would you send the convoy to the second camp?

While the utilitarian decision would send the convoy to the second camp, participants were more likely to choose to send the convoy to the first camp. Baron and Spranca (1997) suggested the existence of *sacred values*, which are not allowed to be traded off, no matter what the consequences. Further, they suggest that these sacred values "arise out of deontological rules about actions rather than outcomes". In our example, given that life is a sacred value, people often refuse to take an action which would result in taking lives.

This paper describes a cognitively motivated model of moral decision-making, called MoralDM, which operates in two modes of decision-making: utilitarian and deontological. MoralDM models the different impacts of secular versus sacred values via qualitative reasoning, using an order of magnitude representation. To reduce tailorability, a natural language understanding system is used to semi-automatically produce formal representations from psychological stimuli re-rendered in simplified English. MoralDM combines first-principles reasoning and analogical reasoning to implement rules of moral decision-making and utilize previously made decisions. We evaluate our system by comparing it with established psychological results and by examining how the performance of system changes as a function of the number of available cases.

We begin by summarizing relevant psychological results on quantity insensitivity and how an order of magnitude formalism can be used to capture this phenomenon. Next, we describe MoralDM and how it works. Then we show that MoralDM can account for results from two psychological studies, and that its performance can be improved by accumulating examples. Finally, we discuss related and future work.

Decision-Making and Quantity Insensitivity

In the presence of sacred values, people tend to be less sensitive to outcome utilities in their decision-making. This results in decisions which are contrary to utilitarian models. We claim that this can be accounted for using an existing qualitative reasoning formalism. After summarizing the relevant moral decision-making findings, we present a simplified version of Dague's (1993) ROM(R) qualitative order of magnitude formalism which we use to capture these results.

Sacred or protected values concern acts and not outcomes. When dealing with a case involving a protected value, people tend to be concerned with the nature of their action rather than the utility of the outcome. Baron and Spranca (1997) argue that when dealing with protected

values people show insensitivity to quantity. That is, in trade-off situations involving protected values, they are less sensitive to the outcome utilities of the consequences. The amount of sensitivity (or insensitivity) towards outcomes vary with the context. Lim and Baron (1997) show that this effect varies across cultures.

In addition to contextual factors, the causal structure of the scenario affects people's decision-making. Waldmann and Dieterich (2007) show that people act more utilitarian, i.e., become more sensitive to the outcome utilities, if their action influences the patient of harm rather than the agent. They also suggest that people act less quantity sensitive when their action directly, rather than indirectly, causes harm. Bartels and Medin (2007) argue that the agent's sensitivity towards the outcome of a moral situation is dependent on the agent's focus of attention.

We model quantity sensitivity by using Dague's (1993) ROM(R) qualitative order of magnitude formalism. Order of magnitude reasoning is a form of commonsense reasoning which provides the kind of stratification that seems necessary for modeling the impact of sacred values on reasoning. Raiman (1991) uses the analogy of a coarse balance to describe the intuitions behind order of magnitude reasoning: a course balance can weigh quantities with more or less precision. This precision level depends on the order of magnitude scale used to map quantities onto course values. He uses two granularity levels *Small* and *Rough* to build a multitude of order of magnitude scales. These two granularity levels provide three qualitative relations between quantities which have been formally defined in FOG (Raiman, 1991). Both O(M) (Mavrovouniots and Stephanapoulos, 1987) and ROM(K) (Dague, 1993) are attempts to provide a more comprehensive order of magnitude formalism.

ROM(R), the mapping of ROM(K) onto $\mathcal{R}$, is the only system that guarantees validity in $\mathcal{R}$. Some order of magnitude representations (e.g. FOG) do not allow values at different levels to ever be comparable. One of the features of ROM(R) is that it includes two degrees of freedom, k_1 and k_2, which can be varied to capture differences in quantity sensitivity. Dague defines four classes of relationship between two numbers: "close to", "comparable to", "negligible with respect to" and "distant from". While FOG and O(M) fail to capture gradual change, the overlapping relations in ROM(K) allow a smooth, gradual transition between the states.

Although for engineering problems two degrees of freedom and four relations is quite useful, we believe for the task that we are interested in one degree of freedom and three binary relations are more plausible. Therefore, we implemented a simplified version of ROM(R) using one degree of freedom, k, resulting in three binary relations; almost equal, greater than, and orders of magnitude different. These three classes can be computed using the following rules:

- $A \approx_k B \Leftrightarrow |A-B| \leq k * \text{Max}(|A|,|B|)$
- $A <_k B \Leftrightarrow |A| \leq k * |B|$
- $A \neq_k B \Leftrightarrow |A-B| > k * \text{Max}(|A|,|B|)$

These relations respectively map to "close to", "greater than" and "distant from". k can take any value between 0 and 1. Figure 1 demonstrates the interval landmarks of the system. The analog in the above system of the parameter ε of ROM(K) depends on the value of k. When $k < ½$, ε is $k/(1 - k)$, and, when $k \geq ½$, ε is $(1 - k)/ k$. Quantity

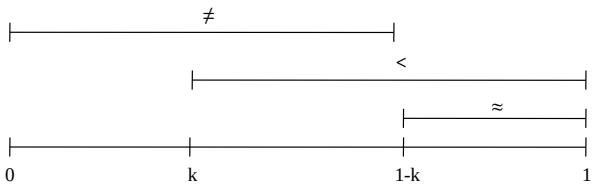


Figure 1: Interval Landmarks

sensitivity can be varied by changing k: setting k to $k - \varepsilon$ shifts the relationship between the compared values and moves it from ≈ to < or from < to ≠ resulting in higher quantity sensitivity. Depending on the sacred values involved and the causal structure of the scenario, we vary k to capture sensitivity towards the utility of the outcome.

MoralDM

Our model of moral decision-making, MoralDM, has been implemented using the FIRE reasoning engine and underlying knowledge base. The knowledge base contents are a 1.2 million fact subset of Cycorp's ResearchCyc knowledge base[1], which provides formal representations about everyday objects, people, events and relationships. The KB also includes representations we have developed to support qualitative and analogical reasoning. The scenarios, decisions and rule sets used in MoralDM are all represented uniformly and stored in this KB.

MoralDM operates in two mutually exclusive modes of decision-making: utilitarian and deontological. If there are no sacred values involved in the case being analyzed, MoralDM applies traditional rules of utilitarian decision-making by choosing the action which provides the highest outcome utility. On the other hand, if MoralDM determines that there are sacred values involved, it operates in deontological mode and becomes less sensitive to the outcome utility of actions, preferring inaction to actions that would cause harm.

To solve a given moral decision-making scenario, MoralDM begins by using a natural language understanding system, to automatically translate simplified English scenarios into predicate calculus. Given this representation, the Orders of Magnitude Reasoning (OMR) module calculates the relationship between the utility of each choice. Using the outcome of OMR, MoralDM utilizes a hybrid reasoning approach consisting of a First-

[1] http://research.cyc.com

Principles Reasoning (FPR) module and an Analogical Reasoning (AR) module to arrive at a decision. FPR suggests decisions based on rules of moral reasoning. AR compares a given scenario with previously solved decision cases to suggest a course of action. We believe using hybrid reasoning improves the robustness of the system and provides a more cognitively plausible approach to decision-making. Figure 2 depicts the MoralDM architecture.

FRP and AR work in parallel and complement each other by providing support (or disagreement) for a decision. If both succeed and agree, the decision is presented. When one module fails to arrive at a decision, the answer from the other module is used. If the modules do not agree, the system selects FPR's choice. If both fail, the system is incapable of making a decision. After a decision is made for a given scenario, it can be stored in the case library for future use. This enables the system to make decisions in more scenarios as it accumulates experience. Next, we discuss each module in detail.

Explanation Agent NLU

Our inputs are dilemmas from the psychological literature, expressed in natural language. To construct formal representations of these stimuli, we extended the Explanation Agent Natural Language Understanding system (EA NLU, Kuehne and Forbus, 2004). Unrestricted automatic natural language understanding is currently beyond the state of the art. Consequently, EA NLU uses a controlled language and operates semi-automatically, enabling experimenters to interactively translate natural language stimuli into simplified syntax and guide the generation of predicate calculus. This practical approach allows us to broadly handle syntactic and semantic ambiguities and to build deep formal representations suitable for complex reasoning. This is a significant advantage over having experimenters construct representations by hand for two reasons. First, constructing representations by hand is very time-consuming and requires substantial expertise. Second, hand-coding increases tailorability, i.e., the possibility that representation choices were made to get a particular example to work, as opposed to being uniform, independently motivated conventions. Since EA NLU is used by multiple projects and relies on an off-the-shelf knowledge base, tailorability is greatly reduced.

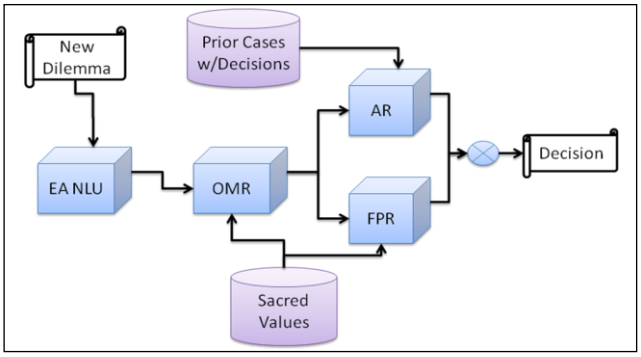


Figure 2: MoralDM Architecture

EA NLU uses Allen's bottom-up chart parser (Allen, 1995) in combination with the COMLEX lexicon (Macleod *et al.* 1998) and a simplified English grammar (Kuehne and Forbus, 2004). The parser uses subcategorization frames from ResearchCyc for word and common phrase semantics. Each frame represents a case for the term encoded as predicate calculus with syntactic/semantic role variables. Roles are filled during the parsing process. Frames are filtered according to both case constraints and syntactic requirements explicitly included in the frame.

Sentences within a stimulus are parsed separately. The resulting parse trees are presented, together with the semantic frames they entail, to the user in an interactive interface. The user can selectively include or exclude trees as well as individual frames. These selections serve as input to a transformation process using dynamic logic principles from Discourse Representation Theory (DRT) (Kamp & Reyle 1993) to construct a description of the sentence content. This description supports numerical and qualitative quantification, negation, implication, modal embedding and explicit and implicit utterance sub-sentences. Anaphoric references are resolved via selectional restrictions from the Cyc ontology guiding sentence attachment and thereby the integration into the representation of the discourse as a whole.

```
A convoy of trucks is transporting food
to a refugee camp during a famine in
Africa. 1000 people in a second refugee
camp will die. You can save them by
ordering the convoy to go to that
refugee camp. The order will cause 100
people to die in the first refugee camp.
```

Figure 3: Starvation scenario in simplified English

Figure 3 contains the controlled language for the starvation scenario. Given these statements, EA NLU identifies events of transporting, famine, dying (1000 people), saving, ordering, going and dying (100 people) together with the two quantified sets of people, the convoy, food, two refugee camps and the proper name Africa. There is also an explicit reference to the listener, "you". Figure 4 contains the frame-based interpretation of the order.

Causal links are explicitly stated between the order and the saving and the order and the second set of deaths. The abstraction of saving drives inferential attention to events in the description that the beneficiary may be being saved from. The expected future modality of the first set of deaths makes it a reasonable candidate. Based on the possible modality of the saving/ordering sequence,

```
(isa order131049 Ordering-CommunicationAct)
(performedBy order131049 you128898)
(recipientOfInfo order131049 convoy127246)

(infoTransferred order131049
 (and
  (isa refugee-camp129739 RefugeeCamp)
  (isa convoy127246 Convoy)
  (isa go129115 Movement-TranslationEvent)
  (primaryObjectMoving go129115 convoy127246)
  (toLocation go129115 refugee-camp129739)))
```

Figure 4: Predicate calculus for ordering

combined with the use of the explicit reference to the listener, the system infers an abstraction of choice being presented with known consequences resulting from both action and inaction. Figure 5 contains the inferred abstraction of choice and its causal consequences.

```
(isa Sel131949 SelectingSomething)
(choices Sel131949 order131049)
(choices Sel131949 Inaction131950)
(causes-PropSit
  (chosenItem Sel131949 Inaction131950)
  die128829)
(causes-PropSit
  (chosenItem Sel131949 order131049)
  save128937)
```

Figure 5: Predicate calculus for the choice presented

Order of Magnitude Reasoning Module

The inputs to OMR include the sacred values for the culture being modeled and the causal structure of the scenario. Using the predicate calculus produced by EA NLU, OMR calculates the expected utility of each choice by summing the utility of its consequences. For each consequence of a choice, OMR uses its rules to ascertain if the outcome is a positive or negative outcome, and to identify any sets whose cardinality matters in the decision (e.g., number of people at risk).

After computing utilities, OMR selects a k value based upon the context of the scenario. Assuming that the relationship between the utilities, a and b, are "comparable", MoralDM sets k to $1 - (|\ a\ /\ b\ |)$. This results in the relationship between the utilities falling within $<$, right between $\neq$ and $\approx$ (Fig 1). If the decision involves a sacred value for the modeled culture, setting k to $k + \varepsilon$ shifts the relationship between utilities from greater than to close to, resulting in the system being less sensitive to the numeric utility of the outcome. On the other hand, if the there are no sacred values involved, the system substitutes k with $k - \varepsilon$ thereby making the system more quantity sensitive to the computed utilities. In addition to sacred values, the causal structure of the scenario affects k. OMR checks to see if the scenario contains patient intervention or agent intervention. It uses low quantity insensitivity for the first case and high otherwise, consistent with psychological findings (Waldmann and Dieterich 2007). The system also checks for direct versus indirect causation. In the case of indirect causation, a higher degree of insensitivity is applied.

Returning to the starvation scenario, there are two choices: ordering and inaction. For ordering, there are two consequences, 1000 people in the second camp will be saved and 100 people in the first camp will die. Consulting the KB, the system determines that dying has negative utility and saving positive, resulting in a choice utility of 900 for the ordering choice. Using the same procedure, the utility for inaction is calculated to be -900. Using the formula given above, k is initially set to 0 with $\varepsilon = 1$. Given that both choices involve agent intervention and indirect causation, there are no structural differences between the two choices. Therefore, the k value is set solely by the existence of sacred values. In this case, causing someone to die is a sacred value resulting in k being set to $k + \varepsilon = 1$, therefore causing the system to act less quantity sensitive. Using ROM(R), the relationship between the utilities of the two choices is calculated to be $\approx$. On the other hand, if there had not been a sacred value, the value of k would have remained 0 causing the relationship between the utilities to be $\neq$. These utilities, 900 and -900, and the computed relationship, $\approx$, are provided to FPR and AR.

First-Principles Reasoning Module

Motivated by moral decision-making research, FPR makes decisions based upon the following factors: the orders of magnitude relationship between utilities, sacred values, computed utilities, and action vs. inaction. FPR uses three methods for making decisions. First, the utilitarian method, which selects the choice with the highest utility, is invoked when the choice does not involve a sacred value. Second, in situations with sacred values and without an order of magnitude difference between outcomes, the pure-deontological method selects the choice that does not violate a sacred value. Third, the utilitarian-deontological method operates when the scenario contains sacred values and an order of magnitude difference between outcomes, selecting the choice with the higher utility. Therefore, the pure-deontological method is the only method that makes decisions that violate utilitarian norms.

In the starvation scenario, there is a sacred value, people dying, and no order magnitude difference between the utility of the two choices. Therefore, the system uses the pure deontological method to select the inaction choice.

These methods are mutually exclusive, returning at most one choice per scenario. Given the breadth of moral reasoning scenarios, the rules implementing FPR are not complete. Therefore, FPR necessarily fails on some scenarios. These cases highlight the need for the hybrid-reasoning approach taken in MoralDM. The resulting choice is compared with the results of the analogical reasoning module of MoralDM.

Analogical Reasoning Module

An important role that analogy plays in decision-making is framing the situation. When making a choice, decision makers frequently use past experiences and draw inferences from their previous choices (Markman and Medin, 2002). For more details and examples about the use of analogy in decision-making please refer to Dehghani et al. (2008a). To model analogy in decision making, we use the Structure-Mapping Engine (SME) (Falkenhainer *et al.* 1989), a computational model of similarity and analogy based on Gentner's (1983) structure mapping theory of analogy in humans. SME operates over structured representations, consisting of entities, attributes of entities and relations. Given two descriptions, a *base case* and a *target case*, SME aligns their common structure to find a mapping between the cases. This mapping consists of a set of correspondences between entities and expressions in the two cases. SME produces mappings that maximize *systematicity*; i.e., it prefers mappings with higher-order relations and nested relational structure. The *structural evaluation score* of a mapping is a numerical measure of similarity between the base and target. SME identifies elements in the base that fail to map to the target and uses the common relational structure to calculate *candidate inferences* by filling in missing structures in target.

Running concurrently with FPR, AR uses comparisons between new cases and previously solved cases to suggest decisions. When faced with a moral decision scenario, AR uses SME to compare the new case with every previously solved scenario in its memory. The similarity score between the novel case and each solved scenario is calculated using SME by normalizing the structural evaluation score against the size of the scenario. If this score is higher than a certain threshold and both scenarios contain the same order of magnitude relationship between outcome utilities, then the candidate inferences are considered as valid analogical decisions. If the scenarios have different orders of magnitude relationships, it is likely that a different mode of reasoning should be used for the target scenario and AR rejects the analogical inference. After comparing against all of the solved scenarios, AR selects the choice in the new scenario with the highest number of analogical decisions. In the case of a tie, AR selects the choice with the highest average similarity score supporting it. Because analogical alignment is based upon similarities in structure, similar causal structures and/or sacred values align similar decisions. Therefore, the more structurally similar the scenarios are, the more likely the analogical decision is going to be the correct moral one.

Returning to our starvation example, AR can solve this decision problem through an analogy with a traffic scenario given below, in which the system chose to not transfer funds:

> A program to combat accidents saves 50 lives per year in a specific area. The same funds could be used to save 200 lives in another area, but the 50 lives in the first area would be lost.
>
> Do you transfer the funds?

The analogical decision is determined by the candidate inferences where the decision in the base, inaction, is mapped to the choice in the target representing inaction. Because the traffic scenario contains the same the order of magnitude relationship, almost equal, as in the starvation scenario, the system accepts the analogical decision.

Evaluation

We evaluated MoralDM by running it on 8 moral decision-making scenarios taken from two psychology studies (Waldmann and Dieterich 2007; Ritov and Baron 1999). In all the scenarios used, traditional utility theories fail to predict subjects' responses, as often the subjects choose the choice which provides a smaller overall outcome utility. We compare MoralDM's decisions to subjects' responses in these experiments. If the decision of MoralDM matched those of the subjects, as reported by the authors, we consider it a correct choice.

EU NLU translated all 8 cases into predicate calculus. MoralDM made the correct choice in each of the scenarios using the result from FPR. This illustrates MoralDM's ability to do complex reasoning from natural language input and provides evidence for its psychological fidelity.

One of the more difficult aspects in building the FPR module is the number of rules to handle the broad range of situations covered in moral decision making. The AR module is capable of making moral decisions in situations when gaps in the KB or rule set would prevent the FPR module from coming up with an answer. Therefore, we evaluated the AR module independently of the FPR module, to answer two questions: (1) Can we use analogy to do moral decision-making from natural language input? (2) How is AR performance affected as the number of previously solved cases stored in memory increases?

Given the 8 solved scenarios, we created case libraries of every combination of these scenarios. This provided us with 254 different case libraries (8 of size 1, 28 of size 2, 56 of size 3…). Then, with each case library, we tested the AR module by running it on each of the scenarios not in the case library. So for each of the 8 libraries of size 1, the test consisted of 7 decision scenarios for a total of 56 decision scenarios.

Figure 6 shows the performance of AR as a function of the number of available cases. There is a monotonic increase in the number of correct answers as the size of the library increases (Pearson's $r = .97$, $p < .0001$). Also, there is a significant decrease in the number of cases where AR does not come up with an answer ($r = -.95$, $p < .001$). The number of incorrect decisions changes insignificantly from 18% to 25% ($r = .53$, $p < .22$). The statistics reported have been computed by comparing each series against the size of the case library.

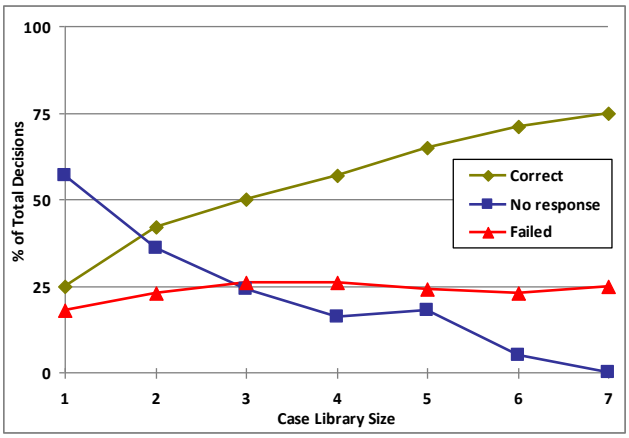


Figure 6: Analogical reasoning results

The results of these evaluations are very encouraging. First and foremost, our system matches human behavior on 8 decision-making scenarios provided in natural language. In addition to this result, we also found that there was a significant improvement in AR module performance as the number of cases in MoralDM's memory increased. For a more comprehensive analysis of evaluation of each of the modules please refer to Dehghani et al. (2008b).

Related Work

Reasoning with orders of magnitude is a form of commonsense reasoning which is mostly suitable when complete quantitative information is not available or when tackling problems involving complex physical systems. Order of magnitude reasoning has been used in several engineering tasks (e.g. Dague, 1994; Mavrovouniots and Stephanapoulos, 1988; Dague, Deves and Raiman 1987).

Several research projects have focused on building ethical advisors. The MedEthEx system uses ILP techniques to learn decision principles from training cases (Anderson *et al.* 2006). Mclaren's Truth-Teller and SIROCCO systems (2005) use case-based reasoning to highlight relevant ethical considerations and arguments to a human user. Like them, we use prior cases, but to guide the system's own reasoning, rather than give advice. They also were not designed to model the effects of sacred versus secular values that MoralDM captures.

Computational models of cultural reasoning are receiving increasing attention. For example, the CARA system (Subrahmanian *et al.* 2007) is part of a project to "understand how different cultural groups today make decisions and what factors those decisions are based upon". CARA uses semantic web technologies and opinion extraction from weblogs to build cultural decision models consisting of qualitative rules and utility evaluation. While we agree that qualitative reasoning must be integrated with traditional utility evaluation, we also believe that analogy plays a key role in moral reasoning. Moreover, we differ by evaluating our system against psychological studies, which helps ensure its judgments will be like those that people make.

Our combination of analogical and first-principles reasoning is inspired in part by Winston's (1982) use of both precedents and rules to reason about a situation. His work was hampered by the lack of off-the-shelf large-scale knowledge bases, and the technologies for NLU and analogical reasoning have improved since then.

Our use of simplified English is inspired by both CMU's KANT project (cf. Mitamura & Nyberg 1995) and Boeing's controlled language work (cf. Clark *et al.* 2005).

Conclusions and Future Work

MoralDM uses qualitative modeling to reason about utilities, capturing the differences between sacred and secular values via an order of magnitude representation. It uses a combination of first-principles logical reasoning and analogical reasoning to determine the utility of outcomes and make decisions based on this information. The hybrid approach produces answers in a wider range of circumstances than either alone. Natural language input of scenarios, in simplified English, reduces tailorability, a key problem in cognitive simulation research. We showed that MoralDM can be used to model psychological results from two studies. While there is still more to be done, we think MoralDM represents an important step in computational modeling of moral decision-making.

We plan to pursue several lines of investigation next. First, we plan to extend the valuation rules to model different cultures, based on existing collaborations with cognitive psychologists and anthropologists. This will require extending the first-principles reasoning rules to cover a broader range of scenarios. Constructing these rules is a time consuming and error prone process. One alternative is to automatically extract rules by generalizing over previously made decisions. By focusing on decisions from a specific culture, we can explore automatic model construction for making novel predictions about the behavior of a certain group (Dehghani et al. 2007). Second, we plan to extend the range of EA NLU coverage to handle a wide range of cultural stories. This will enable us to create story libraries for different cultural groups, and translate transcripts from interview data more easily. Third, we plan to incorporate a cognitively plausible model of similarity-based retrieval, MAC /FAC (Forbus et al., 1995), to make analogical reasoning more scalable as the story library grows. Finally, we plan to test MoralDM on a wider range of problems, using data gathered from participants from multiple cultural groups.

References

Allen, J. F. 1995. *Natural Language Understanding*. (2nd ed). Redwood City, CA.: Benjamin/Cummings

Anderson, M., Anderson, S., and Armen, C. 2006. An Approach to Computing Ethics. *IEEE Intelligent Systems*. 21(4): 56-63.

Bartels, D. M. & Medin, D. L. 2007. Are Morally-Motivated Decision Makers Insensitive to the Consequences of their Choices? *Psychological Science,* 18, 24-28.

Baron, J., and Spranca, M. 1997. Protected Values. *Organizational Behavior and Human Decision Processes* 70: 1–16.

Clark, P., Harrison, P., Jenkins, T., Thompson, J., and Wojcik, R. 2005. Acquiring and Using World Knowledge using a Restricted Subset of English. In *Proceedings of The 18th International FLAIRS Conference*.

Dauge, P. 1993. Symbolic Reasoning with Relative Orders of Magnitude. In *Proceedings of the 13th IJCAI.*

Dauge, P. 1993. Numeric Reasoning with Relative Orders of Magnitude. In *Proceedings of the 7th International Workshop on Qualitative Reasoning*.

Dague, P., Deves, P., and Raiman, O. 1987. Troubleshooting: when Modeling is the Trouble, In *Proceedings of AAAI87 Conference*, Seattle.

Dague, P. 1994. Model-based Diagnosis of Analog Electronic Circuits. *Annals of Mathematics and Artificial Intelligence*,11, 439-492

Dehghani, M., Tomai, E., Forbus, K., Iliev, R., Klenk, M. (2008a). MoralDM: A Computational Modal of Moral Decision-Making. To appear in Proceedings of the 30th Annual Conference of the Cognitive Science Society (CogSci), Washington, D.C

Dehghani, M., Tomai, E., Forbus, K., Klenk, M. (2008b). An Integrated Reasoning Approach to Moral Decision-Making. To appear in Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI). Chicago, IL.

Dehghani, M., Unsworth, S., Lovett, A., Forbus. K. (2007) Capturing and Categorizing Mental Models of Food Webs using QCM. *21st International Workshop on Qualitative Reasoning*. Aberystwyth, U.K.

Falkenhainer, B., Forbus, K. and Gentner, D. 1989. The Structure-Mapping Engine: Algorithms and Examples. *Artificial Intelligence,* 41: 1-63.

Forbus, K., Gentner, D. and Law, K. 1995. MAC/FAC: A Model of Similarity-based Retrieval. *Cognitive Science*, 19(2), 141-205.

Gentner, D. 1983. Structure-Mapping: A Theoretical Framework for Analogy. *Cognitive Science*, 7: 155-170.

Kuehne, S. and Forbus, K. 2004. Capturing QP-Relevant Information from Natural Language Text. *Proceedings of QR04*.

Lim, C. S, and Baron, J. 1997, Protected values in Malaysia Singapore, and the United States. Manuscript, Department of Psychology, University of Pennyslvania.

Markman, A and Medin, D.L. 2002. *Decision Making.* Stevens Handbook of Experimental Psychology, 3rd edition: Volume 2, Memory and Cognitive Processes. New York: Wiley.

Macleod, C., Grishman, R., and Meyers, A. 1998. COMLEX Syntax Reference Manual, Version 3.0. Linguistic Data Consortium. University of Pennsylvania: Philadelphia, PA.

Mavrovouniotis M. L. and Stephanopoulos, G. 1987, Reasoning with orders of magnitude and approximate relations, In *Proceedings of AAAI 1987 Conference*, Seattle.

Mavrovouniotis M. L. and Stephanopoulos, G. 1987, Order of Magnitude Reasoning in Process Engineering, Computer. chem. Engng. 12, 1988 .

McLaren, B. 2005. Lessons in Machine Ethics from the Perspective of Two Computational Models. In Anderson, M. *et al.* (Eds.) *Machine ethics: Papers from the AAAI Fall Symposium*, Technical Report FS-05-06. Menlo Park, CA.

Mitamura, T., & Nyberg, E. H. 1995. Controlled English for Knowledge-Based MT: Experience with the KANT System. In *Proceedings of 6th International Conference on Theoretical and Methodological Issues in Machine Translation*, Leuven, Belgium.

Raiman, O. 1991. Order of magnitude reasoning. *Artificial Intelligence* 51.

Ritov, I. and Baron, J. 1999. Protected Values and Omission Bias. *Organizational Behavior and Human Decision Processes*, 79(2): 79-94.

Subrahmanian, VS., Albanese, M., Martinez, V., Nau, D., Reforgiato, D., Simari, G., Sliva, A., and WIlkenfeld, J. 2007. CARA: A Cultural Adversarial Reasoning Architecture. *IEEE Intelligent Systems.* 22(2): 12-16.

Waldmann, M. R., and Dieterich, J. 2007. Throwing a Bomb on a Person Versus Throwing a Person on a Bomb: Intervention Myopia in Moral Intuitions. *Psychological Science*, 18 (3), 247-253.

Winston, P.H. 1982. Learning New Principles from Precedents and Exercises. *Artificial Intelligence* 19(3), 321-350.

A qualitative model on sexual behaviour: mate guarding and extra-pair copulation in birds

R. I. Dias[a] and P. Salles[b]

[a] Graduate Program in Ecology, University of Brasília, Brasília, Brazil
[b] Institute of Biological Sciences, University of Brasília, Brasília, Brazil
{raphaeligor, psalles}@unb.br

Abstract

Extra-pair paternity is widespread among avian species, showing high inter and intra-specific variation both in frequency and in typical behaviours. These variations are the result of individual, ecological and phylogenetic influences upon the behavioural searching pattern for extra-pair copulations. A qualitative reasoning model was developed to show how decisions made by females and males could affect the occurrence of extra-pair paternity and fitness. The model demonstrates that population density and the genetic quality of the male with whom the female is mated influences female predisposition to search for extra-pair copulations. Accordingly, high-quality males should have a higher number of within and extra-pair young than low quality males. The model shows that since the interests of males and females are similar, both sexes may achieve an increment in their fitness.

1. Introduction

Monogamy in birds is conceptually defined as a web of complex interactions and conflicts of interest between paired males and females (Westneat and Stewart, 2003). The study of this subject has suffered significant changes with the application of molecular tools, showing it to be less simple than was initially assumed, with a number of hypotheses being suggested to explain the wide variation (0-70%) in the rate of extra-pair fertilization (EPF) among species. Variation in EPF levels must be the result of different ecological, phylogenetic and individual constraints that should have influenced the evolution of this behaviour. The use of a qualitative approach may play an important role in helping to understand the causal relations of this process.

This paper presents a simulation model that intends to increase understanding about the role of individual features and behavioural factors associated with the occurrence of extra-pair paternity (EPP), based on qualitative reasoning (QR) techniques (Weld and de Kleer, 1990). QR models contribute to ecological theory development by determining the logical consistency and the consequences of long and complex chains of ecological reasoning and rapid assessment of assumptions, hypotheses and other ideas (Rykiel, 1989). In the context of EPF studies, QR techniques are useful for establishing causal relations and predicting the system's behaviour using incomplete knowledge.

The model presented here attempts to answer questions such as: *What is the effect of mate quality on female pursuit for extra-pair copulations? Is mate guarding an effective strategy for decreasing paternity loss? Does male genetic quality affect male and female individual fitness?*

2. Extra-pair paternity in birds

Despite the variation in the frequency of EPP among species, the determinants of this behaviour are poorly understood. Moreover, it is difficult to separate cause and effect relations, as some variables may have a causal effect on EPP, whereas others may result from the consequences of this behaviour (Westneat and Stewart, 2003). A number of hypotheses have been proposed to explain how such reproductive strategies evolved. These possible explanations include breeding synchrony, need for paternal care, rate of adult mortality, and ecological explanations, such as the relation between breeding density and levels of EPP (Griffith *et al*. 2002). The breeding density hypothesis states that the proximity to neighbours would increase accessibility to mates for extra-pair copulation (EPC), and would enhance opportunities to evaluate available mates (Birkhead and Møller, 1992).

Considering a bird's fitness (i.e. a measure of the individual's ability, relative to others, to leave viable offspring), it is possible for males to increase their reproductive gain without additional parental investment by seeking for EPCs (Birkhead and Møller, 1992). However, the reproductive gain for females is less obvious, because their reproductive capacity is limited by the

number of eggs they can produce. As sexual promiscuity does not increase the number of female offspring, the role of female behaviour in determining the level of EPP is unclear.

According to the *good genes hypothesis*, if males differ in genetic quality, females paired with low quality males are compensated when searching for EPF with males of higher quality, because this behaviour would improve their offspring's survival and reproductive chances (reviewed in Jennions and Petrie, 2000). Nevertheless, from the male's perspective, EPC's are only valuable for those males that can obtain extra-pair fertilizations, but are unfavorable for those males that lose paternity in their own nest. Thus, males must prevent the occurrence of EPCs by their own females to avoid the future cost of rearing offspring sired by other males. Thus, males develop counterstrategies to avoid, or at least interfere with extra-pair behaviour of their females (Trivers, 1972). The more commonly used for the males is mate guarding, which consists of following the female during her fertile period, to prevent them from copulating with other males (Birkhead and Møller, 1992). On the other hand, males also try to increase their fitness through EPCs, but with this, they incur the risk of losing paternity by leaving their female unguarded, since males cannot maximize within and extra-pair paternity simultaneously (Hasselquist and Bensch, 1991). The result is that males must decide how much time to allocate to these two mutually exclusive activities, and must adjust mate guarding behaviour according to the risk of being cuckolded.

Males and females must follow adaptive rules of differential allocation, where mating and reproductive effort depend on the attractivity, or quality of their mates as well as their own quality (reviewed in Magrath and Komdeur, 2003). Thus, it is reasonable to expect that low quality (LQ) males invest more in mate guarding than high quality (HQ) males, and also that females mated with HQ males are less likely to search for EPCs. The level of EPP in a population reflects the response to a series of possible conflicts between females, their social mates, and one or more extra-pair males (Lifjeld *et al*., 1994).

3. Qualitative models

The model presented in this study was implemented in the qualitative simulator Garp3 (Bredeweg *et al.*, 2006), using elements of ontology provided by the qualitative process theory (Forbus, 1984). Following the compositional modelling approach (Falkenhainer and Forbus, 1991), the model was developed by creating a library of reusable components, model fragments, used to represent processes, agents and static views of the modeled system. Causal relations are captured by means of relations between the quantities, direct *influences* (*I*+ and *I*-) for representing processes and *qualitative proportionalities* (*P*+ and *P*-) for propagating the effects of processes to other parts of the system. *Static* model fragments are used to model events that do not change with time, while *process* and *agent* model fragments implement system components that define dynamic aspects of the system. *Scenarios* provide the context of the initial values from where the simulations progress. Once quantity values and relations are defined, the simulator generates *states*. Transitions between states are determined by transition rules and options defined by the user, and each sequence of states created during the simulation is a *behaviour path*. The collection of all the states produced by the simulation is denominated *state graph*. With Garp3 functionalities, it is possible to inspect the causal model, the equation history and the quantity value history. This approach has been used in different ecological studies (for example, Salles *et al*., 2003; Salles and Bredeweg, 2006; Salles *et al*., 2006), but not in behavioural ecology.

4. The model

Relevant assumptions in this model are: (a) males cannot maximize both within-pair and extra-pair paternity simultaneously. Accordingly, mate guarding can only be totally efficient if it occurs continuously throughout the entire period of female fertility; (b) males can be of either high (HQ) or low quality (LQ) and female propensity to search for EPC is related to the quality of their mates; (c) mate guarding varies as a function of male quality; (d) females do not obtain other non-genetic benefits from an extra-pair mating.

The current version of the model consists of three entities associated to 16 quantities, that express properties of the entities and ultimately characterize the states of the system under study. The system structure is organized around entities and the configurations 'Female' *mates with* 'Male' and these two entities are *included* in the 'Population'. Quantities associated to the entity 'Population' represent aspects of population dynamics. *Growth rate* represents the population growth process, and population *Density* is a proxy for the concept of breeding density. The entity 'Male', is associated to the quantities *Mate guarding* and *Search for EPC*. These quantities have quantity spaces spanning from zero to maximum, to capture the idea that a male has a limited amount of time to expend in each of these behaviours. The quantity *Control rate*, which is motivated by mate guarding, represents the level of control males have regarding female behaviour. The quantity *N of within partner chicks* represents offspring produced with the male's pair and its quantity space includes a maximum point because the number of chicks produced is constrained by the number of eggs a female can lay. Females are also associated to the quantity *N of within partner chicks* with the same quantity space. The quantities

related to the number of extra-pair young produced by males (*N of male extra-pair young*) and females (*N of female extra-pair young*) have different quantity space for a simple reason; males can fertilize a high number of females, while females are again constrained by the number of eggs they produce. Therefore, only the female's quantity space includes the value maximum. The *Fertilization rate* refers to the rate of female fertilization. The quantity *N of female EPC solicitations* represents female extra-pair behaviour – solicitations for extra-pair copulation, that may result in fertilization. Male and female *fitness* are calculated in different ways (see specific fragments below), in relation to number of chicks produced (for males) and the quality of chicks produced (for females). Female fitness changes are seen as the result of a process, with a *Fitness variation rate* that varies according to the levels of within pair and extra-pair behaviour.

The library consists of 33 model fragments, 13 of them representing ecological processes (population growth, female reproduction, constraining female behaviour, and female fitness variation). Reproduction of females is represented by a model fragment in which there are two opposite direct influences: I+(*N of within partner chicks*, *Fertilization rate*) and I–(*N of female extra-pair young*, *Fertilization rate*). An inverse correspondence between the quantity spaces of the two types of offspring assures that increase in one of them corresponds to decrease in the other, given the limited number of eggs produced by the female in her lifespan. This representation expresses the idea that females exert some control over the offspring produced.

Mate guarding is a central concept for the control of EPP. It is assumed that, given that the male's time and energy budgets are limited, there is a tradeoff between mate guarding and search for EPC, so that low quality males invest more in the former and less in the latter, while high quality males invest more in the search for EPC. This hypothesis is captured by model fragments that represent male fitness variation. In this model, the quantity *Mate guarding* is determined by breeding *Density* and by the quantity *N of female EPC solicitations*. The male reaction is captured by the model fragment 'Constraint on female behaviour', a process that constrains females from engaging in EPC.

The most important model fragments are those related to the fitness variation of females and males. Female fitness changes due to a process, represented in the model fragment 'Female fitness variation', which introduces the relation I+(*Female fitness*, *Fitness variation rate*). The rate of this process is influenced both by *N of within partner chicks* and by *N of female extra-pair young*. As mentioned above, females mated with low quality males may increase their fitness via EPC. This idea is captured by the model fragment 'Fitness of female mated with low quality male' (Figure 1) that shows a negative proportionality between *N of within partner chicks* and *Fitness variation rate*, and a positive one linking *N of female extra-pair young* and the rate. This model fragment also shows the assumption '*Female fitness variation rate follows the n of extra-pair young*' related to the correspondence between derivatives, causing the rate to follow the direction of change of female extra-pair offspring.

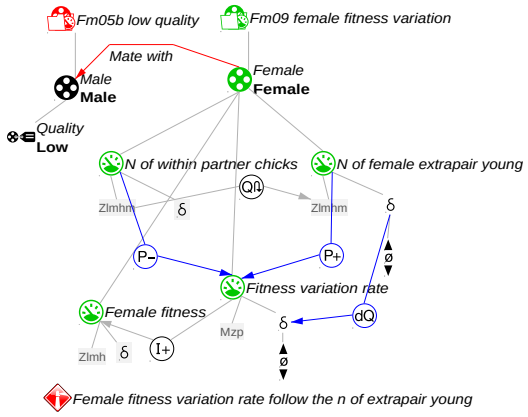


Figure 1. Model fragment 'Fitness of female mated with low quality male'

If the female is mated to a high quality male, her fitness variation process is organized in a similar way, except for a positive proportionality between *N of within partner chicks* and *Fitness variation rate*, that replaces the negative proportionality shown in Figure 1.

5. Simulations

The model has 31 scenarios, from simple simulations exploring only population dynamics and male or female features to complex interactions between HQ and LQ males and female within-pair and extra-pair reproduction and the effects of mating success and mate quality on their fitness. The initial scenario 'Female mated with a HQ male' produces the most complex simulation within this model. The following assumptions hold in this scenario: 'Female fitness variation rate follow the n of chicks with the partner' and 'N of within partner chicks are the same for males and females'. In this scenario, population *Growth rate* starts with the value <*plus*,?>, *Density* with <*medium*,?>, and the trade off between male *Mate guarding* and *Search for EPC* have initial values in the interval <*low*,?> and <*high*,?> respectively. All the other quantities started with the value <*medium*,?>, except the rates that have value <*zero*,?>. The simulation produced three initial states; the full simulation, 166 states. The causal model, as it appears in state 10, is presented in Figure 2. It shows that an increase in *Density* propagates to

Mate guarding, which has three influences: (a) it negatively affects the male *Search for EPC*, reducing the *N of male extra-pair young*; (b) it positively affects *Constraining rate*, triggering the feedback loop that reduces the *N of female EPC solicitations* and causes *Mate guarding* to decrease; and (c) it positively affects *Fertilization rate*, causing *N of within partner chicks* to increase and *N of female extra-pair young* to decrease. These two quantities influence the female *Fitness variation rate*, but given that there is a derivative correspondence, the rate will take the derivative of *N of within partner chicks*.

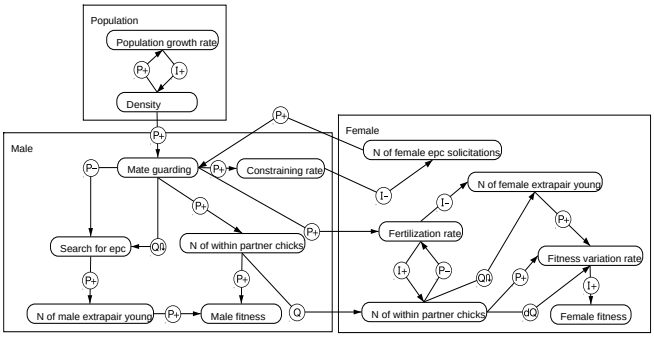


Figure 2. Causal model in state 10 of the simulation starting with the scenario 'Female mated with a high quality male'

Values of selected quantities in the behaviour path [2 → 6 → 13 → 14 → 58 → 64 → 143] are presented in Figure 3. These diagrams show that, in this behaviour path, while *Density* is increasing male *Mate guarding* is also increasing and *Search for EPC is decreasing* and *N of female EPC solicitations* is decreasing. In both partners, *N of within partner chicks* is increasing and *N of male extra-pair young* and *N of female extra-pair young* are decreasing, reaching zero in state 143.

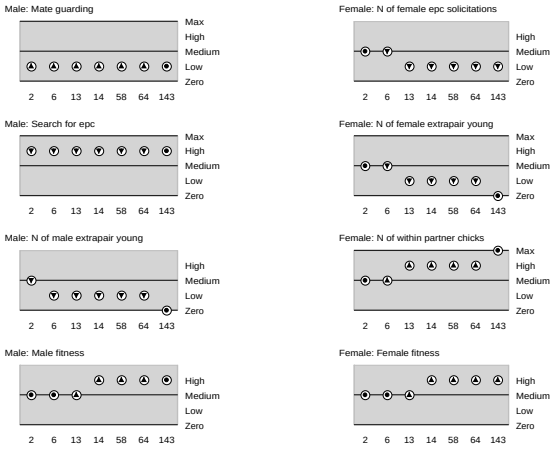


Figure 3. Value history diagrams of selected quantities in a simulation of the scenario 'Female mated with a HQ male'

In the behaviour path explored above, *Male fitness* followed the derivative of *N of within partner chicks*, but there are behaviour paths in which it follows the derivative of *N of male extra-pair young* and is decreasing. In fact, *Male fitness* may enter a cyclic path, and oscillate between the values *low* and *high*, as shown in Figure 4:

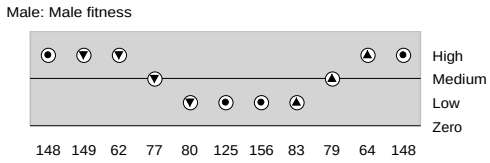


Figure 4. Cyclic behaviour expressed by the quantity *Male fitness* in a simulation starting with the scenario 'Female mated with a high quality male'

6. Discussion

Qualitative reasoning models may contribute to the development of ecological theories in many ways. They can be used to formalize scientific hypotheses, or qualitative theories about particular domains, that may support the development of new ideas that explore applications or further theoretical developments of those theories. The model described in this paper also has the potential to support further studies on sexual selection. As genetic polyandry with extra-pair offspring seems to happen in approximately 90% of bird species (Griffith *et al*., 2002) it is of great interest to understand why such sexual behaviour evolved and how these traits may affect bird fitness. However, modelling sexual behaviour in birds presents new challenges for qualitative reasoning models. Among them, the need for representing interactions involving phenomena that occur in different time scales, and in different generations. Integration of changes in successive generations in phylogenetic studies will require the implementation of evolutionary mechanisms such as mutation, adaptation and natural selection that may explain why current generations of birds behave as they do.

The model presented here was designed to help students and researchers interested in behavioural ecology, especially in sexual selection, to understand processes involved in evolution of extra-pair paternity on a narrow scale, based on individual decisions. Although not evaluated so far by end users, the model was evaluated by an expert in animal behaviour in a step-by-step procedure, during which relevant concepts, model fragments, scenarios and simulation results were presented and discussed (Rykiel, 1996). The expert made comments and suggestions to improve knowledge representation, which were included in the model, and the results obtained were considered satisfactory and potentially useful.

7. Conclusions

In this paper, qualitative reasoning techniques are used to formalize representations of hypotheses related to sexual behaviour and extra-pair paternity in avian species, expressing cause – effect relations and exploring complex chains of reasoning about system behaviour. The answers provided by the model to the questions formulated in the introduction are based on the most relevant mechanisms identified in the literature to explain extra-pair paternity in birds, such as the trade-off between mate guarding and male search for EPC and the use of different strategies for increasing fitness in females mated to high and low quality males.

Ongoing work includes improving representations of both male and female behaviour and of the factors that affect their fitness. For males, current modelling effort aims to improve the representation of mate guarding and paternal care; for females, the goal is to explore alternative hypotheses about how their own interests would drive behaviour and influence the level of extra-pair paternity.

The results obtained so far confirm the potential of QR modelling contribution to the ecologists' understanding of the theoretical basis of complex aspects of sexual selection in birds.

Acknowledgements We are grateful to Regina Macedo for comments and suggestions on the model and on this version of the manuscript.

References

Birkhead, T. R. and Møller, A. P. 1992. *Sperm Competition in Birds- Evolutionary Causes and Consequences*. London, Academic.

Bredeweg, B., Bouwer, A., Jellema, J., Bertels, D., Linnebank, F. and Liem, J. 2006. Garp3 - A new Workbench for Qualitative Reasoning and Modelling. *Proceedings of the 20th International Workshop on Qualitative Reasoning (QR06)*, 21-28.

Falkenhainer, B., and Forbus, K. 1991. Compositional modeling-finding the right model for the job. *Artificial Intelligence*, 51: 95-143.

Forbus, K. D., Qualitative process theory. *Artificial Intelligence*, 24: 85-168, 1984.

Griffith, S.C.; Owens, I.P.F. and Thuman, K.A. 2002. Extra pair paternity in birds: a review of interspecific variation and adaptive function. *Molecular Ecology*, 11: 2195-2212.

Hasselquist, D. and Bensch, S. 1991. Trade-off between mate guarding and mate attraction in the polygynous great reed warbler. *Behavioural Ecology and Sociobiology*, 28: 187-193.

Jennions, M. and Petrie, M. 2000. Why do females mate multiply? A review of the genetic benefits. *Biological Reviews of the Cambridge Philosophical Society*, 75: 21-64.

Lifjeld, J. T., Dunn, P. O.,Westneat, D.F. 1994. Sexual selection through sperm competition in birds- male-male competition or female choice? *Journal of Avian Biology*, 25: 244-50.

Magrath, M. J. L., Komdeur, J. 2003. Is male care compromised by additional mating opportunity? *Trends in Ecology and Evolution*, 18: 424-430.

Rykiel, E.J. 1989. Artificial Intelligence and Expert Systems in Ecology and Natural Resource Management. *Ecological Modelling*, 46: 3-8.

Rykiel, E.J. 1996. Testing ecological models: the meaning of validation. *Ecological Modelling*, 90:229-244.

Salles, P.; Bredeweg, B.; Araújo, S. e Neto, W. 2003. Qualitative models of interactions between two populations. *AI Communications* 16(4): 291-308.

Salles, P. and Bredeweg, B. 2006. Modelling population and community dynamics with qualitative reasoning. *Ecological Modelling*, 195: 114-128

Salles, P., Bredeweg, B., Bensusan, N. 2006. The ants' garden: qualitative models of complex interactions between populations. Ecological Modelling, 194: 90-101.

Trivers R.L. 1972 Parental investment and sexual selection. *In* B. Campbell (ed.) *Sexual Selection and the Descent of Man* , pp. 136-179. Aldine Press, Chicago.

Weld, D. and de Kleer, J. (eds.) 1990. *Readings in Qualitative Reasoning about Physical Systems*. San Mateo, CA: Morgan Kaufmann.

Westneat, D. F. and Stewart, I. R. 2003. Extra-pair paternity in birds: causes, correlates, and conflict. *Annual Review of Ecology and Systematics*, 34: 365-396.

Qualitative Models of Shape, Size, Orientation and Distance Applied to the Description of Images Containing 2D Objects

Zoe Falomir and M. Teresa Escrig

Universitat Jaume I, Engineering and Computer Science Department
E-12071 Castellón, Spain
{zfalomir, escrigm}@icc.uji.es

Abstract

In this paper, we present an approach to qualitative description of images containing non-overlapping 2D objects. This approach consists of extracting the properties of the main points of the objects in an image and applying qualitative models to them. Qualitative models of shape and size describe visual features of the objects in the image, while qualitative models of orientation and distance describe spatial features of those objects. Finally, a string which describes the image in qualitative terms is obtained. This string can be used to compute the relationships between the objects in the image and also to compare images by obtaining a degree of similarity between them.

Introduction

Service robots need a system of visual perception similar to that of human beings in order to interact with people and to navigate efficiently through real environments dealing with problems such as unpredictable obstacles or changing targets. Qualitative techniques for representing information and reasoning can help robots to interpret their environment: to describe a new place and to recognize a known one only by using the main landmarks/features of that environment.

Human beings use language to describe images. How we do it is studied by psycho-linguistic researchers (Landau & Jackendoff 1993). In general, nouns are used to refer to objects, adjectives to express properties of these objects and prepositions to express relations between them. These nouns, adjectives and prepositions are qualitative labels which extract knowledge from images and which can be used to communicate and compare image content.

Because of the numerical properties of digital images, most of the image treatment in computer vision has been carried out by applying mathematical models and other quantitative techniques to detect objects in an image. Our aim is to use some of these techniques in order to obtain the main points of the objects contained inside an image (such as edges and vertices) and then to apply qualitative models to them, so that we could obtain the main visual and spatial information of these objects.

The approach presented in this paper applies qualitative models of shape (Museros & Escrig 2004), size, distance (Escrig & Toledo 2001) and orientation (Hernández 1991; Freksa 1992) to image description. These models represent visual (shape and size) and spatial (orientation and distance) information about 2D objects in an image. The visual information obtained from an image describes *which* objects are placed in an environment, and the spatial information obtained describes *where* all the objects are located with respect to (wrt) each other and wrt the observer. Contextualizing our approach to robot navigation, a mobile robot with a camera could qualitatively describe its environment by describing the images taken by the camera and, by interconnecting all the images, it could localize itself wrt all the objects contained in any of the images, that is, in any place of the environment.

In the literature, related works which describe images qualitatively have appeared. In (Museros & Escrig 2004), an approach for the qualitative description of the shape of a 2D object contained in an image is applied to assemble tile mosaics. In (Qayyum & Cohn 2007; Bañuelos 2000), qualitative description of images are used for image retrieval in data bases. In (Qayyum & Cohn 2007), landscape images are divided by a grid for its description so that semantic categories (grass, water, etc.) are identified and qualitative relations of compared size, time and topology relations are used to describe the image. In (Bañuelos 2000), images composed by squares, triangles and circles are described qualitatively by using the approach in (Chang *et al.* 1989) which uses projections to find the spatial relations between the objects in an image. In (Socher 1997), a verbal description of an image is provided to a robotic manipulator system, so that it can identify and pick an object. These objects are described qualitatively by predefined categories of type, colour, size, shape and spatial relations. In (Lovett *et al.* 2006), a qualitative description for sketch image recognition is proposed, which defines lines, arcs and ellipses as basic elements and also considers relative position, relative length and relative orientation of pairs of edges.

We believe that all the works described above provide evidence for the effectiveness of using qualitative information to describe and compare images. However, to our knowledge, none of them is intended to be applied to the description of the robot environment and to the enhancement of robot navigation. This is our main aim. Although we have tackled the problem by considering images containing 2D objects, in the near future, we will describe real images taken from the robot environment where doors, corners and other 3D objects will appear.

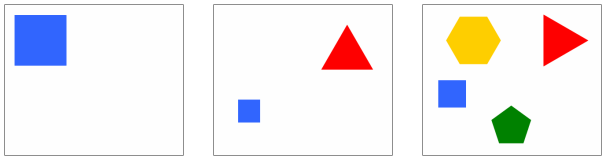
Outlining our Approach for Qualitative Image Description

In order to describe qualitatively images composed of 2D objects, our approach applies qualitative models of shape, size, orientation and distance.

If the considered image contains just one object, only its shape, its size with respect to (wrt) the image, and its orientation wrt the centre of the image can be described. For example, a possible qualitative description of the image in Figure 1a could be "an image containing a medium-sized blue square situated left-front wrt the centre of the image".

If the considered image contains two objects, the relative size of each object wrt each other and the orientation of an object wrt the other objects can be described. For example, a possible qualitative description of the image in Figure 1b could be "an image containing a blue small square and a red medium-sized triangle (…) the triangle is situated right-front wrt the centre of the image and right-front wrt the square".

Finally, if the considered image contains more than two objects, relations of compared distance and relative orientation between objects can be described. For example, a possible qualitative description of the image in Figure 1c could be "an image containing a triangle, an hexagon, a square and a pentagon (…) the square is closer to the pentagon than to the triangle (…) the pentagon is located left-middle wrt the reference system defined from the centroid of the triangle to the centroid of the hexagon, while the square is located left-front wrt the same reference system (…)".



(a) (b) (c)

Figure 1. Images composed by 2D objects.

Table 1 shows the qualitative models that our approach uses to describe images which contain an object, two objects and more than two objects.

Table 1. Qualitative models our approach uses to describe images composed by 2D objects.

Qualitative Model of	Number of Objects 1	2	> 2
Shape	x	x	x
Fixed Orientation	x	x	x
Compared Size (Object wrt Image)	x	x	x
Compared Size (Object wrt Object)	-	x	x
Relative Orientation	-	-	x
Compared Distance	-	-	x

The following sections describe the qualitative models of shape, size, orientation and distance used by our approach. Then, the structure of the string for the qualitative description of any image is presented. Finally, our conclusions and future work are explained.

Qualitative Model for Shape Description

Our approach uses Museros and Escrig's qualitative model for shape description (Museros and Escrig 2004) to obtain a description of any 2D object in an image. This model has been extended to identify regularity and convexity of the objects. Next section summarizes Museros and Escrig's model and, in the following section, our extension to this model is explained.

Museros and Escrig's Approach

Museros and Escrig's approach extracts the boundary of each object in an image by applying Canny's edge detector (Canny 1986). Then the slope between the pixels that compose that boundary is compared in order to obtain the main points of each object: vertices and points of curvature. These points are described qualitatively by this approach, which also obtains the colour and the centroid of each object.

The **vertices** of each object are described by a set of three elements $<A_j, L_j, C_j>$ where:

$$A_j \in \{right, acute, obtuse\},$$
$$L_j \in \{smaller, equal, bigger\} \text{ and}$$
$$C_j \in \{convex, concave\}$$

- A_j or the qualitative amplitude of the angle j is calculated by obtaining the circumference that includes the previous and following vertices of vertex j (j-1 and j+1) (Figure 2). If vertex j is included in that circumference, then the angle is *right*. If vertex j is external to the circumference, then the angle is *acute*. Finally, if vertex j is included in the circle that this circumference defines, then the angle is *obtuse*.
- L_j or the relative length of the two edges related to vertex j is obtained by comparing the Euclidean distance between two segments: the segment defined from vertex j-1 to vertex j and the segment defined from vertex j to vertex j+1. If the first distance obtained is smaller/equal/bigger than the second one, the relative length between the two edges in vertex j is *smaller*/*equal*/*bigger*, respectively.

- C_j or the convexity of the angle defined by the edges related to vertex j is calculated by obtaining the segment from the previous vertex (j-1) to the following vertex (j+1). If vertex j is on the left of that segment, then the angle is *convex*. If vertex j is on the right of that oriented segment, then the angle is *concave*.

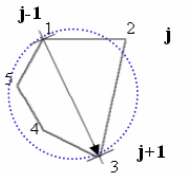


Figure 2. Characterization of a vertex in Museros and Escrig's approach.

The **points of curvature** of each object are characterized by a set of three elements <*curve*, TC_j, C_j> where:

$TC_j \in$ {*acute, semicircular, plane*} and

$C_j \in$ {*convex, concave*}

- TC_j or the type of curvature in point j is obtained by comparing the size of the segment defined by the previous point of curvature j-1 and the centre of the curve (*da* in Figure 3) with the size of the segment defined by the point of curvature j and the centre of the curve (*db* in Figure 3). If *da* is smaller than *db*, the type of curvature in j is *acute*; if *da* is equal to *db*, the type of curvature in j is *semicircular*; and finally, if *da* is bigger than *db*, the type of curvature in j is *plane*.

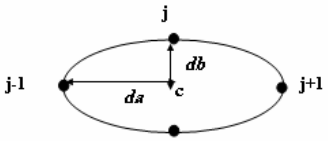


Figure 3. Characterization of a curve in Museros and Escrig's approach.

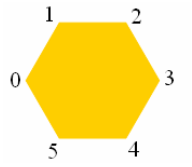
- C_j or the convexity of the point of curvature j is calculated by obtaining the segment from the previous vertex (j-1) to the following vertex (j+1). If vertex j is on the left of that segment, then the angle is *convex*. If vertex j is on the right of that segment, then the angle is *concave*.

Thus, the complete description of a 2D object is defined as a set of qualitative tags as:

[Type, Colour, [A_1,C_1,L_1]|[*curve*,TC_1,C_1], …, [A_n,C_n,L_n]|[*curve*,TC_n,C_n]]

where *n* is the number of vertices and points of curvature of the object, *Type* belongs to the set {*without-curves, with-curves*}, *Colour* describes the RGB colour of the object by a triple [*R,G,B*] which stands for the Red, Green and Blue coordinates and $A_1,\ldots,A_n$, $C_1,\ldots,C_n$, $L_1,\ldots,L_n$ and $TC_1,\ldots, TC_n$, which have been previously explained.

Finally, as an example, Figure 4 shows the qualitative description of the hexagon in Figure 1c provided by Museros and Escrig's approach. The first vertex detected is that with the smaller coordinate x (considering that the origin of coordinates in computer vision is the upper-left corner of the image), while the first vertex described is that defined by the three first vertices obtained.



QualitativeShapeDescription(S)=
[*without-curves*, [255, 206, 0],
 [
1: [*obtuse*, *equal*, *convex*],
2: [*obtuse*, *equal*, *convex*],
3: [*obtuse*, *equal*, *convex*],
4: [*obtuse*, *equal*, *convex*],
5: [*obtuse*, *equal*, *convex*],
0: [*obtuse*, *equal*, *convex*],
 ],
].

Figure 4. Qualitative description of the hexagon in Figure 1c.

Characterizing the Shape of 2D Objects

In Museros and Escrig's approach, a qualitative tag is included in order to distinguish if the object has curves or not (*with-curves*, *without-curves*), so that the comparison process can be accelerated. However, a more accurate characterization of the objects (according to geometry principles) can be defined by using the qualitative features described for each vertex.

The characterization defined for our approach consists on: (1) giving a name to the object that could represent it geometrically, (2) describing the regularity of its edges and (3) defining the convexity of the whole object.

Therefore, **objects without curves** can be characterized by a set of three elements:

[Name, Regularity, Convexity]

where,

Name ∈ {*triangle, quadrilateral, pentagon, hexagon, heptagon, octagon, …, polygon*},

Regularity ∈ {*regular, irregular*} and

Convexity ∈ {*convex, concave*}

- **Name** is the name given to the object depending on its number of edges (or vertices qualitatively described) and it can take values from *triangle* to *polygon*;
- **Regularity** indicates if the object have *equal* angles and *equal* edges (so it is *regular*), or not (so it is *irregular*);
- **Convexity** indicates if the object has a *concave* angle (so it is *concave*) or not (so it is *convex*).

However, for triangular and quadrilateral objects a more accurate characterization can be made.

Triangular objects can be characterized as *right*, *obtuse* or *acute triangles* according to the kind of angles they have, and as *equilateral*, *isosceles* or *scalene triangles* according to the relation of length between its edges. Therefore, the element **Name** for a triangle is made up by three elements:

triangle–Kind_of_angles–Sides_relation

where,

Kind_of_angles ∈ {*right, obtuse, acute*}
Edges_Relation ∈ {*equilateral, isosceles, scalene*}

- **Kind_of_angles** indicates if the triangle has got a right angle (so it is *right*), an obtuse angle (so it is *obtuse*), or if all its angles are acute (so it is *acute*); and
- **Edges_relation** shows, if the edges of the triangle are all equal (so it is *equilateral*), or two equal (so it is *isosceles*), or none equal (so it is *scalene*).

Quadrilateral objects can be also characterized more accurately as *square, rectangle* or *rhombus* depending on the compared length between its edges and on its kind of angles. Therefore, the element **Name** for a quadrilateral is made up by two elements:

quadrilateral–Type_quadrilateral

where,

Type_quadrilateral ∈ {*square, rectangle, rhombus*}

- **Type_of_quadrilateral** specifies if the quadrilateral is a *square* (if all their angles are right and their edges equal), a *rectangle* (if all their angles are right and their opposite edges are equal), or a *rhombus* (if all their edges are equal and their opposite angles are equal).

On the other side, **objects with curves** can be also characterized by a set of three elements:

[Name, Regularity, Convexity]

where,

Name ∈ {*circle*, *ellipse*, *polycurve*, *mix-shape*}
Regularity ∈ {*regular, irregular*}
Convexity ∈ {*convex, concave*}

- **Name** is the name given to the object depending on its properties: *mix-shape* (if the shape of the object is made up by curves and straight edges), *polycurve* (if the shape of the object is made up only by curves), *circle* (if the shape of the object is a polycurve with only four relevant points, two of them defined as semicircular points of curvature) and *ellipse* (if the shape of the object is a polycurve with only four relevant points, two of them defined as points of curvature with the same type of curvature but different from *semicircular*, that is, both *plane* or *acute*).
- **Regularity** regarding to curves is not defined by our approach from the point of view of geometry. We consider 2D objects with circular or elliptical shapes as *regular* and the rest of objects with curvaceous shapes as *irregular*.
- **Convexity** of objects with curvaceous shapes is defined in the same way as for objects containing only straight edges: if an object has a *concave* vertex or point of curvature, that object is defined as *concave*; otherwise it is defined as *convex*.

Finally, according to this characterization, the triangle in Figure 1b will be characterized as *[triangle-acute-equilateral, regular, convex]*.

Qualitative Model of Compared Size

In order to describe images which contain objects with the same features but with different size, a qualitative model of Compared Size (CS) has been developed.

Represented Information. In order to represent the size of the objects in the image a Compared Size Reference System (CSRS) with two levels of granularity has been defined. The reference system with coarse level of granularity represents the size of an object A wrt the size of the image (A wrt Image), while the reference system with fine level of granularity represents the size of an object A wrt the size of another object B (A wrt B). Two reference systems are used in order to distinguish situations where the object compared can be larger than the other object (object wrt object comparison) or not (object wrt image comparison).

The CSRS has three components, CSRS = {US, LAB_{csrs}, INT_{csrs}}, where US refers to the relation obtained after comparing the area of an object wrt the area of another object or wrt the area of the image; LAB_{csrs} refers to the set of qualitative labels which represent compared size; and INT_{csrs} refers to the intervals associated to each compared size label of LAB_{csrs}, which will describe the size of the object in terms of US and which depends on the application.

The CSRS with coarse level of granularity is used to obtain the size of each object wrt the size of the image and it is defined as:

$CSRS_{LAB1}$ = {*small (s)*, *medium (m)*, *large (l)*}

$CSRS_{INT1}$ = {]0, 1/8 us[, [1/8 us, 1/4 us[, [1/4 us, 1us[}.

The CSRS with fine level of granularity is used to obtain the size of each object (A) wrt the size of another object (B) and it is defined as:

$CSRS_{LAB2}$ = {*smaller_than_half (sh)*, *half (h)*, *larger_than_half (lh)*, *equal (e)*, *smaller_than_double (sd)*, *double (d)*, *larger_than_double (ld)*}

$CSRS_{INT2}$ = {]0, 1/2 us[, [1/2 us, 1/2 us],]1/2 us, 1 us[, [1 us, 1 us],]1 us, 2 us[, [2 us, 2 us],]2 us, ∝[}.

Application to qualitative description of images. First, our approach for qualitative description of images calculates: (1) the area of the image (n pixels of height x m pixels of width) and (2) the area of each object by using the cross-product of its vertices, as described in (Goldman 1991). After that, our approach compares the area of each object wrt the area of the image and obtains a qualitative tag of compared size from the reference system of compared size at a coarse level of granularity. Finally, our approach compares the area of each object wrt the area of

the other objects in the image and obtains a qualitative tag of compared size from the reference system of compared size at a fine level of granularity.

As an example, the square in Figure 5b will be characterized as: *[2, quadrilateral-square,..., small ..., [0, larger_than_half],[1,smaller_than_half],[3,larger_than_half]],...],* which means "a small square (Object 2) which is larger than half the triangle (Object 0), smaller than half the hexagon (Object 1) and larger than half the pentagon (Object 3)".

Qualitative Models of Orientation

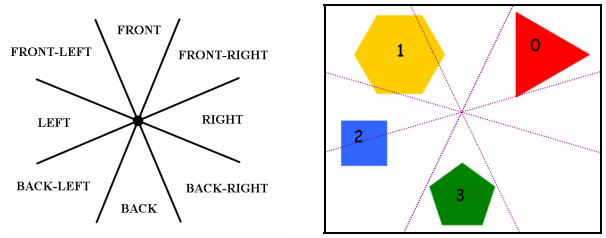
In order to provide a description of the location of the objects wrt the centre of the image, our approach uses Hernandez's qualitative model of Fixed Orientation (FO) (Hernández 1991) and fixes the area named as *front* of the defined reference system to the upper edge of the image.

Moreover, to obtain relative orientations between the objects of an image, our approach applies Freksa's model of Relative Orientation (RO) (Freksa 1992).

Qualitative model of fixed orientation

Represented Information. In order to represent the orientation of an object A wrt the image (A wrt Image) or to represent the orientation of an object A wrt the orientation of another object B (A wrt B), Hernandez's Fixed Orientation Reference System (FORS) is used. This FORS divides space into eight regions (Figure 5a):

FORS $_{LAB}$ = {*front (f)*, *back (b)*, *left (l)*, *right (r)*, *left-front (lf)*, *right-front (rf)*, *left-back (lb)*, *right-back (rb)*}



(a) (b)

Figure 5. (a) Representation of the FORS; (b) FORS applied to image in Figure 1(c).

Application to qualitative description of images. First, our approach for qualitative description of images situates the FORS in the centre of the image in order to obtain a global orientation of all the objects inside that image. Secondly, our approach locates the centre of the FORS on the centroid of each object in order to obtain the location of all the other objects in the image wrt the current one.

The orientation of an object is determined by the union of all the orientation labels obtained for each of the vertices/points of curvature of the object.

Our approach calculates the orientation of a vertex wrt a FORS by obtaining the slope between this vertex and the centre of the FORS and then comparing this slope with the slope of each of the straight lines which define the regions of orientation in the FORS (Figure 5a).

Finally, as an example, the orientation description for the hexagon (Object 1) in Figure 5b is the following string: *[1, hexagon, ..., [front, front_left, left], [[0, left], [2, front, front_right], [3, front, front_left]], ...]* which means "an hexagon (Object 1) located: front/front-left/left wrt the centre of the image, left wrt the triangle (Object 0), front/front-right wrt the square (Object 2) and front/front-left wrt the pentagon (Object 3).

Qualitative model of relative orientation

Represented information. Freksa's model divides the space into 15 qualitative regions by means of a Reference System (RS). This RS is formed by an oriented line determined by two reference points *a* and *b*. From *a* to *b* a line is defined, which determines the left/right dichotomy; another line is defined perpendicular to *b*, which defines the first front/back dichotomy; and another line is defined perpendicular to *a* to establish the second front/back dichotomy. The information which can be represented by this model is the qualitative orientation of a point *c* wrt the RS formed by the points *a* and *b*, that is, *c wrt ab* (Figure 6). This model defines the following regions of orientation:

RORS $_{LAB}$ = { *left-front (lf)*, *straight-front (sf)*, *right-front (rf)*, *left (l)*, *identical-front (idf)*, *right (r)*, *left-middle (lm)*, *straight-middle (sm)*, *right-middle (rm)*, *identical-back-left (ibl)*, *identical-back (ib)*, *identical-back-right (ibr)*, *back-left (bf)*, *straight-back (sb)*, *back-right (br)* }

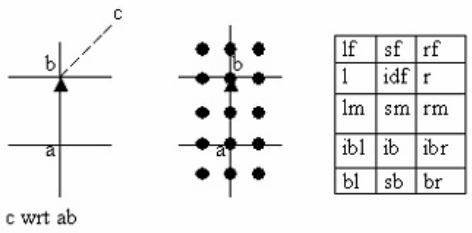


Figure 6. Freksa's model and its iconical representation: RS(a,b) and the 15 qualitative tags of orientation located according to RS(a,b).

Application to qualitative description of images. Our approach establishes a reference system (RS) between all pairs of objects in the image. The points *a* and *b* of the RS are the centroids of the selected objects. All the vertices and points of curvature of the objects that do not compose the RS are located with respect to the corresponding RS. The orientation of an object wrt each RS is the union of all the orientation labels obtained by each vertex or point of curvature of the object.

Our approach calculates the orientation of a vertex wrt a RORS by projecting each vertex to two lines of the RS: first to the line defined by the points *a* and *b* and then to the line perpendicular to *a* or to the line perpendicular to *b*, depending on which are closer to the vertex. If any of the vertices is included in any of these lines, no projection is needed. After having the vertices projected into the RS, we study the orientation of the RS by observing the tendency of the coordinates x and y of the points *a* and *b,* that is if they increase, decrease or remain constant from *a* to *b*. The tendency of the coordinates inside the lines perpendicular to the points *a* and *b* is related to the orientation of the RS. Therefore, if we compare the tendency of the coordinates of a vertex with the tendency of each line of the RS, we could locate the vertex to the *left*/*right* of the line *a*-*b* and to the *front/back* of the line perpendicular to *a* or to the *front/back* of the line perpendicular to *b*, depending on the situation of the vertex. By combining these locations and taking into account if the original vertex was included into a line of the RS or was projected into it, the final orientation of the vertex wrt the RORS is obtained.

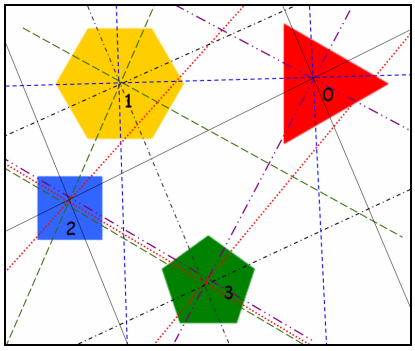


Figure 7. Reference systems obtained for the image in Figure 1c.

In Figure 7, all possible reference systems obtained for the image in Figure 1c are shown. Therefore, according to Figure 7, the orientation of the square (Object 2) is described as the following string:

[2, square, ..., [[[0, 1], left_front], [[0, 3], right_middle, right_front], [[1, 0], back_right], [[1, 3], right_middle], [[3, 0], left_middle, back_left], [[3, 1], left_middle]], ...]

which means that "the square is located: left-front wrt the RS from the triangle (Object 0) to the hexagon (Object 1), right-middle/right-front wrt the RS from the triangle (Object 0) to the pentagon (Object 3), back-right wrt the RS from the hexagon (Object 1) to the triangle (Object 0), right-middle wrt the RS from the hexagon (Object 1) to the pentagon (Object 3), left-middle/back-left wrt the RS from the pentagon (Object 3) to the triangle (Object 0) and left-middle wrt the RS from pentagon (Object 3) to the hexagon (Object 1)".

Qualitative Model of Compared Distance

In order to describe the distance between the objects in the image, our approach applies Escrig and Toledo's Compared Distance (CD) model (Escrig & Toledo 2001).

Represented Information. In order to compare the distance from the current object or point of view (PV) to another object (A) and the distance from that PV to another object (B), the Compared Distance Reference System (CDRS) defined by Escrig and Toledo is used. This CDRS has two components, CDRS = {RP, LAB}, where RP refers to the Reference Points between which the distances to be compared are calculated; and LAB refers to the labels which represent compared distances. The Referent Points (RP) in CDRS are: the point of view (PV), the first end point (A) to which the first distance is obtained and the second end point (B) to which the second distance is obtained. Compared distance labels are:

$$CDRS_{LAB} = \{closer\_than\ (ct),\ nearby\ (nb),\ further\_than\ (ft)\}$$

Application to qualitative description of images. In our approach, the Euclidean distance between the centroid of the current object and the centroid of the other objects in the image is obtained. Then, those distances are compared among them by using the CDRS. Our reference points in the CDRS are: the centroid of the current object or the PV, the centroid of another object (A) to which the first Euclidean distance is calculated, and the centroid of the other object (B) to which the second Euclidean distance is calculated.

As an example, the compared distance for the triangle in Figure 7 is described as the string: *[0, triangle-acute-equilateral, ..., [[1,2, closer_than], [2,1, further_than], [1,3, closer_than], [3,1, further_than], [2,3, further_than], [3,2, closer_than]],* which means that "the triangle (Object 0) is: closer to the hexagon (Object 1) than to the square (Object 2), and viceversa, further from the square than from the hexagon; closer to the hexagon (Object 1) than to the pentagon (Object 3), and viceversa, further from the pentagon than to the hexagon; and finally, further from the square (Object 2) than from the pentagon (Object 3), and vicerversa, closer to the pentagon than to the square".

Final String for the Qualitative Description of an Image

Finally, an application that provides the qualitative description of an image containing two-dimensional objects has been implemented.

In general, the structure of the string provided by the application, which describes any image composed by K two-dimensional objects, is defined as a set of qualitative tags as:

[[QVisual, QSpatial]$_1$, …, [QVisual, QSpatial]$_K$]

For each object in the image, its qualitative visual characteristics (QVisual), and its qualitative spatial characteristics inside the image (QSpatial) are described.

The qualitative visual characteristics of the objects (QVisual) consist of the identifier of the object, its name, its regularity, its convexity, its colour in RGB coordinates, the type of object (if it has curves or not), its size wrt the image, the qualitative description of its shape, and a list of compared sizes wrt the other objects:

QVisual = [ObjId, Name, Regularity, Convexity, Colour, Type, Size, QShapeDesc, ListSize$_{wrt}$Obj]

where,

ObjId = N ∈ [0, ∞]

Name ∈ {*triangle, quadrilateral,..., polygon, circle, ellipse, polycurve, mix-shape*}

Regularity ∈ {*regular*, *irregular*}

Convexity ∈ {*convex, concave*}

Colour = [R, G, B] / R, G, B = N ∈ [0, 255]

Type ∈ {*without-curves*, *with-curves*}

Size = {*small*, *medium*, *large*}

QShapeDesc = [[PointQDesc$_1$, …, PointQDesc$_{NmP}$]]

PointQDesc = [Angle|*curve*, Lenght|TypeCurvature, Convexity]

where,

Angle ∈ {*acute, right, obtuse*}

Length ∈ {*smaller, equal, bigger*}

TypeCurvature ∈ {*acute, semicircular, plane*}

Convexity ∈ {*concave*, *convex*}

ListSize$_{wrt}$Obj = [[ObjId, CompSize]$_1$,…, [ObjId, CompSize]$_{K-1}$],

where,

CompSize = {*smaller_than_half*, *half*, *larger_than_half*, *equal*, *smaller_than_double*, *double*, *larger_than_double*}

Finally, the qualitative spatial situation of each object inside the image (QSpatial) consists of (1) the fixed orientation of the current object wrt the centre of the image and a list of fixed orientations of the current object wrt the other objects; (2) a list of relative orientations of the current object wrt all the reference systems in the image; and (3) a list of qualitative compared distances obtained by comparing the distance from the current object to all pairs of other objects in the image.

QSpatial = [Orient$_{wrt}$Image, LOrien$_{wrt}$Obj, LOrien$_{wrt}$RS, LCompQDistance]

where,

Orient$_{wrt}$Image = {*front*, *back*, *left, right*, *front-left*, *front-right*, *back-left*, *back-right, centre*}

LOrien$_{wrt}$Obj =[[ObjId, LOrien]$_1$,…,[ObjId, LOrien]$_{K-1}$]

LOrien = [Orien$_1$, …, Orien$_{no}$]

Orien = {*front*, *back*, *left, right*, *front-left*, *front-right*, *back-left*, *back-right*}

LOrien$_{wrt}$RS = [Orien$_{wrt}$RS$_1$, … Orien$_{wrt}$RS$_m$]

Orien$_{wrt}$RS = [ObjId$_A$, ObjId$_B$, [RelOrien$_1$,..., RelOrien$_{nr}$]]

RelOrien = {*left_front, straight_front, right_front, left, identical_front, right, left_middle, straight_middle, right_middle, identical_back_left, identical_back, identical_back_right, left_back, straight_back, right_back*}

LCompQDistance = [CQDistance$_1$,…, CQDistance$_m$]

CQDistance = [ObjId$_A$, ObjId$_B$, Qdistance]

Qdistance = {*closer_than*, *nearby*, *further_than*}

As a result of the application which implements our approach, the qualitative description of the image shown by Figure 8 is presented. The string obtained describes the four objects contained in the image in the order presented in Figure 8b. Objects containing vertices with smaller coordinates x and y are described first, considering that, in traditional computer vision, the origin of coordinates (x = 0 and y = 0) is located on the upper-left corner of the image. Figure 8b also shows the location of the vertices detected by our approach.

Finally, the qualitative description obtained by our application for the image in Figure 8 is the following one:

[

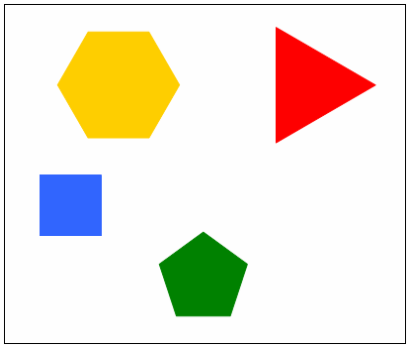
[0, triangle-acute-equilateral, regular, convex, [255,0,0], without_curves, small,
[[acute,equal,convex],[acute,equal,convex],[acute,equal,convex]],
[[1, larger_than_half],[2, smaller_than_double],[3, smaller_than_double]],
[front, right, front_right],
[[1, right], [2, front_right, right], [3, front, front_right]],
[[[1, 2], back_left], [[1, 3], left_middle], [[2, 1], right_front], [[2, 3], left_middle, left_front], [[3, 1], right_middle], [[3, 2], right_middle, back_right]],
[[1, 2, closer_than],[1, 3, closer_than],[2, 1, further_than],[2, 3, further_than],[3, 1, further_than],[3, 2, closer_than]]
],

[1, hexagon, regular, convex, [255, 206, 0], without_curves, small,
[[obtuse,equal,convex],[obtuse,equal,convex],[obtuse,equal,convex],[obtuse,equal,convex],[obtuse,equal,convex],[obtuse,equal,convex]],
[[0, smaller_than_double],[2, larger_than_double],[3, smaller_than_double]],
[front_left, front, left],
[[0, left], [2, front, front_right], [3, front, front_left]],
[[[0, 2], right_middle], [[0, 3], right_middle], [[2, 0], left_middle], [[2, 3], back_left, left_middle], [[3, 0], left_middle], [[3, 2], right_front, right_middle]],
[[0, 2, further_than],[0, 3, closer_than],[2, 0, closer_than],[2, 3, closer_than],[3, 0, further_than],[3, 2, further_than]]
],

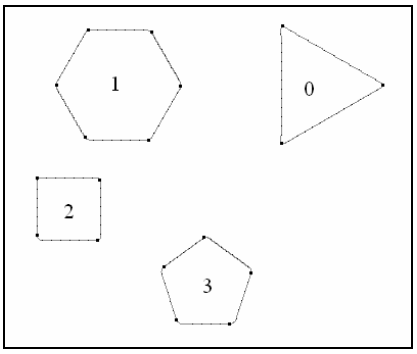
[2, quadrilateral-square, regular, convex, [49, 101, 255], without_curves, small,
[[right,equal,convex],[right,equal,convex],[right,equal,convex],[right,equal,convex]],
[[0, larger_than_half],[1, smaller_than_half],[3, larger_than_half]],
[left, back_left],
[[0, left, back_left], [1, back_left, back], [3, front_left, left]],

[[[0, 1], left_front], [[0, 3], right_middle, right_front], [[1, 0], back_right], [[1, 3], right_middle], [[3, 0], left_middle, back_left], [[3, 1], left_middle]],
[[0, 1, further_than],[0, 3, further_than],[1, 0, closer_than],[1, 3, closer_than],[3, 0, closer_than],[3, 1, further_than]]
],

[3, pentagon, irregular, convex, [0,129,0], without_curves, small,
[[obtuse,smaller,convex],[obtuse,equal,convex],[obtuse,bigger,convex], [obtuse,smaller,convex],[obtuse,equal,convex]],
[[0, larger_than_half],[1, larger_than_half],[2, smaller_than_double]],
[back],
[[0, back_left, back], [1, back_right, back], [2, right, back_right]],
[[[0, 1], left_middle], [[0, 2], left_middle], [[1, 0], right_middle], [[1, 2], left_middle, left_front], [[2, 0], right_middle], [[2, 1], right_middle, back_right]],
[[0, 1, further_than],[0, 2, further_than],[1, 0, closer_than],[1, 2, further_than],[2, 0, closer_than],[2, 1, closer_than]]
].



(a)



(b)

Figure 8. (a) Original image which has been processed by our application; (b) output image after the processes of segmentation and location of the relevant points of the objects. The numbers of the objects have been added previously to the image in order to arrange the qualitative description obtained.

As it can be observed from the qualitative description of the image in Figure 8, Museros and Escrig's method for detecting the vertices of the objects sometimes can be inaccurate, this is the reason why some regular objects are sometimes described as irregular by our approach.

Conclusion and Future Work

This paper has presented an approach for describing images containing 2D objects by applying qualitative models of shape, compared size, fixed and relative orientation and compared distance. Qualitative models of shape and size describe visual features of the objects in the image, while qualitative models of orientation and distance describe spatial features of those objects. As the final result, our approach obtains a string of qualitative labels which can be easily (1) used for computing relationships between the objects in the image and (2) compared to another string that describes another image in order to obtain a degree of similarity between both images.

While the obtained results are encouraging, much research remains in order to apply our approach to real images taken from a camera of a mobile robot. As for future work, we intend to (1) improve the accuracy of the detection process of the vertices of the objects in the image; (2) use the final string obtained by our approach in order to compare images and calculate a degree of visual and/or spatial similarity between them; (3) extend our approach to include topology relations between the objects in the image; and finally (4) apply our approach to qualitatively describe visual landmarks (such as corners or doors) for robot map building and navigation.

Acknowledgements

This work has been partially supported by CICYT and Generalitat Valenciana under grant numbers TIN 2006-14939 and BFPI06/219, respectively. We would like to thank the referees for their comments which helped improve this paper.

References

Bañuelos, J. I.: Algoritmo de indexado en bases de datos de imágenes. Master thesis in Maestría Electrónica supervised by Leopoldo Altamirano Robles. Instituto Nacional de Astrofísica, Óptica y Electrónica. México (2000).

Canny, J. F.; A computational approach to edge detection. IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-8, pp. 679--697 (1986).

Chang, S-K., Jungert, E., Li, Y.: Representation and retrieval of symbolic pictures using generalized 2D strings. In: SPIE Conference on Visual Communications and Image Processing, pp. 1360--1372. Philadelphia, USA (1989).

Escrig M. T., Toledo F.: Reasoning with Compared Distances at Different Levels of Granularity. In: 9th Conferencia de la Asociación Española para la Inteligencia Artificial (ECAI), Spain (2001).

Freksa, C.: Using Orientation Information for Qualitative Spatial Reasoning. In: Frank, A.U., Campari, I.,

Formentini, U. (eds.), Theories and Methods of Spatio-Temporal Reasoning in Geographic Space, Proc. Int. Conf. GIS- From Space to Territory. LNCS, vol. 639, pp. 162--178 (1992).

Goldman R. N., Area of Planar Polygons and Volume of Polyhedra. In Arvo J. Ed., Graphics Gems II, The Graphics Gems series, ISBN 0-12-064481-9, pp. 170 -- 171 (1991).

Hernández, D.: Relative Representation of Spatial Knowledge: The 2-D Case. In: Mark, D.M., Frank, A.U. (eds), Cognitive and Linguistic Aspects of Geographic Space, pp. 373--385, Kluwer, Academic Publishers, Dordrecht (1991).

Landau, B., Jackendoff, R.: 'What' and 'Where' in spatial language and spatial cognition. Behavioral and Brain Sciences, 16 (2), 217--265 (1993).

Lovett A., Dehghani M., Forbus K.: Efficient Learning of Qualitative Descriptions for Sketch Recognition. In: 20th International Workshop on Qualitative Reasoning. Hanover, USA, 2006.

Museros L., Escrig M. T.: A Qualitative Theory for Shape Representation and Matching for Design. In: 16th European Conference on Artificial Intelligence (ECAI), pp. 858--862. IOS Press. ISSN 0922-6389 (2004).

Qayyum, Z. U., Cohn, A. G.: Image retrieval through qualitative representations over semantic features. In: 18th British Machine Vision Conference (BMVC2007), pp. 610--619. Warwick, UK (2007).

Socher, G.: Qualitative Scene Descriptions from Images for Integrated Speech and Image Understanding. Dissertationen zur Kunstlichen Intelligenz (DISKI 170). Infix-Verlag, Sankt Augustin (1997).

Learning Qualitative Causal Models via Generalization & Quantity Analysis

Scott E. Friedman and Kenneth D. Forbus

Qualitative Reasoning Group, Northwestern University
2145 Sheridan Road, Evanston, IL 60208-0834 USA
{friedman, forbus}@northwestern.edu

Abstract

Learning causal models is a central problem of qualitative reasoning. We describe a simulation of learning causal models from exemplars that uses progressive alignment and qualitative process theory to derive plausible qualitative causal models from observations. We show how *protohistories* can be created via progressive alignment and used to infer causality. The result, a *causal corpus*, can make simple predictions and set the stage for more sophisticated qualitative models. The simulation has been successfully tested with learning causal mechanisms of three physical scenarios, with encouraging results.

Introduction

Forbus & Gentner (1986) proposed decomposing learning of physical domains from experience into four stages. (1) *Protohistories* are prototypical behaviors, generalized from multiple experiences. (2) The *causal corpus* consists of fragmentary causal models, created from protohistories. (3) These fragmentary models are organized into a *naïve physics*, which regularizes the fragmentary causal models by postulating broadly applicable mechanisms. (4) *Expert understanding* consists of deepening the naïve physics and tying it to mathematical and other formal models, typically culturally learned. Importantly, these stages are localized within the understanding of particular phenomena. For example, someone might have an expert understanding of electronics while having only a partial set of causal models for fluids.

This paper focuses on learning initial causal models of a domain from observations. We use qualitative process theory (Forbus, 1984) to formally represent causal models. Causal models are learned from symbolic representations of experiences via a combination of analogical processing (Gentner, 1983) and statistical methods. The simulation has been successfully tested on three scenarios; we use understanding floating versus sinking as a running example for illustration. We first review QP theory and the structure-mapping models we use. Then we discuss how protohistories are learned from experience via progressive alignment, proposing *generalization contexts* as a means of organizing experience around salient questions. Next we discuss *quantity analysis* strategies to develop fragmentary causal models by hypothesizing ordinal conditions, limit points, and new quantities. We summarize results from our simulation and close by discussing other related work and future plans.

Background

Our theoretical framework uses *qualitative process theory* as its account of modeling mechanisms of change. Changes are caused by continuous physical processes, which provide the notion of mechanism for causality (cf. Chi *et al* 1994; Ahn *et al* 1995). These changes propagate through the system via *qualitative proportionalities* which indicate causal relationships between quantities. Qualitative proportionalities provide only partial information about what will happen. This makes them particularly appropriate for representing local causal models, since models learned from one set of experiences can be more easily combined with others.

These causal laws are contextualized by belonging to either processes or *views*, and hold only when their *conditions* are true. Conditions are typically ordinal relations, involving parameters of the entities participating in the process or view. The values that a quantity is compared with in such relations are called *limit points*, since they help determine when processes start and stop, and when views hold or not. Postulating the existence of limit points is an important challenge in learning QP models, since they are crucial for prediction.

QP theory does not describe how these models are learned. We claim that statistical accounts of causality (cf. Pearl, 2000; Gopnik *et al* 2004) can be harnessed to produce QP models. We incorporate statistics via similarity, using structure-mapping operations to construct probabilities as a side-effect of assimilating experiences. The SEQL model of generalization (Kuehne *et al* 2000) constructs generalizations incrementally via analogical comparison. We simulate analogical matching via SME, the Structure-Mapping Engine (Falkenhainer *et al* 1989; Forbus *et al* 1994). Given two structured representations, the *base* and *target*, SME computes one or two *mappings* which describe how the base and target can be aligned.

Mappings include a set of *correspondences* that detail exactly which entities and statements in one description go with entities and statements in the other, a *structural evaluation score* which indicates the overall quality of the match, and a set of *candidate inferences* that are conjectures about the target, using the correspondences to project partially unmapped base structures. Candidate inferences allow predictions and explanations to be generated without rules, via analogy to prior experiences and explanations. This makes them particularly important for accounts of learning like ours that postulate localized, incrementally generated models.

SEQL operates by maintaining a list of generalizations and exemplars. Given a new exemplar, SEQL compares it with existing generalizations. If it is sufficiently similar to one of them, it is assimilated into that generalization. Otherwise, it is compared against the list of unassimilated exemplars. If a pair of exemplars is sufficiently similar, they are combined to form a new generalization.

We call a set of generalizations and exemplars that are being processed together by SEQL a *generalization context*. Generalization contexts can be defined bottom-up, via similarity-based retrieval, or by labeling, e.g., a learner might use a generalization context to process all examples that have been given a verbal label, like "cat".

Learning Protohistories

Protohistories are generalizations of specific observed behaviors. Observed behaviors are typically rich with perceptual information, and in new domains, impoverished with regard to explanations. We postulate that analogical generalization, as modeled in SEQL, is used to construct prototypical behaviors. Below is an example observation given to our simulation. It describes an adult female human, swimming (gliding) in a still pond, and floating:

```
(isa bodyInLiquid0 AdultFemaleHuman)
(isa container0 Pond)
(isa liquid0 (LiquidFn Water))
(in-UnderspecifiedContainer liquid0 container0)
(massOfObject bodyInLiquid0 (Kilogram 60))
(volumeOfObject bodyInLiquid0 (CubicCentimeter 62039))
(isa gliding0 MovementEvent)
(primaryObjectMoving gliding0 bodyInLiquid0)
(isa stillLiquid0 StandingStill)
(doneBy stillLiquid0 liquid0)
(in-Floating bodyInLiquid0 liquid0).
```

The vocabulary of concepts and relations is drawn from the ResearchCyc knowledge base[1], an independently developed representation system for common-sense knowledge. The predicate calculus was produced using a natural-language understanding system (Kuehne & Forbus, 2004) from simplified English, to reduce tailorability.

The simplified English that generates the above predicate calculus observation is:

> The woman *bodyInLiquid0* floats in water *liquid0* in a pond *container0*. The mass of the woman *bodyInLiquid0* is 60 kilograms. The volume of the woman *bodyInLiquid0* is 62039 cubic centimeters. The woman *bodyInLiquid0* is moving but the water *liquid0* is standing still.

For SME processing, `isa` statements are automatically translated into attributes (i.e., `(AdultFemaleHuman bodyInLiquid0)`). SEQL generalizations abstract specific individuals (e.g., `bodyInLiquid0`) into anonymous individuals, not variables. Numerical parameters (e.g., `(Kilogram 60)`) are also abstracted into anonymous individuals, but their values are preserved in a distribution for each quantity in the generalization. These distributions are used to conjecture limit points below. We ignore memory retrieval in this simulation, and provide as input a stream of observations like the above.

How many generalization contexts should be used? Since SEQL automatically constructs multiple generalizations according to similarity, one possibility is to use a single context. The drawback with a single context is that it may not provide enough discrimination for learning. For example, to learn why things float, the learner must distinguish between floating and sinking examples. We have observed that SEQL may, because of attribute information, cluster cases from both types of situations into the same generalization. Consequently, we create separate generalization contexts for each possibility. Every generalization context incorporates a set of *entry patterns* that are tested against new exemplars. When a new exemplar satisfies the entry pattern for a generalization context, it is processed in that context. The same example can be processed in multiple contexts, since a learner might be learning multiple concepts at once.

Consider a learner trying to understand the distinction between floating and sinking, as well as sailboats sailing. Figure 1 illustrates the three example generalization contexts that would be used. If an exemplar arrives with `(SinkingEvent sinking0)` as a constituent fact, with no mention of floating, it will be incorporated into the rightmost context alone. If another exemplar arrives with `(isa boat0 SailBoat)` and `(floating-in boat0 (LiquidFn Water))` as constituent facts, it will be incorporated into both leftmost and middle contexts.

[1] http://research.cyc.com/

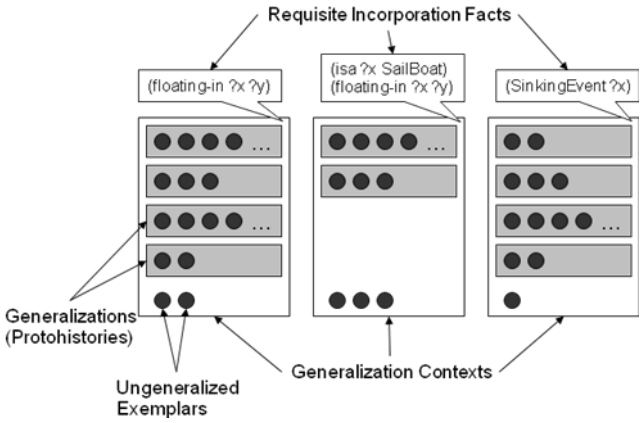


Figure 1: Example contextual protohistory organization

Learning a Causal Corpus

The causal corpus consists of a set of causal models grounded in, and connecting, protohistories. These causal models are local to particular protohistories or collections of protohistories. Restructuring these local models into general domain theories, of the kind typically used in qualitative reasoning, occurs only after a reasonable causal corpus has been constructed (Forbus & Gentner 1986). Even fragmentary causal models are quite powerful: Understanding what qualitative proportionalities hold in a protohistory yields a means of predicting the immediate consequences of parameter changes. Similarly, understanding quantity conditions that determine which protohistory represents the behavior that occurs in a situation enables predictions of state changes.

Our simulation uses three *causal learning strategies* – procedures that take protohistories and quantities as input, and generate causal hypotheses, expressible using the vocabulary of QP theory. We also describe a method for deriving complex quantities from constituent input quantities. We do not view this set of strategies as complete, but we believe they are a good starting point.

Analyzing quantity values enables us to hypothesize limit points, quantity conditions, and qualitative proportionalities. The *quantity condition strategy* identifies relevant ordinal relationships. The *limit point strategy* hypothesizes new causally-relevant values. The *quantity derivation strategy* hypothesizes compound quantities. We discuss each in turn.

Quantity Condition Strategy. Conditions for processes and views typically include ordinal relations between quantities. For instance, for a body to be floating in a liquid, its density must be less than the liquid's density. Quantity conditions are conjectured as follows:

1. Protohistories that summarize experience related to the target phenomenon are divided into two groups: those that express it (P^+) and those that do not (P^-).

2. For each protohistory p_i within ($P^+ \cup P^-$), the ordinal relationships $R_i = \{r_1, r_2, \ldots, r_n\}$ are identified that hold for every exemplar within P_i. The ordinal relationships tested are =, >, <, ≥, and ≤, over the set of exemplars that were used in forming P_i.

3. Conditions are identified that pertain to the entirety of P^+ and P^-, such that $R^+ = \{R^+_1 \cap \ldots \cap R^+_n\}$ and $R^- = \{R^-_1 \cap \ldots \cap R^-_n\}$.

4. Conditions that coincide with the phenomenon are the set $R_{cause} = R^+ - R^-$. Relationships that coincide with the absence of the phenomenon are the set $R_{prevent} = R^- - R^+$.

We use exemplars in step 2 because our encoding process does not automatically generate ordinal relationships from numerical values in observations. (The quantity value distribution information stored with generalizations cannot be used to compute this, because links to particular exemplars is not included.) This is a simplification: We believe that psychologically, encoding choices are driven in part by learning goals, which would propose encoding particular ordinal relationships in order to test conjectures via this strategy. Such goals might be generated based on trying various ordinals on a small number of exemplars, but that is left for future work.

Limit Point Strategy. Some physical phenomena occur when a quantity's value is above or below a specific limit point. Like the quantity condition strategy, the limit point strategy assumes that two sets of protohistories have been identified, such as water being heated and boiling and water being heated and not boiling. Recall that protohistories preserve the set of exemplar values $\{v_1, v_2, \ldots, v_n\}$ for each quantity. This information can be summarized via an interval V, where $V = [\min(v_1, v_2, \ldots, v_n), \max(v_1, v_2, \ldots, v_n)]$.

After calculating quantity intervals for individual protohistories, we first compute possible limit points by grouping protohistories into two sets: those that express the given phenomena $P^+ = \{p^+_1, p^+_2, \ldots, p^+_n\}$ and those that do not $P^- = \{p^-_1, p^-_2, \ldots, p^-_n\}$. For each quantity-type q, we merge the protohistory intervals so that

$$P^+_q = [\min(p^+_{1q}, p^+_{2q}, \ldots, p^+_{nq}), \max(p^+_{1q}, p^+_{2q}, \ldots, p^+_{nq})]$$
$$P^-_q = [\min(p^-_{1q}, p^-_{2q}, \ldots, p^-_{nq}), \max(p^-_{1q}, p^-_{2q}, \ldots, p^-_{nq})].$$

If the intervals P^+_q and P^-_q do not overlap for a quantity, it could be the case that a limit point exists within the interval $[\max(\min(P^+_q, P^-_q)), \min(\max(P^+_q, P^-_q))]$, or between the maximum point of the lower interval and the minimum point of the higher interval. This interval is then added to the causal corpus, as a limit point approximation.

If the intervals P^+_q and P^-_q overlap, there could still be an uninterrupted interval $[q_{min}, q_{max}]$ that represents a condition under which the phenomenon occurs. Instead of merging protohistory intervals into P^+_q and P^-_q, we test for exclusiveness, such that no protohistory intervals in P^+ overlap protohistory intervals in P^- for a quantity q.

Uninterrupted intervals in q are then added to the causal corpus as possible conditions for the target phenomenon.

Quantity Derivation Strategy. Understanding many physical phenomena requires introducing quantities beyond those observed. To understand why something floats versus sinks, for example, requires introducing the idea of density. If the quantity analysis fails to distinguish between two behaviors within the encoded quantities, the quantity derivation strategy proposes new quantities that are then searched for limit points and ordinal relationships. For all explicitly mentioned quantities a and b such that $a \neq b$, a set of new quantities C is derived:

$$C = \{a/b, b/a, a*b, a+b, a-b, b-a\}.$$

The units for the derived quantities may be identical to their constituent quantities ($kg + kg = kg$), or they may be combinations of their constituent units ($kg/cc = kg/cc$).

Simulation Results

We demonstrate how these methods combine to produce plausible causal corpus elements from a set of observations. We first go through a single learning task in detail, then summarize the results of others.

To investigate learning floating versus sinking, we encoded 30 unique exemplars – 16 floating and 14 sinking – in simplified English, which were fed into our natural language understanding system to automatically produce predicate calculus descriptions like our earlier example. Many factors used in the scenarios were based on Piaget's (1930) interviews with children: the motion of the water (still or wavy); the body in water (man, woman, log, cruise ship, or tree branch); the body of water (ocean, sea, lake, pond, bath-tub, or bowl); and autonomous motion of the body (moving/gliding or still). In all scenarios, a body floats when the body's density is less than 1 g/cc.

The simulation first generates protohistories from the exemplars. Two generalization contexts were used, with entry patterns `(in-Floating ?x ?y)` and `(isa ?x SinkingEvent)`, to model the focus on understanding when something floated or sank. The assimilation threshold for SEQL was set to 0.75. This yielded six protohistories, five for floating and one for sinking. All exemplars were assimilated into a generalization. Table 1 shows the protohistory abstractions with the generic entities in bold, and the protohistory size, |P|. Protohistories P_1 and P_4 preserved *tree branch* and *cruise ship* in their abstractions, respectively; the rest contain only generic entities.

The abstractions for P_2 and P_3 are identical, yet they are still distinct. This is due to uncertain facts within the generalization. Specifically, in P_2, P(body = man) = .66, and in P_3, P(body = woman) = .66. Thus, although the abstractions are identical, the underlying representations differ. Low-probability facts are considered for similarity processing, so they remained distinct.

Context	#	Protohistory Abstraction	|P|
Floating	1	Idle tree branch, wavy **water**	2
	2	Moving **body**	3
	3	Moving **body**	3
	4	Moving cruise ship	3
	5	Wavy **water**	5
Sinking	6	Idle **body**, still **water**	14

Table 1: Protohistories for floating and sinking

To generate causal corpus information for these protohistories, the strategies defined above were executed in the order given.

Given a set of protohistories, the simulation proceeds to analyze its quantities, searching for limit points and quantity conditions that help explain floating. The observable quantities yielded no causal hypotheses, so the simulation used the quantity derivation strategy to create new quantities and try again. One of the derived quantities does yield a limit point, as shown in Table 3. Since this limit point (which we know as density) was derived as the ratio of mass and volume, we also obtain the qualitative proportionalities shown in Table 3, imposing a causal direction on what was an algebraic relationship by assuming that observable parameters are more primitive than derived parameters. (This is a heuristic, of course, that could be incorrect – consider heat derived from temperature, for example.)

Causal Hypothesis Type	Formula
Derived Quantity	$q = \text{mass}_{body}/\text{volume}_{body}$
Limit Point	$q < [0.001, 0.00102]$ kg/cc
Qualitative Proportionality	*floatability* α_{Q-} q
	q α_{Q+} mass_{body}
	q α_{Q-} volume_{body}
	floatability α_{Q-} mass_{body}
	floatability α_{Q+} volume_{body}

Table 3: Causal hypotheses generated about floating

In addition to floating/sinking, we tested the simulation on two other learning scenarios. This involved creating new stimuli descriptions and changing the entry patterns of the generalization contexts to suit the scenarios. The remainder of the learning process remained the same.

To model learning how balance scales work (Siegler, 1983), we encoded nine scenarios using the methodology above, varying the kinds of objects on the balance and the posture of the object (e.g., sitting or kneeling or upright). Using two generalization contexts, one for *right-side sinking* and one for *left-side sinking*, the simulation generated two protohistories for each context. The

quantity condition strategy creates the sensible quantity hypothesis

```
(> (massOfObject leftside0) (massOfObject rightside0))
```

to predict when the left side will sink.

In another learning experiment conjecturing when boiling would occur, six exemplars were encoded using the methodology above. The limit point strategy conjectures a limit point for temperature to predict when boiling occurs:

```
Hypothesis: phenomena occurs when
 (temperatureOfObject kettle0)
 is above some point in the range:
 [95.0-100.0] DegreeCelsius.
```

The lower bound of the range could be refined by more experience.

While the number of learning experiments conducted to date is small, the results obtained so far are very reasonable.

Related Work

The closest previous simulation is COBWEB (Fisher, 1987) which utilized conceptual clustering, but did not introduce causal models, nor was it tested on semi-automatically generated stimuli. Our quantity derivation strategy is inspired by Langley's (1981) BACON simulation.

Some of diSessa's (1983) p-prims (for "phenomenological primitives") can be viewed as causal corpus elements while others may be viewed as protohistories. No computational model for learning them was ever implemented.

Discussion and Future Work

Our simulation combines symbolic, relational representations with quantity analysis to learn causal models. We think this is a very promising approach to developing deep qualitative models of physical domains.

In cognitive psychology, many advocates of statistical accounts of causality do not include any notion of mechanism, and we obviously (along with Chi *et al* 1994; Ahn *et al* 1995) do not believe that is sufficient. As demonstrated in this paper, generalization and quantity analysis can be used to generate fragmentary qualitative models of these causal mechanisms.

This simulation is obviously only a beginning. In addition to testing the simulation on a broader range of learning problems, we also plan to incorporate retrieval, using MAC/FAC (Forbus *et al* 1995). Having the simulation generate its own distinctions to explore, perhaps via failed predictions made with protohistories, is also an important problem to investigate.

Acknowledgments

This work was funded by the Office of Naval Research under grant N00014-08-1-0040.

References

Ahn, W. k., Kalish, C. W., Medin, D. L., & Gelman, S. A. (1995). The role of covariation versus mechanism information in causal attribution. *Cognition, 54*, 299-352.

Chi, M., Slotta, J., & De Leeuw, N. (1994). From things to processes: A theory of conceptual change for learning science concepts. *Learning and Instruction*, **4**(1), 27-43

diSessa, A. (1983). Phenomenology and the evolution of intuition. In D. Gentner and A. Stevens (Eds), *Mental Models*. 15-33. Hillsdale, NJ: Lawrence Erlbaum.

Falkenhainer, B., Forbus, K. & Gentner, D. (1989). The Structure-Mapping Engine: Algorithm and examples. Artficial Intelligence. 41.

Fisher, D.H. (1987). Knowledge acquisition via incremental concept clustering. Machine Learning 2, pp. 139-172.

Forbus, K. (1984). Qualitative Process Theory. *Artificial Intelligence*, 24, 85-168.

Forbus, K.D & Gentner, D. (1986). Learning physical domains: towards a theoretical framework. In R.S. Michalski, J.G. Carbonell and T.M. Mitchell" *Machine Learning: An Artificial Intelligence Approach, Vol. 2.*

Forbus, K., Gentner, D. & Law, K. (1995). MAC/FAC: A model of Similarity-based Retrieval. *Cognitive Science*, 19(2), April-June, pp 141-205.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2), 155-170

Gopnik, A., Glymour, C., Sobel, D., Schulz, L., Kushnir, T., & Danks, D. (2004). A theory of causal learning in children: Causal maps and Bayes nets. *Psychological Review, 111, 1*, 1-31.

Kuehne, S. & Forbus, K. (2004). Capturing QP-relevant information from natural language text. *Proceedings of the 18th International Qualitative Reasoning Workshop*, Evanston, Illinois, USA, August

Kuehne, S.E., Forbus, K.D., Gentner, D., & Quinn, B. (2000). SEQL: Category learning as progressive abstraction using structure mapping. *Proceedings of CogSci 2000*, August.

Langley, P. (1981). Data-driven discovery of physical laws. *Cognitive Science* 5, 31-54.

Pearl, J. (2000). *Causality:Models, Reasoning, and Inference*. Cambridge University Press.

Piaget, J. (1930). The Child's Conception of Physical Causality. Kegan Paul, London.

Siegler, R. (1983). Five generalizations about cognitive development. *American Psychologist* 38, 263-277.

Immersive Examination of the Qualitative Structure of Biomolecules

Kenny Gruchalla
gruchall@colorado.edu
Department of Computer Science
University of Colorado at Boulder

Mark Dubin
dubin@colorado.edu
Department of Molecular, Cellular, and Developmental Biology
University of Colorado at Boulder

Jonathan Marbach
marbach@colorado.edu
Department of Computer Science
University of Colorado at Boulder

Elizabeth Bradley
lizb@colorado.edu
Department of Computer Science
University of Colorado at Boulder

Abstract

The geometry of biomolecules dictates their function, but reasoning about that structure is difficult because of their 3D complexity and the range of scales involved. The wooden or plastic ball-and-stick models that are common in high-school chemistry labs help people reason about these issues when the molecules involved are small, but they are useless in the study of large biomolecules. Largely for this reason, 3D computer visualization tools have become essential in this field. However, these tools are limited by their interfaces. Traditional graphics workstations project a 3D model onto 2D screen, and interaction with the 3D model is indirect, using 2D mouse or pointing device. Immersive visualization is a potential solution to this: it allows a user to visualize a biomolecule in 3D and interact with it directly in 3-space. This paper reports upon a pilot study about the effects of immersive visualization upon an expert's reasoning about the qualitative structure of these molecules. We ported a standard visualization application (PyMOL) to a CAVE-like immersive virtual environment (IVE), then invited three separate biochemistry research groups—people who use PyMOL routinely on desktop computers—to examine their favorite molecule in the IVE. Within ninety minutes of immersive investigation, each group reported a new discovery about the qualitative structure of that molecule. We believe that the immersive environment facilitated these discoveries by supporting and facilitating the natural spatial reasoning abilities of its users.

An immersive virtual environment is a combination of hardware and software that provides a psychophysical experience of being surrounded by a computer-generated scene (see Figure 1). Immersive virtual environments provide users with an egocentric three-dimensional perspective: users are immersed in a virtual world, where they can explore complex spatial systems by looking through them, walking around them, and viewing them from different perspectives. Immersive environments may help people see and understand the structure of complex three-dimensional datasets; in contrast to more traditional graphics workstations, these environments allow one to visualize data using the well-practiced, non-concious analysis that automatically accompanies an embodied, egocentric visual perspective. There are several studies that have investigated the added value of immersive environments (Pausch, Proffitt, & Williams 1997; Ruddle, Payne, & Jones 1999; Arns, Cruz-Neira, & Cook 1999; Swan *et al.* 2003; Gruchalla 2004; Schulze *et al.* 2005; Demiralp *et al.* 2006). However, the results of these studies are mixed and the issue is somewhat controversial. There are few studies that clearly demonstrate the effectiveness of immersive environments for real-world problems, and none that approach this issue from the standpoint of qualitative reasoning. Our study does so, and our results indicate that experts understand more about the geometry of biomolecules if they use an immersive environment than if they use the same visualization tools on a standard desktop. Within ninety minutes of immersive investigation, each of the three groups in our study reported a new discovery about the qualitative structure of an important biomolecule—molecules that these groups had been studying for years in with the same software visualization tool on desktop environments.

Immersive visualization has long been proposed as a means to analyze the complex three-dimensional structure of biological molecules (Ihlenfeldt 1997), and it is used by numerous investigators in basic research and industrial settings. Qualitative spatial analysis of the structure of these molecules at a range of scales is essential, because their overall three-dimensional configuration dictates the atomic interactions that are the basis of their function. Understanding the geometry of the building blocks of a biomolecule, and their relationships, is key to many of the grand-challenge problems in biochemistry: the rational design of drugs that enhance or inhibit molecular activity, the understanding of how steps in embryonic development normally proceed or go wrong in the presence of genetic mutations of molecular structure, and so on.

This paper documents a pilot study in which three separate groups of biochemists visualized and interacted with individual biological molecules in a CAVE-like IVE. In each case, the immersive working session yielded new insights that the same biochemists had not previously achieved with their extensive use of the same visualization package on standard desktop computer displays. Large-scale spatial features, such as pockets and ridges, were readily identi-

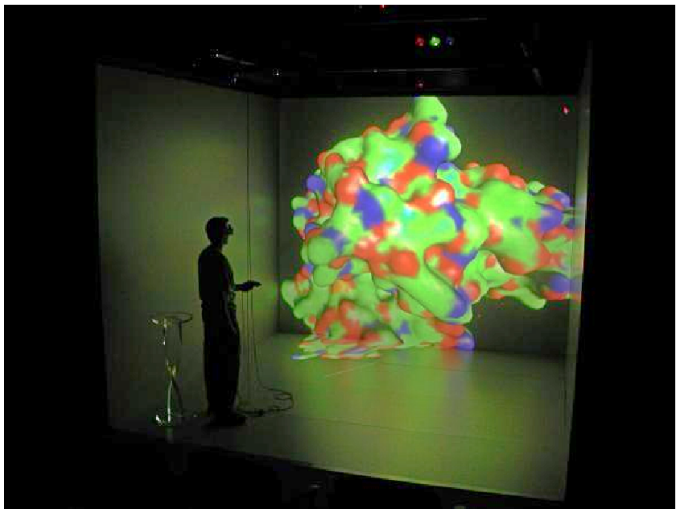


Figure 1: A user interacting with a PyMOL visualization of a molecular surface inside a CAVE-like immersive virtual environment, which provides the opportunity to visualize the molecule using normal, everyday-world perceptual abilities that have been tuned and practiced from birth.

fied when walking around the molecule displayed at human scale.

Methods

Three University of Colorado at Boulder (UCB) biochemistry research groups were invited to study a molecule of their choice—one central to their current research—in a FakeSpace Flex, a CAVE-like immersive virtual environment. The research groups had each intensively studied their chosen molecule using *non-immersive* visualization techniques—the desktop version of PyMOL, a popular open-source molecular visualization system (DeLano 2002)—for at least a year prior to conducting their research in the IVE. We ported this same tool to a stereoscopic, interactive IVE (Gruchalla, Marbach, & Dubin 2007) to provide some informal control in our study.

The Flex is configurable large-screen projection-based 12'x12'x10' theater, consisting of four walls: three rear-projected screens measuring 12'x10' that form the right wall, back wall, and left wall of the IVE. The fourth wall is the 12'x12' floor that is projected from above. A three-dimensional effect is created inside the IVE through active stereo projection and motion parallax. Stereo projection is achieved by projecting two images in sequence on each screen: an image for the viewer's left eye, followed by an image for the viewer's right eye. Viewers wear active stereo LCD shutter glasses to view the stereoscopic images. Infrared emitters synchronize the glasses with the graphics pipes. When the computer renders the image for the left eye, the right eye shutter is closed. Similarly, when the computer renders the image for the right eye, the left eye shutter is closed. This shuttering action creates the illusion of three-dimensional images. A motion parallax is supported by tracking the position and orientation of the viewer's head and using this information to generate an egocentric perspective. Virtual objects can be manipulated inside the IVE using a tracked wand.

PyMOL (DeLano 2002) is a powerful and versatile open-source, cross-platform real-time molecular visualization system that supports standard representations for molecular structures (e.g., wire bonds, cylinders, spheres, ball-and-stick, dot surfaces, solid surfaces, wire meshes, backbone ribbons, and cartoon ribbons). PyMOL's primary interface is an embedded Python interpreter, which is the basis for its sophistication. Our immersive port of PyMOL allows users to view PyMOL visualizations in a head-tracked IVE and manipulate molecular structures using a six-degree-of-freedom input device. Only the visualization and 3D interaction elements of PyMOL were ported to the IVE; its python-based command-line interface ran on a desktop computer. The visualization is *composed* (e.g., loading pdb files, choosing representations, selecting colormaps, ...) using the PyMOL command-line interface on this desktop, then viewed and manipulated in the IVE. Clearly, an IVE is poorly suited to support a command-line interface. Dividing the workflow between the two environments allows all the power and sophistication of the command-line interface to be used to construct the 3D model, while the visualization of the model and the spatial reasoning about its nature can be done in the 3D space of the IVE.

In this environment, three biochemistry groups conducted actual research about how the structure of their molecule relates to its function:

- The laboratory of Professor Arthur Pardi studying the anti-VEGF aptamer (Ruckman *et al.* 1998)
- The laboratory of Professor Natalie Ahn studying the extracellular signal-regulated kinase ERK2 (1erk.pdb) (Zhang *et al.* 1994)
- The laboratory of Professor Shelley Copley studying the enzyme maleylacetoacetate isomerase (1fw1.pdb) (Polekhina *et al.* 2001)

With one exception[1], the participants had no previous experience in viewing or manipulating objects in the IVE. Each group was given a brief introduction to the environment and how to manipulate molecular structures using the wand. Each group worked for about 90 minutes, with three of four members of the team working collaboratively inside the IVE, while one team member controlled the content of the visualization from a desktop computer using the PyMOL command-line and desktop interfaces. This similar to a traditional team working session, in which one member group would control the visualization from a desktop computer using the PyMOL command-line and desktop interface; however, in a traditional working session the rest of the team would gather around the computer to view and try to understand the resulting visualization.

[1]Professor Pardi had toured the immersive facilities and seen several immersive demos prior to the pilot study.

Results

Despite having a long and extensive research history with their respective molecules, all three groups arrived at a new insight from their 90-minute IVE research session. All of these insights were similar, and all involved qualitative reasoning about geometry. Each group became newly aware of a large spatial feature, such as an empty space or ridge, that they had not noticed during their (considerable) previous PyMOL work with the molecule on desktop computer monitors. In each case, the newly recognized feature led to insights about the molecule's function that follow directly from geometry: how its pieces move, for instance, or how they fit together. These are described in the following paragraphs. Each group left with the intention of exploring a new theoretical possibility based on these insights; the Ahn group actually integrated a hypothesis concerning the structure into a new grant proposal.

The Copley group recognized an empty pocket indenting from the surface in the enzyme maleylacetoacetate isomerase (MAAI) (see Figure 2). MAAI is normally a dimer, in which the interface between the dimer molecules blocks this pocket. However, the monomer of MAAI is similar to—and is used by the Copley group as—a model for another molecule, tetrachlorohydroquinone (TCHQ) dehalogenase, which is a monomer. This group is studying how a key component of the molecule's active site, amino acid cysteine at position 16 (cys16), interacts with substrate molecules that must be able to diffuse into TCHQ dehalogenase in order to reach cys16. The pocket represents a large enough opening for such entry, with cys16 lying at its base (darkened area in Figures 2c and 2d). When viewing this molecule on workstations, the researchers had discounted this region as a potential active site of TCHQ dehalogenase because they did not judge it to be spacious enough for the substrate to penetrate to cys16. The immersive visualization gave the researchers the ability to stand inside the pocket, which gave them enough information to reverse their decision.

The Ahn group was interested in a long "ridge" of potentially interacting amino acids that link two important sites of the enzyme ERK2 (see Figure 3). ERK2 is crucial component of the machinery that underlies normal and malignant cell production. Previous experiments had determined that small conformational changes caused by mutations of amino acids in the region shown in orange can cause a change in shape all the way across the molecule, in the region shown in purple. Biochemists are interested in understanding how such conformational changes are transferred across molecules, and the Ahn group used the IVE port of PyMOL to investigate this issue in ERK2. During their immersive investigation, they recognized the "ridge" between these two regions (shown in green in figure 3) as a possible physical linkage. This ridge could "transmit" changes in the orange region across the molecule to the purple region.

The Pardi group was interested in understanding how the complex surface regions of two molecules fit together in a complementary way. They used the IVE's six-degree-of-freedom handheld wand to manipulate the positions and orientations of the regulatory molecule VEGF and the anti-VEGF aptamer. VEGF has been implicated in human macular degeneration, a progressive disease that causes loss of high-acuity, central vision. The synthetic, anti-VEGF aptamer has been shown to be effective in slowing the progression of macular degeneration; however, the ability of anti-VEGF aptamer to inhibit the VEGF molecule is deterimined by the quality of the fit between the two. Using the interactive capabilities of the IVE, the Pardi group discovered a new possible fitting between the two molecules (see Figure 4).

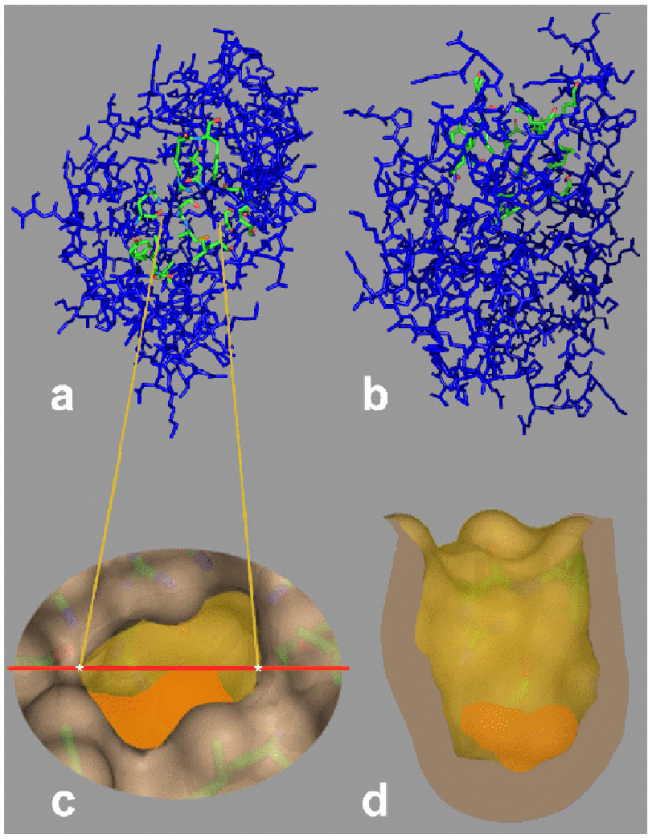


Figure 2: A molecular "pocket" that was discovered in the IVE: At top, the molecule (maleylacetoacetate isomerase) is shown in stick-representation with the region of interest shown with bright, non-dark-blue sticks; (a) is a view looking down into the pocket, (b) is a side view of the molecule at the same scale. The bottom two images show partial surface views of the region of the molecule immediately surrounding the pocket, with the approximate inside "surface" of the pocket in gold, and the amino acid cys16 in orange. (c) is a view looking down into the pocket; the mouth of the pocket corresponds to the region shown in the stick representation, as indicated by the lines. For (d), the pocket was bisected by the plane indicated by the line (x-axis) in (c), and rotated 90 degrees about the x-axis to yield a view of half of the pocket seen in side view, corresponding to the portion of the pocket at the top of (c). Panels (c) and (d) are to the same scale; the width of visible pocket in (c) is approximately 10 angstroms.

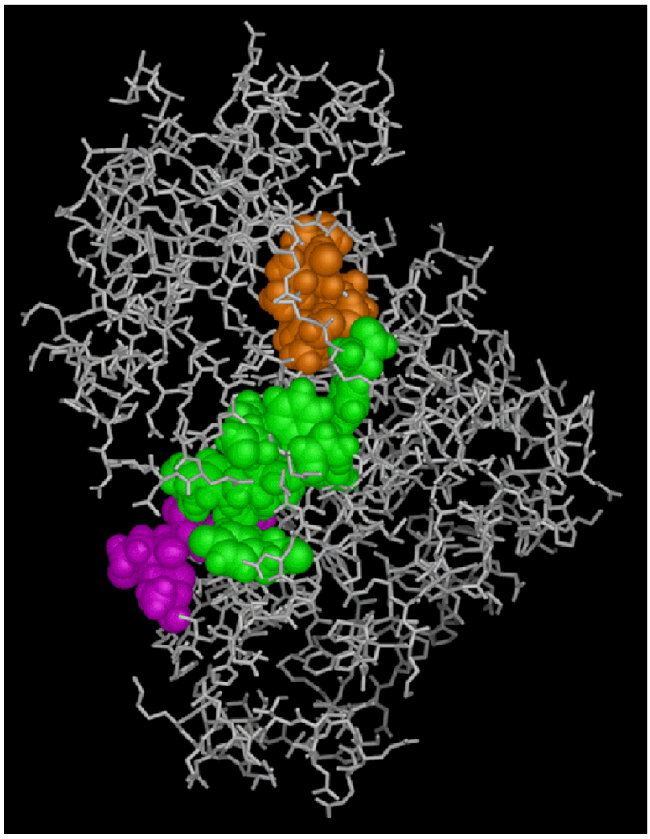


Figure 3: A stick-figure model of ERK2 with regions of interest shown in space-fill representation. Mutations in the orange region are known to cause shape changes in the magenta region. The "ridge" of green colored atoms, recognized by the Ahn group in the IVE, is a possible linkage between these regions.

Discussion

Reasoning about relationally generated space (such as a pocket in a molecule) depends on the scale of presentation, varying points of view, and movement in and around it. Thus, we suggest it is not surprising that the naturalistic IVE display allowed each of the three research groups to recognize important spatial features that they had previously overlooked in small, flat-screen, computer displays that must be indirectly manipulated with a mouse. The importance of naturalistic viewing is supported by numerous elegant experiments involving real-world, normal-size scenes (Purves & Lotto 2003; Yang & Purves 2003). Probabilistic matching of images like this provides a good explanation of numerous visual phenomena. This is not surprising; perceptual experience has molded our genetic makeup and it is tuned by the learning that each of us accumulates as we exist and develop day-to-day (Geary & Huffman 2002). This kind of knowledge is wired—and continuously rewired—in the neural circuitry of our brain, based on polysensory activities. That is, cognitive structures are developed from perception and action, and grounded in the physical interactions with the

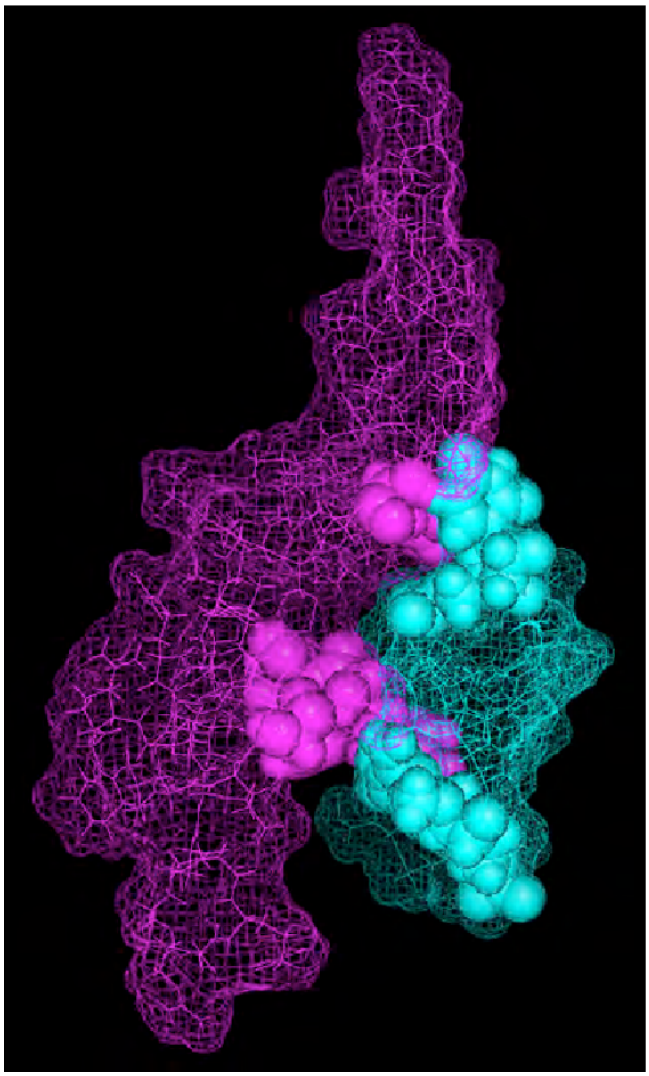


Figure 4: The Pardi group used the 3D interactive wand inside the IVE to investigate how the loop region of anti-VEGF aptamer (cyan) fits to a site on the outside of VEGF (magenta). The surfaces of both regions are show as meshes surrounding stick-figure representations, with atoms considered important shown in space-fill view.

environment (Pecher & Zwaan 2005).

In this context, it makes complete sense that working in an IVE allows people to reason more effectively about the geometry of biomolecules. Studies that demonstrate this effect are surprisingly rare, though, and the added value of immersion is controversial in the visualization community. This study is part of a larger effort that addresses this broader issue: a general definition of conditions under which the use of fully interactive, three-dimensional, immersive visualization adds value to research activities. As indicated in the previous section, our results suggest that short, intense sessions of IVE viewing valuably augment more extensive, non-IVE based research of molecular function. The underlying reasons for this, we believe, are threefold:

- First, spatial judgments are body-relative in everyday activity (Hatfield 2003). It is easier, for example, to judge whether you could crawl through a passageway in a cave if you are in the cave and looking at the passageway than if you are examining a five-inch-tall rendering of it on a flat screen monitor. Our hypothesis is that examining a molecule *at a human scale* made it easier for the biochemists to reason about the different spatial structures. Because the IVE presented the biomolecules at a natural and familiar scale—similar to the way many complex-shaped, everyday objects appear in the world—it facilitated effective reasoning about there shape.
- Second, the egocentric perspective improves spatial reasoning, object recognition, and stresses the role of action in building knowledge. Much of this happens automatically: people do not stop and think about how to move their heads or bodies in order to get a better view of something. The IVE supports this very naturally. Its unique features—natural body movements and well-practiced automatic brain function as the basis for examination of the structure in question—is consistent with recent research on *embodied cognition*, cognition that is based on perceptual knowledge accumulated through what we have encountered and manipulated with our bodies as we move within and examine the world (Wilson 2002; Wolputte 2002).
- Finally, the collaborative nature of the environment facilitates collaborative reasoning about the data. The large scale of the environment allowed multiple biochemistry researchers to gather inside the environment simultaneously. All the groups commented that they found working collaboratively in the IVE to be much easier than crowding around a small computer screen. The large scale made it easy to see what atoms and regions another member of the group was referring to. Often they used bodily references to direct each other, such as, "that group of bonds near your left shoulder."

Reasoning about the geometry of objects has a long and rich history in the qualitative reasoning field, and there are interesting papers about ontologies, paradigms, techniques, and applications for this in every QR workshop—beginning with an augmented version of Hayes's "pieces of stuff" ontology that was presented at QR '87 for reasoning about collections of molecules (Collins 1987). A few QR systems have been built over the years specifically for reasoning about molecular structure (Bandini, Cattaneo, & Stofella 1988). Most of the geometry-related work in the QR community has involved mechanical devices, an application (like biomolecules) where shape and function are intimately inter-related. Iwasaki, Joskowicz, Nielsen, and Faltings have made significant contributions to this over the years (Joskowicz 1987; Iwasaki 1987; Nielsen 1987; 1988; Faltings, Baechler, & Kun 1991; Tessler, Iwasaki, & Law 1993; Faltings 1993; Sun & Faltings 1994; Joskowicz & Sacks 1997). There has also been some work in the QR community that considers the cognitive science perspective along with the representation and the geometry, notably from Ken Forbus's group (Ferguson & Forbus 1999; Forbus, Ferguson, & Usher 2000; Forbus, Tomai, & Usher 2003; 2005; Lovett, Dehghani, & Forbus 2006).

The study reported here has a much more complicated application area than most of these papers, and much less lofty aims. We are not trying to simulate, design, or deduce anything. We rely on the human experts to figure out what's meaningful; we want simply to understand how immersive environments support their reasoning about the geometry that factors into that determination. Because of the comparative nature of our study, the ontology and the model are pre-specified. Our goal is not to figure out whether a better model or ontology exists for these purposes, as in many interesting QR papers, e.g., (Pacheco, Escrig, & Toledo 2002) but rather to study how the presentation & interface affects the spatial reasoning about the molecules. We are not trying to generalize ideas about structure across application domains, as in (Adorni *et al.* 1988) nor are we trying build more-abstract modelling paradigms, as in the elegant work of Escrig (which is concerned with many of the concepts that arise here, like how things fit together) (Museros & Escrig 2004). There are obviously many interesting problems to tackle involving the kinematics & dynamics of the biomolecules in our study, as well as the role of geometry in those processes, but these are "grand-challenge" problems and outside our scope.

Conclusion

This pilot study suggests that immersive environments enhance the ability of human experts to reason about the geometry of complex biomolecules. It also contributes further evidence to the general debate about the added value of large-scale immersive environments in the investigation of complex interactive spatial domains. The small sample size and lack of formal controls, however, mean that the results are only preliminary. The discoveries reported by the scientists in the study may have been facilitated by the opportunity to use embodied perceptual mechanisms afforded by the environment. Comments from the subjects suggest that the environment may have also provided a much improved collaborative atmosphere. Regardless of the specifics mechanisms, the results are very promising: all three user tests in this study generated a new piece of science as a result of their improved geometric reasoning about a complex problem.

Acknowledgements

We thank Geoffrey Dorn, Gwen Pech and Mick Coady of the University of Colorado-Boulder, BP Center for Visualization for their assistance, support and advice. We are grateful to the members of the research groups who participated in this study. Professor Pardi was especially helpful in defining the early stages of this project and in choosing PyMOL for this work. We thank Sara Klingenstein for assisting in the initial research on theoretical considerations. This project was supported by a University of Colorado Butcher Award to Professors Dubin and Pardi and by equipment donations from NVIDIA.

References

Adorni, G.; Burdese, M.; Del Grosso, A.; Loddo, R.; and Zucchini, A. 1988. A qualitative approach to structural mechanics. In *Proceedings of the International Workshop on Qualitative Reasoning about Physical Systems*.

Arns, L.; Cruz-Neira, C.; and Cook, D. 1999. The benefits of statistical visualization in an immersive environment. In *VR '99: Proceedings of the IEEE Virtual Reality 1999 (VR'99)*, 88–95.

Bandini, S.; Cattaneo, G.; and Stofella, P. 1988. A theory for molecule structures: The molecular onthology theory. In *Proceedings of the International Workshop on Qualitative Reasoning about Physical Systems*.

Collins, J. W. 1987. Reasoning about fluids via molecular collections. In *Proceedings of the International Workshop on Qualitative Reasoning about Physical Systems*.

DeLano, W. 2002. The pymol molecular graphics system. http://www.pymol.org.

Demiralp, C.; Jackson, C.; Karelitz, D.; Zhang, S.; and Laidlaw, D. H. 2006. Cave and fishtank virtual-reality displays: A qualitative and quantitative comparison. *IEEE Transactions on Visualization and Computer Graphics* 12(3):323–330.

Faltings, B.; Baechler, E.; and Kun, S. 1991. Efficient qualitative kinematics. In *Proceedings of the International Workshop on Qualitative Reasoning about Physical Systems*.

Faltings, B. 1993. Qualitative structural analysis using diagrammatic reasoning. In *Proceedings of the International Workshop on Qualitative Reasoning about Physical Systems*.

Ferguson, R., and Forbus, K. 1999. Georep: A flexible tool for spatial representation of line drawings. In *Proceedings of the International Workshop on Qualitative Reasoning about Physical Systems*.

Forbus, K. D.; Ferguson, R. W.; and Usher, J. M. 2000. Towards a computational model of sketching. In *Proceedings of the International Workshop on Qualitative Reasoning about Physical Systems*.

Forbus, K. D.; Tomai, E.; and Usher, J. 2003. Qualitative spatial reasoning for visual grouping in sketches. In *Proceedings of the International Workshop on Qualitative Reasoning about Physical Systems*.

Forbus, K. D.; Tomai, E.; and Usher, J. 2005. Solving everyday physical reasoning problems by analogy using sketches. In *Proceedings of the International Workshop on Qualitative Reasoning about Physical Systems*.

Geary, D., and Huffman, K. 2002. Brain and cognitive evolution: forms of modularity and functions of mind. *Psychol. Bull.* 128(5).

Gruchalla, K.; Marbach, J.; and Dubin, M. 2007. Porting legacy applications to immersive virtual environments: A case study. In *Proceedings of 2007 International Conference on Computer Graphics Theory and Applications*, 179–184.

Gruchalla, K. 2004. Immersive well-path editing: Investigating the added value of immersion. In *VR '04: Proceedings of the IEEE Virtual Reality 2004 (VR'04)*, 157–164.

Hatfield, G. 2003. Representation and constraints: the inverse problem and the structure of visual space. *Acta Psychol. (Amst.)* 114:355–378.

Ihlenfeldt, W. 1997. Virtual reality in chemistry. *Journal of Molecular Modeling* 3:368–402.

Iwasaki, Y. 1987. Generating behavior equations from explicit representation of mechanisms. In *Proceedings of the International Workshop on Qualitative Reasoning about Physical Systems*.

Joskowicz, L., and Sacks, E. 1997. Qualitative and quantitative mechanical assembly design. In *Proceedings of the International Workshop on Qualitative Reasoning about Physical Systems*.

Joskowicz, L. 1987. Shape and function in mechanical devices. In *Proceedings of the International Workshop on Qualitative Reasoning about Physical Systems*.

Lovett, A.; Dehghani, M.; and Forbus, K. 2006. Solving everyday physical reasoning problems by analogy using sketches. In *Proceedings of the International Workshop on Qualitative Reasoning about Physical Systems*.

Museros, L., and Escrig, M. T. 2004. A qualitative theory for shape representation and matching. In *Proceedings of the International Workshop on Qualitative Reasoning about Physical Systems*.

Nielsen, P. 1987. A qualitative approach to mechanical constraint. In *Proceedings of the International Workshop on Qualitative Reasoning about Physical Systems*.

Nielsen, P. 1988. Qualitative mechanics: Envisioning the clock. In *Proceedings of the International Workshop on Qualitative Reasoning about Physical Systems*.

Pacheco, J.; Escrig, M. T.; and Toledo, F. 2002. Qualitative spatial reasoning on three-dimensional orientation. In *Proceedings of the International Workshop on Qualitative Reasoning about Physical Systems*.

Pausch, R.; Proffitt, D.; and Williams, G. 1997. Quantifying immersion in virtual reality. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, 13–18. ACM Press/Addison-Wesley Publishing Co.

Pecher, D., and Zwaan, R. 2005. Introduction to grounding cognition. In Pecher, D., and Zwaan, R., eds., *Grounding*

cognition, 1–7. Cambridge, England: Cambridge University Press.

Polekhina, G.; Board, P.; Blackburn, A.; and Parker, M. 2001. Crystal structure of maleylacetoacetate isomerase/glutathione transferase zeta reveals the molecular basis for its remarkable catalytic promiscuity. *Biochemistry* 40(6):567–576.

Purves, D., and Lotto, B. 2003. In *Why We See What We Do: An Empirical Theory of Vision*. Sunderland, MA: Sinauer.

Ruckman, J.; Green, L.; Beeson, J.; Waugh, S.; Gillette, W.; Henninger, D.; Claesson-Welsh, L.; and Janjic, N. 1998. 2'-fluoropyrimidine rna-based aptamers to the 165-amino acid form of vascular endothelial growth factor(vegf(165)) - inhibition of receptor binding and vegf-induced vascular permeability through interactions requiring the exon 7-encoded domain. *Journal of Biological Chemistry* 273:20556–20567.

Ruddle, R.; Payne, S.; and Jones, D. 1999. Navigating large-scale virtual enviroments: What differences occur between helmet-mounted and desk-top displays? *Presence: Teleoperators and Virtual Environments* 8:157–168.

Schulze, J.; Forsberg, A.; Kleppe; Zeleznik, R.; and Laidlaw, D. H. 2005. Characterizing the effect of level of immersion on a 3D marking task. In *Proceedings of HCI International*.

Sun, K., and Faltings, B. 1994. Supporting creative mechanical design. In *Proceedings of the International Workshop on Qualitative Reasoning about Physical Systems*.

Swan, J. E.; Gabbard, J. L.; Hix, D.; Schulman, R. S.; and Kim, K. P. 2003. A comparative study of user performance in a map-based virtual environment. In *VR '03: Proceedings of IEEE Virtual Reality 2003 (VR'03)*, 259–266.

Tessler, S.; Iwasaki, Y.; and Law, K. 1993. Qualitative structural analysis using diagrammatic reasoning. In *Proceedings of the International Workshop on Qualitative Reasoning about Physical Systems*.

Wilson, M. 2002. Six views of embodied cognition. *Psychon. Bull. Rev.* 9(4):625–636.

Wolputte, S. V. 2002. Hang on to your self: of bodies, embodiment, and selves. *Ann. Rev. Anthropol.* 33:251–269.

Yang, Z., and Purves, D. 2003. A statistical explanation of visual spaces. *Nat. Neurosci.* 6(6):632–640.

Zhang, F.; Strand, A.; Robbins, D.; Cobb, M.; and Goldsmith, E. 1994. Atomic structure of the map kinase erk2 at 2.3 a resolution. *Nature* 367:704–711.

Qualitative Abstraction of Piecewise Affine Systems

Michael W. Hofbaur and **Theresa Rienmüller**
Department of Automation and Control, Graz University of Technology
Kopernikusgasse 24/2, 8010 Graz, Austria
{michael.hofbaur, theresa.rienmueller}@TUGraz.at

Abstract

Qualitative or symbolic abstractions of hybrid systems received considerable interest recently to solve problems of hybrid systems estimation, control and verification symbolically. To abstract a hybrid system one has to slice the continuously valued input/output/state-space into a (finite) set of partitions. The number of partitions potentially grows exponentially with the dimension of the space. As a consequence, one has to divide the spaces carefully in order to obtain a manageable abstraction. This paper presents a systematic procedure to partition the state-space of piecewise affine (PWA) systems into qualitatively distinct regions. As a consequence, we obtain a moderately large set of partitions that characterises the hybrid dynamics of the PWA system. The abstraction scheme helps also to keep the number of so called *spurious* behaviors of qualitative simulation small, in particular when compared to the typically used grid-based abstractions.

Introduction

Complexity in hybrid systems analysis, estimation and control arises from the close interaction between the system's mode-dependent continuous dynamics and discrete mode changes. Optimal hybrid estimation, for example, has to consider all possible hybrid trajectories that the system can exhibit and performs the associated numerical filtering process for each trajectory. Since the number of trajectories grows exponentially over time, it is easy to see that sub-optimal and computationally efficient methods are key for any real-time operation of hybrid estimation.

A tempting approach is to use finite (qualitative) abstractions of the continuous dynamics together with the discrete dynamics of the hybrid system and re-formulate the hybrid estimation/control task in a pure discrete way. The rich toolset of Qualitative Reasoning and Model Checking can then be used to solve tasks for analysis, simulation, verification, estimation and control. However, there is no free lunch. A qualitative model for the continuously valued dynamics requires a finite abstraction of the input/output/state-space of the hybrid model. Input and output space partitions can arise naturally through quantisation. The state-space abstraction, however, is more demanding. In particular, as the number of partitions potentially grows exponentially with the dimension of the continuous state. Another difficulty is that qualitative models allow trajectories that the underlying real system cannot show. These so called *spurious behaviors* (Kuipers 1994) can significantly deteriorate the reasoning result, for example, in that one fails to prove stability of a hybrid control system.

This paper provides an approach for the qualitative abstraction of *piecewise affine (PWA)* systems that addresses both issues mentioned above. We propose a state-space abstraction scheme that uses distinct features of the system's continuous and discrete dynamics. As a result we obtain a separation that partitions the state-space in *qualitatively distinct regions* only and thus keeps the number of partitions moderate. Another benefit is that our partitioning scheme reduces the number of spurious behaviors compared to using a state-space abstraction through a hyper-dimensional grid.

Related Research

Qualitative or symbolic abstractions of dynamic systems are a major theme in Qualitative Reasoning (Weld & de Kleer 1990; Kuipers 1994), a fruitful branch of AI. Our work on qualitative abstraction of a system's state-space has its origins in the pioneering work of Yip, Zhao and Bailey-Kellogg (Yip 1991; Yip & Zhao 1996; Bailey-Kellogg, Zhao, & Yip 1996) that provide symbolic abstractions for complex non-linear dynamics. Whereas they use advanced reasoning methods for system's analysis, Lunze and coworkers (Lunze 1994; Lunze, Nixdorf, & Schröder 1999; Schröder 2003) build their qualitative abstraction upon the concept of stochastic automata and use them to mainly solve diagnosis problems.

Timed automata (Alur & Dill 1994), a specific class of hybrid systems, lead themself to a symbolic model and thus allow one to apply analysis and verification methods from computer science, such as bisimulation. This line of research received considerable interest, e.g. (Alur *et al.* 2000) and much effort was devoted to extending the applicability of symbolic approximations and bisimulation techniques to solve analysis, verification and control problems for other, more general, classes of hybrid systems (Tiwari 2003; Girard & Pappas 2006; Tabuada 2007). Most recent research limits its scope to systems with 'strong' stability properties so that the artifacts of discrete approximation, i.e. spurious behaviors, do not prevent one from applying bisimulation type analysis techniques. All of these techniques face also the curse of dimensionality. Discrete abstractions of contin-

uous state-spaces lead to a number of domains for the state that grows exponentially with the state's dimension. We cannot fully avoid this difficulty for our proposed stochastic automata encoding of PWA systems, but provide an abstraction technique that slices the state-space carefully according to qualitative distinctions of the system's dynamics. This keeps the number of state partitions moderate and, as a nice side-effect, actively reduces spurious behaviors. To deal with complexity, we can further draw upon our work to efficiently encode stochastic automata in an OBDD-like fashion (Kleissl & Hofbaur 2005).

PWA Systems

A widely adopted and versatile class of hybrid systems are the so-called *piecewise affine (PWA)* systems. PWA systems specify a hybrid model with continuously valued state $\mathbf{x} = [x_1, \ldots, x_{n_x}]^T$, input $\mathbf{u} = [u_1, \ldots, u_{n_u}]^T$ and output $\mathbf{y} = [y_1, \ldots, y_{n_y}]^T$. The model specifies dynamics in discrete-time (sampling period T_d) through the affine discrete-time model

$$\mathbf{x}_{k+1} = \mathbf{A}_i\mathbf{x}_k + \mathbf{B}_i\mathbf{u}_k + \mathbf{f}_i \quad (1)$$

$$\mathbf{y}_k = \mathbf{C}_i\mathbf{x}_k + \mathbf{D}_i\mathbf{u}_k + \mathbf{g}_i \ , \quad (2)$$

where the subscript $i = 1, \ldots, l$ of the model-parameter $\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i, \mathbf{D}_i, \mathbf{f}_i, \mathbf{g}_i$ stands for the *mode* or PWA dynamics that is valid in a particular domain $\mathcal{D}_i$ of the combined state/input space. More specifically, the state traverses at mode i whenever

$$\begin{bmatrix} \mathbf{x}_k \\ \mathbf{u}_k \end{bmatrix} \in \mathcal{D}_i \ .$$

To guarantee *uniqueness* for the PWA trajectories one ensures that the domains $\mathcal{D}_i$ specify a *non-overlapping separation* of the state/input space $\mathcal{D}$. A domain $\mathcal{D}_i$ is usually defined through a *polyhedral* partition of the combined state/input space that can be expressed through constraints of the form

$$\mathbf{G}_i^x\mathbf{x}_k + \mathbf{G}_i^u\mathbf{u}_k \leq \mathbf{G}_i^c \ .$$

For the scope of this paper we use a slightly weaker domain specification that allows us to abstract the continuous state and input space separately, more specifically, we select a mode i, whenever the two inequalities hold

$$\begin{array}{rrcl} \mathbf{x}_k \in \mathcal{D}_{x,i}: & \mathbf{G}_i^x\mathbf{x}_k & \leq & \mathbf{G}_i^{cx} \\ \mathbf{u}_k \in \mathcal{D}_{u,i}: & \mathbf{G}_i^u\mathbf{u}_k & \leq & \mathbf{G}_i^{cu} \ . \end{array} \quad (3)$$

Again, to guarantee uniqueness of PWA trajectories, the partitions $\mathcal{D}_i = \mathcal{D}_{x,i} \times \mathcal{D}_{u,i}$ do not overlap, i.e. $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset, i \neq j$.

Example

Figure 1 shows trajectories[1] for an autonomous PWA system with $l = 3$ modes and a 2-dimensional state-space that is limited to

$$-6 \leq x_1 \leq +6, \quad -3 \leq x_2 \leq +3 \ . \quad (4)$$

[1] As in many real-world applications of hybrid systems, we obtain the PWA model through sampling of the continuous time dynamics shown in Fig. 1.

Mode switching occurs at $x_1 = -2$ (between modes 1 and 2) and at $x_1 = +2$ (between modes 2 and 3). Our discrete-time PWA model operates at a sampling period $T_d = 0.1$ [sec.] and defines the dynamics as follows: PWA Mode 1 specifies dynamics with a stable equilibrium at $\mathbf{x}_r = [-3 \quad 0]^T$, eigenvalues $z_1 = z_2 = e^{-2T_d}$ and an eigenvector $\mathbf{p} = [1 \quad -1]^T$. The dynamics of mode 2 are characterised through an unstable equilibrium (saddle point) at the origin, eigenvalues $z_1 = e^{-T_d}, z_2 = e^{+T_d}$ and associated eigenvectors $\mathbf{p}_1 = [2 \quad 1]$, $\mathbf{p}_2 = [2 \quad -1]$. Finally, at mode 3 the system exhibits an undamped oscillatory behavior with equilibrium $\mathbf{x}_r = [2 \quad 0]^T$ and frequency $\omega = 1$. We will use the same scalar measurement $y_k = x_{1,k} + x_{2,k}$ for all three PWA modes.

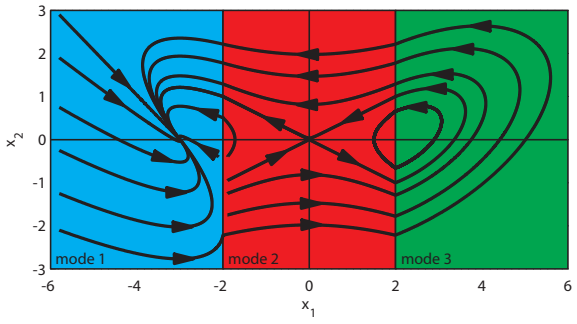


Figure 1: 3-mode PWA system

Qualitative PWA Model

A qualitative PWA model abstracts the continuously valued affine dynamics (1-2) symbolically. Thus, it merges the continuous dynamics with its discrete dynamics (mode changes) into one common discretely valued behavioral description. For this purpose, one defines *partitions* for the continuously valued state-, input- and output-space. The input- and output-space partitions can arise naturally through quantisation, for example, whenever one deals with real-world signals with low resolution (e.g. 4-bit A/D converter). In terms of the input-space, we only have to make sure, that the qualitative abstraction allows us to formulate the input inequality of the PWA guard condition (3). The state-space abstraction can either be derived recursively during qualitative reasoning (Kuipers 1994) or verification (Girard & Pappas 2006) or the partitions are specified explicitly prior compiling an automaton abstraction for the PWA model. Since we intend to use a qualitative model for fast on-line reasoning, we choose the second form and compute a so-called *stochastic automaton* from the PWA model that encodes the model's dynamics through a state machine with stochastic transition specification.

Stochastic Automaton PWA Model

A stochastic automaton (Lunze 1994; Bukharaev 1995; Schröder 2003) defines a tuple

$$\mathcal{A} = \langle \mathcal{X}, \mathcal{U}, \mathcal{Y}, P_{\mathcal{T}}, P_{\mathcal{O}} \rangle \ , \quad (5)$$

where $\mathcal{X} = \{X_1, \ldots, X_{N_x}\}$, $\mathcal{U} = \{U_1, \ldots, U_{N_u}\}$ and $\mathcal{Y} = \{Y_1, \ldots, Y_{N_y}\}$ denote the finite domains for the automaton state $\bar{x}$, input $\bar{u}$ and output $\bar{y}$, respectively. With $\bar{x}_k$, $\bar{u}_k$ and $\bar{y}_k$ we denote valuations of the state, input, and output at a particular *time-step* k. The behavior of the automaton is captured through the conditional *transition-* and *observation-probabilities*[2]

$$\begin{aligned} P_{\mathcal{T}}(\bar{x}_{k+1}, \bar{x}_k, \bar{u}_k) &= P(\bar{x}_{k+1}|\bar{x}_k, \bar{u}_k) \\ P_{\mathcal{O}}(\bar{y}_k, \bar{x}_k, \bar{u}_k) &= P(\bar{y}_k|\bar{x}_k, \bar{u}_k) \ . \end{aligned} \tag{6}$$

A stochastic automaton can be almost directly used as a *qualitative model* of our PWA system (1-3). The only entity that we have to add is a map $M : \mathcal{X} \times \mathcal{U} \rightarrow \{1, 2, \ldots, l\}$ that specifies the PWA mode for every qualitative state/input pair. This enables us to define the qualitative abstraction of a PWA system as an extended stochastic automaton through the tuple

$$\mathcal{A}_{pwa} = \langle \mathcal{X}, \mathcal{U}, \mathcal{Y}, P_{\mathcal{T}}, P_{\mathcal{O}}, M \rangle \ . \tag{7}$$

Automaton compilation: To compile a stochastic automaton $\mathcal{A}_{pwa}$ with pre-defined state/input/output-space partitions one has to compute the conditional probabilities $P_{\mathcal{T}}(\cdot) = P(X_i|X_j, U_\zeta)$ and $P_{\mathcal{O}}(\cdot) = P(Y_i|X_j, U_\zeta)$ for all triples $\{i, j, \zeta\}$ according to the PWA dynamics. For this purpose one assumes a uniform distribution for $\mathbf{x}_k \in X_j$ and $\mathbf{u}_k \in U_\zeta$ and computes the distribution on state and output space for $\mathbf{x}_{k+1}$ and $\mathbf{y}_k$. This can be done, for example, through sampling or hyper-box mapping (Schröder 2003). Compilation of the stochastic automaton is computationally expensive. However, once we have compiled a stochastic automaton $\mathcal{A}_{pwa}$ for the PWA system, we can use this automaton model to efficiently perform qualitative simulation, estimation or control.

State-Space Abstraction

Qualitative abstraction through stochastic automaton compilation requires us to divide the continuous state-space

$$\mathcal{D}_x = \bigcup_{i=1,\ldots,l} \mathcal{D}_{x,i} \subset \mathbb{R}^{n_x}$$

into a finite set of non-overlapping partitions $\{\mathcal{D}_{x,1}, \ldots, \mathcal{D}_{x,N_x}\}$ where each partition $\mathcal{D}_{x,i}$ represents a qualitative abstraction or *state* $X_i = [\mathcal{D}_{x,i}]$ of the stochastic automaton. This has to be done carefully, since the number N_x potentially increases exponentially with the dimension n_x of the continuous PWA dynamics. On the other hand, one has to provide sufficiently fine partitions to retain the characteristics of the continuous dynamics.

Grid-based abstraction

The simplest abstraction of a bounded domain $\mathcal{D}_x$ in state-space is to apply an n_x-dimensional grid with fixed or adaptive grid-size. A grid-based abstraction happens naturally, whenever one partitions each state-variable individually and obtains the overall partition of the state-space through the cross product. This often leads to unsatisfactory results since the number of partitions explodes as the dimension of the system increases and the associated qualitative models do not constrain possible behaviors well enough. Simulating such a model would predict an unnecessarily large number of *spurious behaviors*, i.e. behaviors that the original (PWA) system cannot show. Figure 2 illustrates this property for mode 2 of our autonomous PWA system. The eigenvectors $\mathbf{p}_1$ and $\mathbf{p}_2$ of the dynamic matrix A_2 uniquely partition the state-space, however, the grid is not conform with this separation. Take the three partitions X_1, X_2, X_3 of Fig. 2. The continuous behavior at mode 2 clearly allows transitions $\mathbf{x}_k \in X_1 \rightarrow \mathbf{x}_{k+1} \in X_2$ and $\mathbf{x}_k \in X_2 \rightarrow \mathbf{x}_{k+1} \in X_3$. A stochastic automaton would encode these facts in terms of the transition probabilities $P_{\mathcal{T}}(X_2, X_1) \neq 0$ and $P_{\mathcal{T}}(X_3, X_2) \neq 0$. As a consequence, a simulation on the basis of this stochastic automaton would predict the qualitative behavior $X_1 \rightarrow X_2 \rightarrow X_3$, a behavior that the original PWA system cannot show! Another difficulty with a grid-based abstraction is to select the appropriate grid-size. An upper bound for the number of partitions N_x will be most likely the limiting factor for grid-size selection. However, it is difficult to judge, whether the resulting abstraction is fine enough to capture the details of the mode's continuous dynamics unless one performs exhaustive simulation studies.

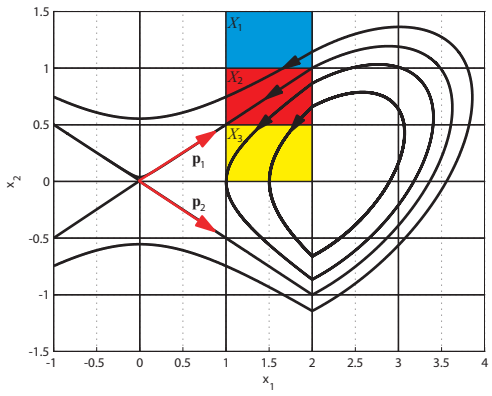


Figure 2: Boxed state-space abstraction

Besides abstracting the continuous dynamics, we have to make sure that the state-space partitions are also conform with the discrete dynamics of the PWA system. In detail, we have to ensure that the state-space abstraction allows us to formulate the state inequality of the PWA guard condition 3. PWA mode domains are in general polyhedral partitions, so that an abstraction through hyper-boxes in state-space can be inadequate.

These arguments illustrate that it is desirable to have a qualitative abstraction of the state-space that (1) respects the properties of the continuous dynamics as well as (2) captures the polyhedral specification of PWA mode domains.

[2] In general, one would define a *behavioral relation* $P_{\mathcal{Q}}(\bar{x}_{k+1}, \bar{y}_k, \bar{x}_k, \bar{u}_k) = P(\bar{x}_{k+1}, \bar{y}_k, |\bar{x}_k, \bar{u}_k)$ that defines $P_{\mathcal{T}}$ and $P_{\mathcal{O}}$ as its boundary distributions. However, the successor state $\mathbf{x}_{k+1}$ and output $\mathbf{y}_k$ of our PWA system are stochastically independent so that $P_{\mathcal{Q}}(\cdot) = P_{\mathcal{T}}(\cdot)P_{\mathcal{O}}(\cdot)$ holds and $P_{\mathcal{T}}$ and $P_{\mathcal{O}}$ represent the same information as $P_{\mathcal{Q}}$.

Qualitative abstraction of continuous dynamics

The example above indicates that we need a more general state-space separation technique that builds upon (non-overlapping) polyhedral partitions. We propose to use the *eigenvectors* or in general *hyper-planes* that are defined through the eigenvectors of the PWA dynamics (or their associated dynamic matrices $\mathbf{A}_i$) to partition the state-space into qualitatively distinct regions. For second order PWA dynamics at mode i that is characterised through the dynamic matrix $\mathbf{A}_i$ with eigenvalues $z_1, z_2, z_1 \neq z_2$ we can write

$$\mathbf{x}_{k+1} = \mathbf{A}_i\mathbf{x}_k + \mathbf{f}_i = \alpha_1\mathbf{p}_1 z_1 + \alpha_2\mathbf{p}_2 z_2 + \mathbf{f}_i \ ,$$

where $\mathbf{p}_j$ $(j = 1, 2)$ denotes the eigenvector for z_j and the parameters α_1 and α_2 are given through

$$\mathbf{x}_k = \alpha_1\mathbf{p}_1 + \alpha_2\mathbf{p}_2 \ .$$

It directly follows from Systems Theory (Hirsch & Smale 1974) that the eigenvectors, centred at the equilibrium point

$$\mathbf{x}_r = (\mathbf{I} - \mathbf{A}_i)^{-1}\mathbf{f}_i \ ,$$

partition the state-space into distinct regions. Each region is characterised through the signs of α_1 and α_2, clearly a qualitative distinction!

Eigenvector-partitioning works with eigenvectors for real-valued eigenvalues z_j, but not for complex-valued eigenvectors/eigenvalues as in mode 3 of our PWA example. In order to abstract these behaviors that spiral round the equilibrium point, we propose to partition the state-space into sectors differently. The complex-valued eigenvectors uniquely characterise the orientation of the elliptical behavior, whereas the eigenvalues determine its stability character. As a consequence, we can use the eigenvectors to determine the ellipse-axes for the spiralling behavior and use these axes to partition the state-space into 4 sectors. This abstraction enables us to uniquely characterise the direction of the behavior (clockwise or counter-clockwise) but not the mode's stability property. Ideally, one would want to use elliptical regions in state-space to specify a Lyapunov-function for an asymptotically stable equilibrium point, for example. However, the approximation of elliptical regions through polyhedral domains is impracticable. One way to overcome this difficulty is to use a combination of sectors and hyper-boxes to enclose the ellipsoids and use additional reasoning concepts (Hofbaur & Dourdoumas 2001) that go beyond the scope of standard stochastic automata theory.

An additional qualitative characterisation of the state-space is given through the sign of a state $\mathbf{x}_k$ or its components $x_{i,k}$, in particular. Consequently, we suggest to additionally partition the state-space according to the unit vectors $\mathbf{e}_1, \ldots, \mathbf{e}_{n_x}$. Figure 3 shows the result of the combined sign, eigenvector and ellipse-axes based state-space separation for our PWA example along with possible trajectories.

Adding inputs and noise: Up to now, we used an autonomous PWA system. However, we intend to use a qualitative model for estimation and control and thus, we have to deal with a non-zero control input $\mathbf{u}_k = [u_{1,k}, \ldots, u_{n_u,k}]^T$

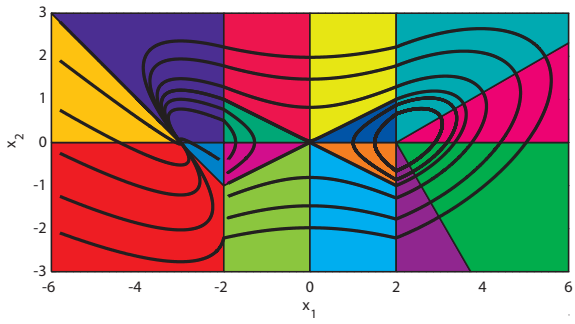


Figure 3: 3-mode PWA system with state-space partitions and system trajectories

and disturbances. We use a typical PWA notation (Kvasnica *et al.* 2006) and introduce bounded additive disturbances $\mathbf{w} = [w_1, \ldots, w_{n_x}]^T$ that act upon the state-variable through

$$\mathbf{x}_{k+1} = \mathbf{A}_i\mathbf{x}_k + \mathbf{B}_i\mathbf{u}_k + \mathbf{f}_i + \mathbf{w}_k \ . \tag{8}$$

We characterise the disturbance in terms of a bounding polytope $\mathcal{W}$, for example, a hyperbox that defines min/max values for each $w_i, i = 1, \ldots, n_x$.

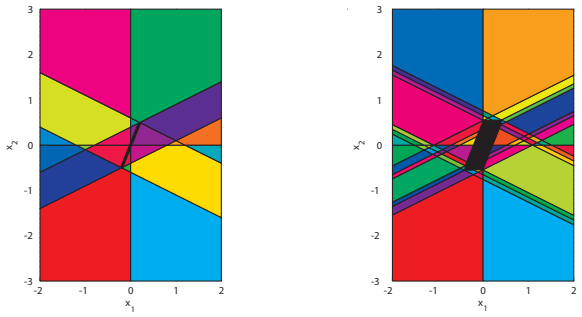
Let us deal with a scalar input u and an associated input vector $\mathbf{B}_i = [\mathbf{b}_i]$ for (8) first. A constant input $u_k = u^*$ shifts the equilibrium point $\mathbf{x}_r$ of mode i to

$$\mathbf{x}_r = \left(\mathbf{I} - \mathbf{A}_i\right)^{-1}\left(\mathbf{b}_i u_k + \mathbf{f}_i\right) \ . \tag{9}$$

Therefore, an input $u_{min} \leq u_k \leq u_{max}$ at time-step k can be interpreted as shifting the origin for our eigenvalue/ellipse-axes based state-space separation according to (9). To partition the state-space into qualitatively distinct regions, we apply our separation scheme at the two extremal points of $\mathbf{x}_r$ ($u_k = u_{min}$ and $u_k = u_{max}$). Figure 4a shows the resulting separation for the PWA mode 2 with an input vector $\mathbf{b}_2 = [0.0992 \quad 0.0075]^T$ and $u_{min} = -1.0$, $u_{max} = 1.0$. The bar at the origin indicates the region for $\mathbf{x}_r$.

Dealing with multiple inputs is straightforward. Through combination of all min/max values for the inputs, we obtain a region (polytope) $\mathcal{X}_r$ in state-space for $\mathbf{x}_r$. We then perform the state-space separation at all extremal points of $\mathcal{X}_r$.

A disturbance $\mathbf{w} = [w_1, \ldots, w_{n_x}]^T$ can be handled in two ways. First, we can treat each disturbance w_j as an additional input (with input vector $\mathbf{e}_j$) and use the noise bounds to enlarge $\mathcal{X}_r$. This adds additional 2^{n_x} extremal points to $\mathcal{X}_r$ and thus introduces many additional partitions of the state-space. Figure 4b shows the resulting separation for bounded noise $|w_i| \leq 0.01, i = 1, 2$. The black region in the center indicates $\mathcal{X}_r$. The second way to deal with disturbances is to perform state-space separation for deterministic inputs only and include the effects of disturbances through the probability specifications of the resulting stochastic automaton. We prefer this approach since it keeps the number of state-space partitions small.



(a) Separation for $|u| \leq 1$

(b) Separation for $|u| \leq 1$ and $|w_i| \leq 0.01, i = 1, 2$

Figure 4: State-space separation for mode 2

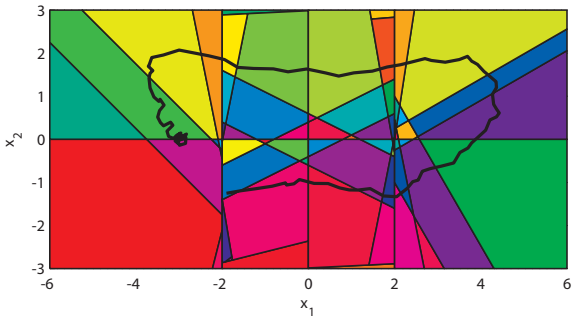


Figure 5: Overall state-space abstraction and PWA trajectory example

Abstraction of PWA mode changes

The PWA model operates on discrete-time. As a result, switching does not occur exactly on the mode-boundary but within its vicinity. Therefore, it is essential to capture the regions in state-space where switching can occur. Additionally, we observed that simulation and particularly PWA mode estimation results improve whenever one specifies also those regions in state-space from where we can reach the switching domains within one time-step. Both domains can be computed easily through a 1-step forward/backward reachability analysis. As before, we can include disturbances in two ways. Directly, in terms of an inclusive reachability analysis, or indirectly through the transition probabilities of the stochastic automaton. Again, we prefer the latter approach.

Combining behavior-based abstractions

The overall state-space abstraction combines the partitions from the dynamics-based separation scheme with the partitions for mode-change characterisation and computes a set of N_x non-overlapping polytopes $\{X_1, \ldots, X_{N_x}\}$ that partitions the continuous state-space into qualitatively distinct regions. Figure 5 shows the partitions for our non-autonomous PWA system with a scalar input u that acts through the input vectors

$$\mathbf{b}_1 = [0.0775\ 0.0585]^T,\ \mathbf{b}_2 = [0.0992\ 0.0075]^T,$$
$$\mathbf{b}_3 = [0.0515\ 0.0022]^T$$

along with a trajectory for $u_k = \sin(0.1k)$ and disturbance $|w_i| \leq 0.1, i = 1, 2$.

Qualitative Estimation

Our main application of the stochastic automaton model is to perform hybrid estimation which can be formulated through the following recursive belief update process

$$b^{(j)}_{k|k-1} = \sum_{X_i \in \mathcal{X}} P_{\mathcal{T}}(X_j, X_i, U_\zeta) b^{(i)}_{k-1} \quad (10)$$

$$b^{(j)}_{k} \propto P_{\mathcal{O}}(Y_\kappa | X_j, U_\xi) b^{(j)}_{k|k-1} \quad (11)$$

that computes the belief (or probability) $b^{(j)}_k$ for $\mathbf{x}_k \in X_j$, given the qualitatively abstracted input values and measurement, i.e. $\mathbf{u}_{k-1} \in U_\zeta$, $\mathbf{u}_k \in U_\xi$ and $\mathbf{y}_k \in Y_\kappa$.

For our example we use an input abstraction with 3 partitions, and a 4-bit measurement resolution ($2^4 = 16$ partitions). Figure 6 shows the PWA mode estimation result that we obtained through selecting the PWA mode with the maximum cumulated belief for the trajectory of Fig. 5 that starts at $\mathbf{x}_0 = [-1.9 \quad -1.25]^T$. We used the non-autonomous system with the input $u_k = \sin(0.1k)$ and disturbances $|w_i| \leq 0.1, i = 1, 2$. The estimation starts with no state knowledge, thus $b^{(j)}_0 = 1/N_x$ and requires some steps to focus. Estimation with our state-space abstraction (69 partitions) provides an estimation result that is similar to a grid-based abstraction with $9 \times 9 = 81$ partitions.

To judge estimation quality better, we performed additional experiments with random initial states. In order to highlight the effects of state-space abstraction and stochastic automaton estimation we used an idealised setting with perfect initial knowledge. This eliminates the focusing process at the beginning of an experiment. We generated 1000 random initial states and simulated the non-autonomous PWA system for 100 time-steps[3]. Hybrid estimation with our state-space separation scheme provides on average a PWA mode estimation error of 3.55%. In comparison, a boxed scheme provided 4.25% for a 6×6 grid, 3.10% for a 9×9 grid and 2.97% for a 12×12 grid.

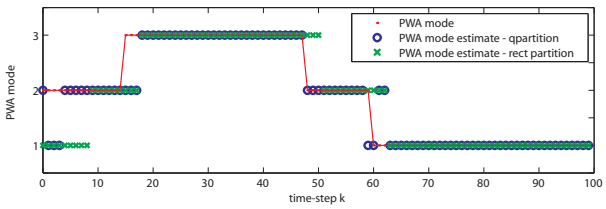


Figure 6: PWA-mode estimation result

[3]Most trajectories in our system get absorbed into the equilibrium point of mode 1 after about 100 time-steps.

Conclusion

The usual approach to abstract continuously-valued state-spaces is to use a grid-based abstraction of non-overlapping hyper-boxes. This requires one to select an appropriate grid-size that (a) is sufficiently fine to capture the system's dynamics and (b) is sufficiently coarse to keep the number of partitions manageable. To overcome this difficulty, we proposed a state-space abstraction scheme for PWA systems that uses *qualitative features* of the system's dynamics to partition the state-space into behavioral distinct regions. We present this abstraction for 2-dimensional systems to introduce the concepts concisely, however, we should note that it is equally well suited for higher order systems.

We used this abstraction to compile a stochastic automaton model for the PWA system and evaluated its estimation capabilities. We performed a random set of experiments and obtained evidence that our abstraction scheme leads to an estimation quality that is comparable with hyper-box abstractions that use a similar number of state-space partitions. However, in contrast to hyper-box approximations where one has do decide the grid resolution manually, we provide a quantisation scheme that *automatically selects an appropriate resolution* according to the system's dynamics. A more detailed analysis of qualitative simulation and estimation capabilities, in particular for higher order systems, is subject to ongoing research.

Our main motivation for a stochastic automata encoding is to formulate hybrid estimation and control schemes that use the discrete abstraction to quickly pre-select feasible and good estimation/control candidates. A consecutive numerical refinement can either validate an estimation/control candidate or identify spurious solutions and reject them. Our initial studies for such approaches (Kleissl & Hofbaur 2005; Kleissl 2006; Richter 2006) showed, that a good qualitative abstraction that avoids spurious behaviors is essential for this strategy. The results of this paper are an important step towards our proposed estimation and control schemes. However, they surely are also valuable for other works in hybrid systems analysis and verification.

Acknowledgement

This research is funded by the Austrian Science Fund (FWF) under grant P20041-N15.

References

Alur, R., and Dill, D. 1994. A theory of timed automata. *Theoretical Computer Science* 126:183–235.

Alur, R.; Henzinger, T.; Lafferriere, G.; and Papas, G. 2000. Discrete abstractions of hybrid systems. *Proceedings of the IEEE* 88(7):971–984.

Bailey-Kellogg, C.; Zhao, F.; and Yip, K. 1996. Spatial aggregation: Language and applications. In *Proceedings of the 10th International Workshop on Qualitative Reasoning (QR96)*, 3–11.

Bukharaev, R. 1995. *Theorie der stochastischen Automaten*. Teubner Verlag.

Girard, A., and Pappas, G. 2006. Verification using simulation. In Hespanha, J., and Tiwari, A., eds., *Hybrid Systems: Computation and Control, HSCC 2006*, volume 3927 of *Lecture Notes in Computer Science*. Springer Verlag. 272–286.

Hirsch, M., and Smale, S. 1974. *Differential Equations, Dynamical Systems, and Linear Algebra*. Academic Press.

Hofbaur, M. W., and Dourdoumas, N. 2001. Lyapunov based reasoning methods. *IEEE Transactions on Systems, Man, and Cybernetics - Part A* 31(4):546–558.

Kleissl, W., and Hofbaur, M. W. 2005. A qualitative model for hybrid control. In *Proceedings of the 19th International Workshop on Qualitative Reasoning (QR05)*, 8–16.

Kleissl, W. 2006. *Control of Multi Component Hybrid Systems through Qualitative Pre-Selection*. PhD thesis, Faculty of Electrical Engineering, Graz University of Technology, Graz, Austria.

Kuipers, B. 1994. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. Cambridge, MA: MIT Press.

Kvasnica, M.; Grieder, P.; Baotic, M.; and Christophersen, F. 2006. Multi-parametric toolbox (mpt). Technical report, Institut für Automatic, ETH - Swiss Federal Intstitute of Technology Zürich.

Lunze, J.; Nixdorf, B.; and Schröder, J. 1999. Deterministic discrete-event representations of linear continuous-variable systems. *Automatica* 35(3):395–406.

Lunze, J. 1994. Qualitative modelling of linear dynamical systems with quantized state measurements. *Automatica* 30(3):417–431.

Richter, S. 2006. *Probabilistic Hybrid Estimation for PWA Systems through Qualitative Pre-Selection*. MSc thesis, Institute of Automation and Control, Graz University of Technology, Graz, Austria.

Schröder, J. 2003. *Modelling, State Observation and Diagnosis of Quantized Systems*, volume 282 of *LNCIS*. Springer Verlag.

Tabuada, P. 2007. Approximate simulation relations and finite abstractions of quantized control systems. In Bemporad, A.; Bicchi, A.; and Buttazzo, G., eds., *Hybrid Systems: Computation and Control, HSCC 2007*, volume 4416 of *Lecture Notes in Computer Science*. Springer Verlag. 529–542.

Tiwari, A. 2003. Approximate reachability for linear systems. In Maler, O., and Pnueli, A., eds., *Hybrid Systems: Computation and Control, HSCC 2003*, volume 2623 of *Lecture Notes in Computer Science*. Springer Verlag. 514–525.

Weld, D., and de Kleer, J., eds. 1990. *Qualitative Reasoning about Physical Systems*. Morgan Kaufmann.

Yip, K., and Zhao, F. 1996. Spatial aggregation: Theory and applications. *Journal of Artificial Intelligence Research* 5:1–26.

Yip, K. 1991. Understanding complex dynamics by visual and symbolic reasoning. *Artificial Intelligence* 51:179–221.

Intelligent Support for Authoring 'Graph of Microworlds' based on Compositional Modeling Technique

Tomoya Horiguchi
Graduate School of Maritime Sciences,
Kobe University
5-1-1, Fukaeminami, Higashinada, Kobe,
Hyogo, 658-0022 Japan
horiguti@maritime.kobe-u.ac.jp

Tsukasa Hirashima
Department of Information Engineering,
Hiroshima University
1-4-1, Kagamiyama, Higashihiroshima,
Hiroshima 739-8527 Japan
tsukasa@isl.hiroshima-u.ac.jp

Abstract

In simulation-based learning environments (SLEs), in order to make students understand the domain theory systematically, it is important to sequence a set of microworlds of various complexity (from relatively simple systems/phenomena to more complicated ones) adaptively to the context of learning. We previously proposed *Graph of Microworlds (GMW)* which is a framework for indexing a set of microworlds based on their models. By using GMW, it is possible to design a function for adaptively selecting the microworld a student should learn next, and for assisting him in transferring between microworlds. However, it isn't easy to describe GMW because, for model-based indexing, an author must have the expertise of model generation in the domain. In this paper, therefore, we propose a method for semi-automating the description of GMW by introducing a mechanism of model generation based on the compositional modeling technique. This method makes it possible to assist the author in generating a set of indexed microworlds and also assist him in considering educational meanings of the relations between microworlds. We present how to design such a function and also illustrate how it works in describing a simple GMW for the domain of mechanics.

Introduction

In science education, it has been proved that simulation-based learning environments (SLEs), in which students can experience various phenomena in the domain (e.g., physics) by computer simulations, are very useful (Towne, 1995; Towne et al., 1993; Wenger, 1987). In SLEs, the range of (physical) systems and their behaviors is usually limited from some educational viewpoint in order for students to be able to understand the laws/principles behind the observed phenomena. This is called a *microworld*. In order to make students understand the whole theory of the domain systematically, therefore, it is necessary to sequence a set of microworlds of various complexity (from relatively simple systems/phenomena to more complicated ones) adaptively to the context of learning.

In designing such a function, it is essential to appropriately index a set of microworlds. Most of the current SLEs and authoring systems for them, by indexing a set of microworlds with the *labels* which represent their educational objectives and/or difficulty, provide the framework for sequencing them according to various teaching strategies (Merrill, 1999; Murray et al., 2003, for example). However, in order to facilitate a conceptual understanding of the domain, it is important to explain why, in the situation given by a microworld, the laws/principles are applicable and why the model is valid. It is also important to explain why/how the model changes if the situation is changed. In order to make such explanations, it is necessary to index a set of microworlds based on their *models* not mere labels.

Therefore, for adaptive sequencing of a set of microworlds in SLEs which aim at a conceptual understanding of the domain, we proposed a *Graph of Microworlds (GMW)*, which is a framework for indexing the microworlds and the relations between them based on their models (Horiguchi and Hirashima, 2005). We also indicated that, by using the ability in model-based inference of this framework, it becomes possible to design a function for adaptively selecting the microworld to which a student should transfer next (i.e., which he should learn next), and a function for assisting a student in transferring between microworlds.

Though GMW provides sufficient indices for designing the above functions, it isn't easy to describe a GMW. An author should make a set of microworlds and organize them by indexing the microworlds and the relations between them based on their models, that is, he must have the expertise in model generation process in the domain. Most of the (non-programmer) authors, therefore, would have great difficulty in describing GMW.

In this paper, therefore, we propose a method for semi-automating the description of GMW by introducing a mechanism of model generation based on expertise in the domain (i.e., *compositional modeling* (Falkenhainer and Forbus, 1991; Levy et al., 1997). This method makes it possible for the author not only to organize a set of microworlds by model-based indexing, but also make a set of microworlds which compose a GMW. The functions for such assistance are designed by using the method's ability to generate the model semi-automatically which embodies the given law(s)/principle(s) in the domain, and to infer how the model changes if the situation is changed.

In this paper, we first describe the related work (section 2), and then illustrate a GMW and how it works for designing the above functions (section 3). In section 4, we discuss

the difficulties in describing GMW and present the framework of our method for assisting the author with compositional modeling technique. In section 5, we present how to describe the domain knowledge in order to implement our method and illustrate how it works. In section 6, we make some concluding remarks.

Related Work

Murray classifies the current ITS-authoring systems and the functions of ITSs built with them into *pedagogy-oriented* and *performance-oriented* (Murray et al., 2003). The former systems, focusing on the function for sequencing learning contents indexed at a relatively shallow level adaptively to the context of learning (i.e., global guidance and planning in the whole domain), pay main attention to the representation of indices and teaching strategies. The latter systems, focusing on providing rich learning environments about each learning content (i.e., authentic content and feedback on errors), pay main attention to the representation of the domain-specific phenomena and problem solving processes.

Current SLE-authoring systems and the SLEs built with them belong to the latter. That is, most of them pay main attention to making the precise model of a specific content (i.e., microworld), but have the framework for indexing learning contents only at a relatively shallow level (Merrill, 1999; Murray et al., 2003, for example).

However, especially in SLEs which aim at a conceptual understanding of the domain, it becomes necessary to index the microworlds and the relations between them at a deeper level, that is, based on their models. The importance of considering differences between models in the SLEs which have multiple microworlds has been pointed out in earlier systems. For example, Burton et al. proposed a methodology for assisting students' progressive learning with a sequence of *increasingly complex microworlds* (called *ICM*) (Burton et al., 1984), and several systems based on ICM have been developed (Fischer, 1988; Towne et al., 1993; White and Frederiksen, 1993; White and Frederiksen, 1990). In these systems, however, the sequences are fixed (i.e., the microworlds and the relations between them aren't explicitly indexed) and they can't be adaptively changed. On the other hand, Hirashima et al. proposed a framework for indexing problems in mechanics based on the models and situations they deal with in order to sequence them adaptively (Hirashima et al., 1994; Hirashima et al., 1993). However, its ability in model-based inference is limited (especially, as it doesn't cover behavioral differences between models), and the method for assisting authors in indexing problems isn't given.

The framework of GMW and the method proposed in this paper for describing GMW present a solution to the problems these current systems have.

Assistance Provided by GMW

GMW (Horiguchi and Hirashima, 2005) consists of a set of microworlds each of which has a model which embodies some specific law(s)/principle(s) in the domain. Each microworld is indexed not only with the law(s)/principle(s) but also with the *situation* in which the model is valid. A situation is the system's structure and its state assumed in a model. It is represented by a set of *modeling assumptions*, which are the descriptions of the viewpoint in modeling the system, the behavioral range of the system to be considered and the boundary conditions of the system. Modeling assumptions represent the conditions concerned with the system's structure and its state on which the model is valid.

By indexing each microworld not only with the law(s)/principle(s) but also with the situation, the *educationally meaningful* transition between microworlds can be designed. Suppose that after learning some law(s)/principle(s) with a model of a situation, a student is ready to learn the next law(s)/principle(s). It isn't desirable for him to learn the next law(s)/principle(s) with a model which embodies it/them but the situation of which is completely different from the previous one. From an educational viewpoint, it is more desirable that he finds the necessity of the *evolution* of the previous model in the new situation, and consequently gets the new model which embodies the next the law(s)/principle(s). By describing modeling assumptions explicitly, it becomes possible to judge whether two microworlds are in such relation (called *educationally meaningful* relation).

[Example-1] An example of GMW for a physical system is shown in Figure 1. It consists of 5 microworlds, each of which is indexed with the following items:

(m1) the physical system and a model of it

(m2) the physical structure of the system: the physical objects and their relations, their attributes, and the physical processes to be considered in the model

(m3) the behavioral range of the system to be considered

(m4) the boundary conditions of the system

(m5) the skills necessary for the model-based problem solving (e.g., how to solve differential equations)

(m6) the tasks to be performed for understanding the model

MW-1 deals with a piece of domain theory *linear uniform motion* as its learning item, and has a model which embodies it. MW-2 deals with *linear accelerated motion* and *frictional force* as its learning items, and has a model which embodies them. As for MW-3, *heat generation* and *melt of the ice* are added to those of MW-2, and MW-3 has a model which embodies all of them. MW-4 and MW-5 deal with *elastic collision* and *inelastic collision* respectively, and have models which embody them respectively. In addition, *parameter-change rules* are attached to the edges between MW-1 and MW-2, between MW-2 and MW-3 and between MW-4 and MW-5, which relate the difference between the situations (i.e., modeling assumptions) of two microworlds to the difference between the behaviors of their models (only the first rule is shown in Figure 1).

For example, when a student learned the learning item in MW-1, MW-2 and MW-4 are identified as the candidates he should learn next by examining the nodes adjacent to MW-1

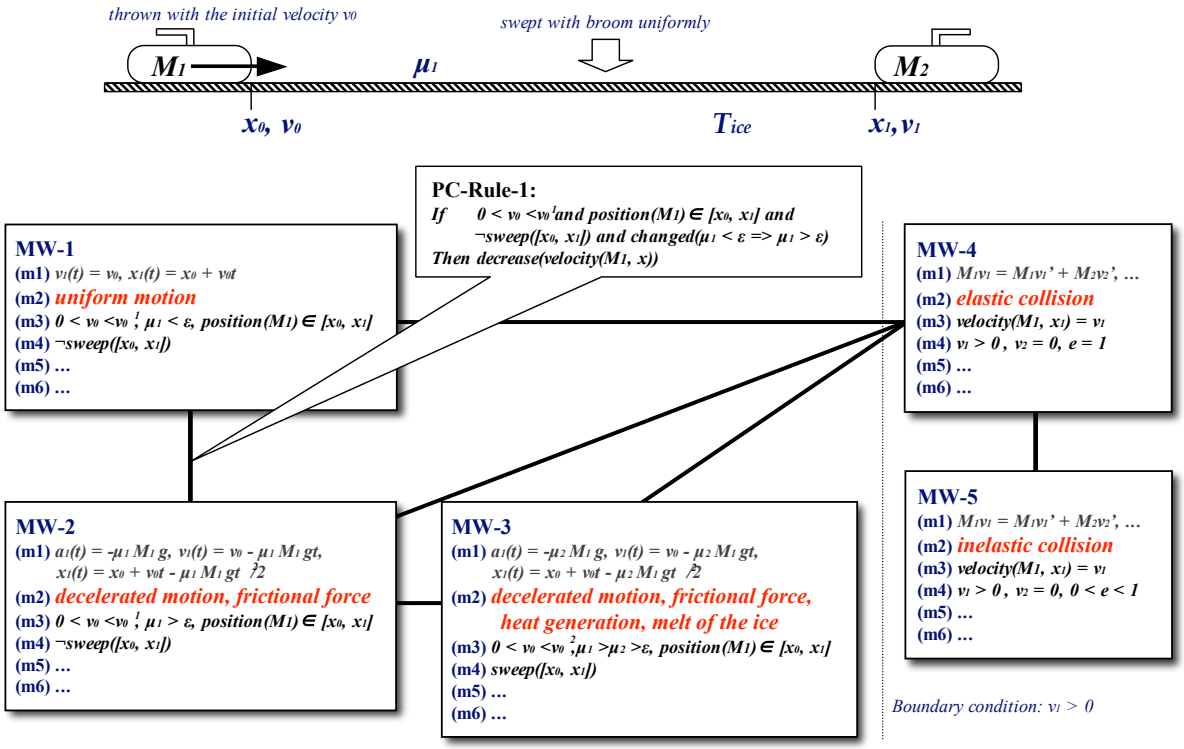


Figure 1: An example of Graph of Microworlds

in GMW. In addition, in order to assist a student in transferring from MW-1 to MW-2, it is possible to generate a task by using the parameter change rule attached to the edge between them, such as: *derive the velocity of* M_1 *when the value of* μ_1 *becomes greater and the friction becomes not negligible*. It is also possible to generate the explanation of how/why the velocity of M_1 changes. In this task, the necessity of the model of MW-2 is strongly suggested because the difference between the velocities of M_1 before/after the change of μ_1 can't be explained only by the model of MW-1. Such a task which needs the transition to another microworld to be performed is called *inter-mw-task*, while a task which can be performed with only the model of the microworld it belongs to is called *intra-mw-task*.

Method for Assisting Authors in Describing GMW

Definition of the Problem

In science education, there are a set of key concepts and laws/principles which compose the domain theory and must be learned in order for a student to understand the theory. We call them *learning items*. The learning items usually have the relations of *prerequisite*, *whole/part*, and others between them (e.g., *acceleration* should be learned before *Newton's 2nd law*, and *linear uniform motion* is the specific case of *linear accelerated motion*). In other words, they have a partial ordering. The lessons in school and the chapters in textbooks are sequenced according to the ordering. Therefore, we suppose a *learning item network* is given which consists of a partially ordered set of learning items each of which deals with some specific law(s)/principle(s) in the domain. An author is required to describe a GMW which satisfies the following requisites (see the upper part of Figure 2):

(1) The set of microworlds in the GMW has the same partial ordering as the learning item network (i.e., they are isomorphic), and each microworld has a model which embodies the law(s)/principle(s) dealt with by its corresponding learning item [1].

(2) Each microworld is indexed by the law(s)/principle(s) it deals with, the model which embodies the law(s)/principle(s) and its modeling assumptions (i.e., (m1)-(m4) in section 3.1). (The relations between two microworlds are indexed by the difference between these.)

(3) Two microworlds which correspond to two adjacent learning items in the learning item network have the *educationally meaningful* relation as much as possible.

As for these requisites, there are the following difficulties:

(1) It is, in general, difficult to find the situation (i.e., the system's structure and its state) which embodies the given law(s)/principle(s) because its search space becomes vast.

[1] A model in a microworld may include the law(s)/principle(s) dealt with by the learning item(s) which is/are upper than the microworld's corresponding learning item in the learning item network.

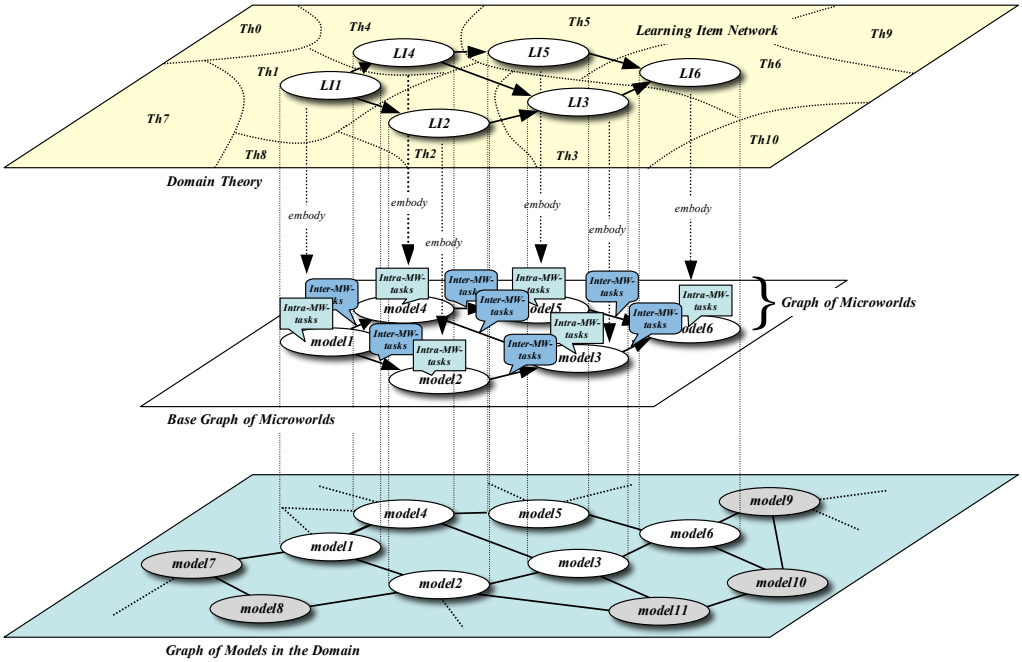


Figure 2: The structure of Graph of Microworlds

(2) It needs the expertise in model generation process in the domain to index the models in microworlds with the law(s)/principle(s) behind them and their modeling assumptions, especially because modeling assumptions are usually implicit information in models.

(3) Because of the same reason as above, it is difficult to identify the relation between two microworlds based on the differences of their models and modeling assumptions and to judge whether the transition between them is educationally meaningful. Moreover, it is also difficult to make a set of microworlds which includes the educationally meaningful relations following the partial ordering in the learning item network.

Method for Assistance

Consider the network which consists of the *possible* models (i.e., consistent combinations of modeling assumptions) and the *possible* relations between them (i.e., consistent perturbations of modeling assumptions) in the domain. We call it *Graph of Models (GoM)* [2] [3] . We suppose that the consistency of models and relations between them can be judged based on the domain theory.

In this paper, we propose a method for assisting an author in describing GMW by a *generation-test* method, in which he semi-automatically generates the models which belong to the GoM in the domain one after another, and judges whether each of them is appropriate to the GMW from an educational viewpoint. That is, we suppose that GMW can be described by extracting a subgraph from the GoM from an educational viewpoint (called *base GMW*), and by adding intra-/inter-MW-tasks (including the tools necessary for performing them) to it (see the lower part of Figure 2). The method we propose here is for the former (i.e., describing base GMW), not for the latter. Hereafter, we call base GMW simply *GMW*.

In order to implement the above method, we need a mechanism of model generation which guarantees that the combination of modeling assumptions in a model generated by it is consistent, and that the difference of modeling assumptions in a relation between two models is consistent, based on the domain theory. That is, the target of this method is a teacher who describes GMW as the teaching material in his class, not a programmer who describes domain theory for the mechanism of model generation. We suppose that domain theory is appropriately described by a programmer according to the guide in organizing it (presented in section 5).

In our method, *compositional modeling* technique (Falkenhainer and Forbus, 1991; Levy et al., 1997) is used as such a mechanism, which generates the models in the domain based on explicit modeling assumptions. In composi-

[2]Possible models/relations mean they can theoretically exist but need to be described in order to exist concretely. Because the number of possible models/relations in the domain is vast, an author can't describe the whole GoM. However, he needn't do so because what he should describe here is a set of possible models/relations which are educationally useful and cover the given learning item network (i.e., a GMW).

[3]The GoM proposed by Addanki et al. (Addanki et al., 1991) deals with only the *general/specific* relation between models, while we extend it for dealing with other types of relations (see section 5).

```
(defModel (kinetic-friction ?obj ?flr)
        :Individuals ((?obj :conditions (m-object ?obj))
                      (?flr :conditions (m-floor ?flr)))
        :Assumptions ((CONSIDER (k-cof ?obj ?flr)))
        :Conditions ((on ?obj ?flr)
                     (gt (mag (normal-force ?obj ?flr)) 0)
                     (gt (mag (velocity-t ?obj ?flr)) 0))
        :Relations ((Quantity (k-friction ?obj) :p-vector)
                    (= (mag (k-friction ?obj))
                       (* (k-cof ?obj ?flr) (mag (normal-force ?obj ?flr))))
                    (= (dir (k-friction ?obj))
                       (rev (dir (velocity-t ?obj ?flr))))))
```

Figure 3: An example of model fragment

tional modeling, domain theory is described as a set of primitives called *model fragments* each of which stands for a specific law/principle in the domain (called a *library* of model fragments) . Each model fragment consists of two parts: One is a partial situation (i.e., a partial system's structure and its state) to which the law/principle can be applied. This is described as a set of modeling assumptions. The other is a set of constraints which becomes valid when such a partial situation does exist. When a situation (i.e., a system's structure and its state) is inputted into the mechanism, a set of model fragments each of which matches the modeling assumptions which are true in the situation are instantiated, and the set of constraints given by these model fragments are outputted as the model of the situation. (Since this technique mainly targets physical systems, our method also does so.) An example of model fragment used in our implementation is shown in Figure 3.

By introducing compositional modeling mechanism, it is automated to index the models of a given situations with their modeling assumptions. It is, however, still difficult to find the situations which embody the given laws/principles. In our method, therefore, the author describes GMW as follows:

(1) First, suppose the author can find a situation which embodies the law(s)/principle(s) dealt with by a learning item in the given learning item network. The compositional modeler automatically generates the model and indexes it by its modeling assumptions.

(2) Then, he perturbs this situation by changing some parameter(s) of the system [4] . The compositional modeler automatically generates the model of this new situation and indexes it by its modeling assumptions [5] .

(3) If the new model embodies the law(s)/principle(s) dealt with by another learning item which is adjacent to the former learning item in the learning item network, he decides whether it is added to the GMW or not. If he judges that the difference between these two models is educationally meaningful, he adds the new one and the new edge between them to the GMW.

(4) By repeating (2) and (3) to grow the GMW, the author would finally get the whole GMW which embodies all the learning items in the given learning item network by a set of microworlds (and the set of microworlds has the same partial ordering as the learning item network).

By this procedure, the difficulties indicated in section 4.1 are solved except the following two points: One is to find the initial situation and its model from which the GMW is grown. The other is to identify the relation between two models based on the perturbation of situation (i.e., the difference of modeling assumptions) and to judge whether it is educationally meaningful or not. As for the former, however, it is sufficient to find only one situation and its model which embody an arbitrary learning item in the learning item network, which would be much easier than find a set of microworlds covering all the learning items. As for the latter, the function is desirable which advises the author on what physical meaning a difference of modeling assumptions has. In order to design such a function, it is necessary to classify modeling assumptions based on their physical meanings. The classification means organizing the library of model fragments based on the modeling assumptions included in each model fragments.

In the next section, therefore, we consider what types of modeling assumptions are used in generating models of physical systems and classify the modeling assumptions. Based on this classification, we also present how to organize a library of model fragments in the domain of physics. Though previous researches have presented several ways of classifying modeling assumptions and organizing a library of model fragments, we try the reclassification of modeling assumptions especially from the viewpoint of difference of models caused by the perturbation of situation. We finally describe the design of the function for inferring the relation between two models based on the difference of their modeling assumptions.

[4] A mechanism is necessary which infers what change of parameter(s) causes what change of a situation (i.e., change of modeling assumption(s)). In this paper, we omit the description of this mechanism on account of limited space.

[5] The consistency of the model of the new situation (i.e., the new combination of modeling assumptions) is guaranteed by the compositional modeler (inconsistent ones are detected and deleted by it).

Relations between Models based on the Difference of Modeling Assumptions

Modeling Assumptions

We classify the modeling assumptions made in generating models of physical systems into *constraints of physical structure (CPS)* and *constraints of operating range (COR)*. In a model or model fragment, at least one of these respective assumptions must be specified.

Constraint of physical structure (CPS) is the assumption which specifies what kind of objects, relations and their attributes in a physical system are considered (where, the specification about objects is called *constraint of physical objects (CPO)*, and the one about relations and attributes is called *constraint of physical attributes (CPA)*). CPS represents the decisions about perspectives and granularity in modeling a physical system. For example, the specification about whether two connected metal blocks are considered as one object or two objects is a CPO. The specification about whether their mechanical relations/attributes (e.g., mass, applied forces) or their electrical ones (e.g., current, resistance) are considered is a CPA.

Physical phenomena occur assuming a physical system is in a specific state. When the state changes, the model may become invalid. Therefore, a model must have the specification about the range (in its state space) within which it is valid. It is called *constraint of operating range (COR)* (where, the one which can be specified by (a set of) physical attributes is called *constraint of physical range (CPR)*, and the one which need to be specified by (a set of) conceptual attributes (e.g., complex shape of an object, complex positional relation between objects) is called *constraint of conceptual range (CCR)*). For example, since a model of two connected blocks' motion with the internal force between them assumes their velocities are the same, such specification is necessary. A model of a resistance assuming its value is constant needs the specification that its current and voltage are within the proportional range. These are CPRs. In a model of a block b descending an inclined plane p by gravity from the gravitational field g, their positional relation must be appropriately specified (e.g., $in(b, g)$, $on(b, p)$). This is a CCR.

In each type of these modeling assumptions, there are often the sets of exclusive ones which can't be made simultaneously. For example, in a physical system which has a CPS, it isn't allowed to make assumptions *transient state* and *steady state* simultaneously as COR. In a physical system which has a CPO, it isn't allowed to make assumptions *consider friction between two blocks* and *not consider friction between them* simultaneously as CPA. Moreover, in a physical system, it isn't allowed to make assumptions *view a block as a rigid object* and *view it as an aggregation of atoms/molecules* as CPO.

Relations between Models

When the domain theory is described as a library of model fragments, each model fragment stands for a specific physical law/principle. A set of CPS and COR is attached to each model fragment as its condition of application. The model of a situation (i.e., the system's structure and its state) is generated as a conjunction of the constraints given by the instantiated model fragments. Its modeling assumption is the conjunction of the ones attached to each model fragments (the consistency of the conjunctions is guaranteed by the compositional modeling mechanism).

By grouping the model fragments each of which has exclusive modeling assumption(s), it is possible to design the function for suggesting the relation between the models in two microworlds before and after the perturbation of situation. That is, first, the two sets of model fragments are compared, each of which composes each model. Then, if a pair of model fragments each of which belongs to each model and matches the same/similar partial situation has exclusive modeling assumption(s), the relation between the models is inferred from the type of the assumption. The procedure is as follows:

(0) Assume that it is possible for each model fragment in one model to find its corresponding model fragment in another model. Two model fragments corresponds to each other (called *a pair of model fragments*) if they are instantiated by matching the same/similar partial physical structure (i.e., physical objects and their relations/attributes) in the system (when two models have different CPOs, it is assumed that the method is given for finding the correspondence between the physical objects considered in them). If a model fragment in one model can't find its corresponding model fragment in another model, the following procedure is carried out based on its modeling assumptions themselves.

(1) When two model fragments which corresponds to each other have exclusively different CPOs, it is inferred that the difference of two models is *change of the viewpoint/granularity* about the partial system which they match.

(2) When two model fragments which corresponds to each other have the same CPOs and exclusively different CORs, it is inferred that the difference of two models is *change of the operating range* about the partial system which they match.

(3) When two model fragments which corresponds to each other have the same CPOs, the same CORs and exclusively different CPAs (i.e., it isn't possible to find the correspondence between the relations/attributes considered in them), it is inferred that the difference of two models is *general/specific* about the partial system which they match [6] .

The differences inferred by the above procedure are sometimes concerned with all the pairs of model fragments which compose the models (i.e., the whole system), or concerned with a pair of model fragments (i.e., the partial system which

[6] It is assumed that there is some kind of *inclusion* relation between the CPAs of two model fragments. For example, Levy et al. defined a *simpler-than* relation based on the *superset/subset* relation between the causal orderings of (the output quantities of) two model fragments (Levy et al., 1997).

they match). In the former case, they stand for the global differences between two models, while in the latter case, they stand for only the local differences between them. In general, because there can be multiple global/local differences in two models, it is difficult to determine *the most appropriate difference* between them.

In this research, therefore, we adopt the following method: (1) the authoring system first enumerates the possible global/local differences between two models by the above procedure, then the author, referring to them, identifies the most appropriate difference and judges its educational meaning (i.e., determines whether there is the edge between them and its type). In our method, since the new model is generated by perturbing the old one's situation (i.e., modeling assumptions), it is expected that there are at most a few differences between them and that the author has little difficulty in the identification and judgement.

[Example-2] Figure 4a shows the physical system in which an object b_1 is put on an inclined plane p_1 (to which a horizontal plane p_2 is connected). Figure 4b shows a model (i.e., a set of instantiated model fragments) of a situation of this system in which b_1 remains at rest on p_1 because the tangential component of b_1's gravity on p_1 is smaller than the maximum static friction between b_1 and p_1. It (called model-1) consists of 5 model fragments, including *static friction* and *rest*. If the coefficient of static friction is decreased in this situation, another situation may occur in which b_1 moves downward accelerated by its gravity (and the kinetic friction). The model of this situation (called model-2) is shown in Figure 4c and it consists of 5 model fragments, including *kinetic friction* and *linear acc-motion*.

Because the model fragments *gravity* in model-1 and model-2 are instantiated by matching with the same physical structure in these models, they correspond to each other. As for the model fragments *normal force* and *acceleration* in both models, the matters are the same. These model fragments compose the common part of model-1 and model-2 because their CORs are also the same in both models respectively.

The model fragments *static friction* in model-1 and *kinetic friction* in model-2 correspond to each other because of the same reason. However, their CPRs which specify the range of the value of the coefficient of static friction are exclusively different. It is, therefore, inferred that there is a difference between these models in 'the change from static friction to kinetic friction because of the change in the value of the coefficient of static friction.' The model fragments *rest* in model-1 and *linear acc-motion* in model-2 also correspond to each other because of the same reason. Their CPRs which specify the range of the value of b_1's acceleration are exclusively different. It is, therefore, inferred that there is a difference between these models in 'the change from rest to linear accelerated motion because of the change in the value of b_1's acceleration.'

Referring to these two differences enumerated by the authoring system, the author identifies the most appropriate difference and judges its educational meaning.

[Example-3] In Figure 4, if the time variable of model-2 is increased, b_1 transfers from p_1 to p_2 (i.e., $on-floor(b_1, p_1)$ changes to $on-floor(b_1, p_2)$). The model of this new situation (called model-3) is shown in Figure 4d and it consists of 4 model fragments, including *linear uni-motion*.

The model fragments *gravity* and *acceleration* in model-2 and model-3 compose the common part of these models because of the same reason as Example-2.

The model fragments *kinetic friction*, *normal force* in model-2 and *normal force* in model-3 don't have their corresponding model fragments. The reason is that their CCRs which specify the positional relation among b_1, p_1 and p_2 exclusively changed because of b_1's transition from p_1 to p_2. It is, therefore, inferred that there are the differences between these models in 'the disappearance of kinetic friction and normal force between b_1 and p_1, and the appearance of normal force between b_1 and p_2 because of the change in the positional relation among b_1, p_1 and p_2.'

The model fragments *linear acc-motion* in model-2 and *linear uni-motion* in model-3 correspond to each other because they are instantiated by matching the same physical structure in these models. Their CPRs which specify the range of the value of b_1's acceleration are exclusively different. It is, therefore, inferred that there is a difference between these models in 'the change from linear accelerated motion to linear uniform motion because of the change in the value of b_1's acceleration.'

Concluding Remarks

In this paper, we proposed a method for assisting an author in describing GMW. The feature of our method is that it uses a problem solver (i.e., model generator) in the domain for indexing a set of microworlds semi-automatically. We think this is inevitable in order to assist the authors in indexing them based on their models, aiming at the ability in inference about the difference between models.

Introducing the powerful model of expertise may cause the problems of its cost and limited applicability. However, we think they can be reduced by adopting compositional modeling technique. That is, it has a framework for judging the consistency of models at a conceptual level (i.e., based on modeling assumptions), and the methods for describing domain knowledge (i.e., library of model fragments) at that level have been developed in literature. We think preparing a set of templates of model fragments in each domain would provide the guideline for describing the domain knowledge. Moreover, because this technique works in any domain of physics and provides the model generator which widely covers its domain, our method would be applicable to many domains of physics. We are planning to verify the usefulness of the prototype system which implements our method, and to discuss the method for describing the expertise in model generators.

References

Addanki, S.; Cremonini, R.; and Penberthy, J.S. 1991. Graphs of models. *Artificial Intelligence* 51: 145-177.

Burton, R.R.; Brown, J.S.; and Fischer, G. 1984. Skiing as a model

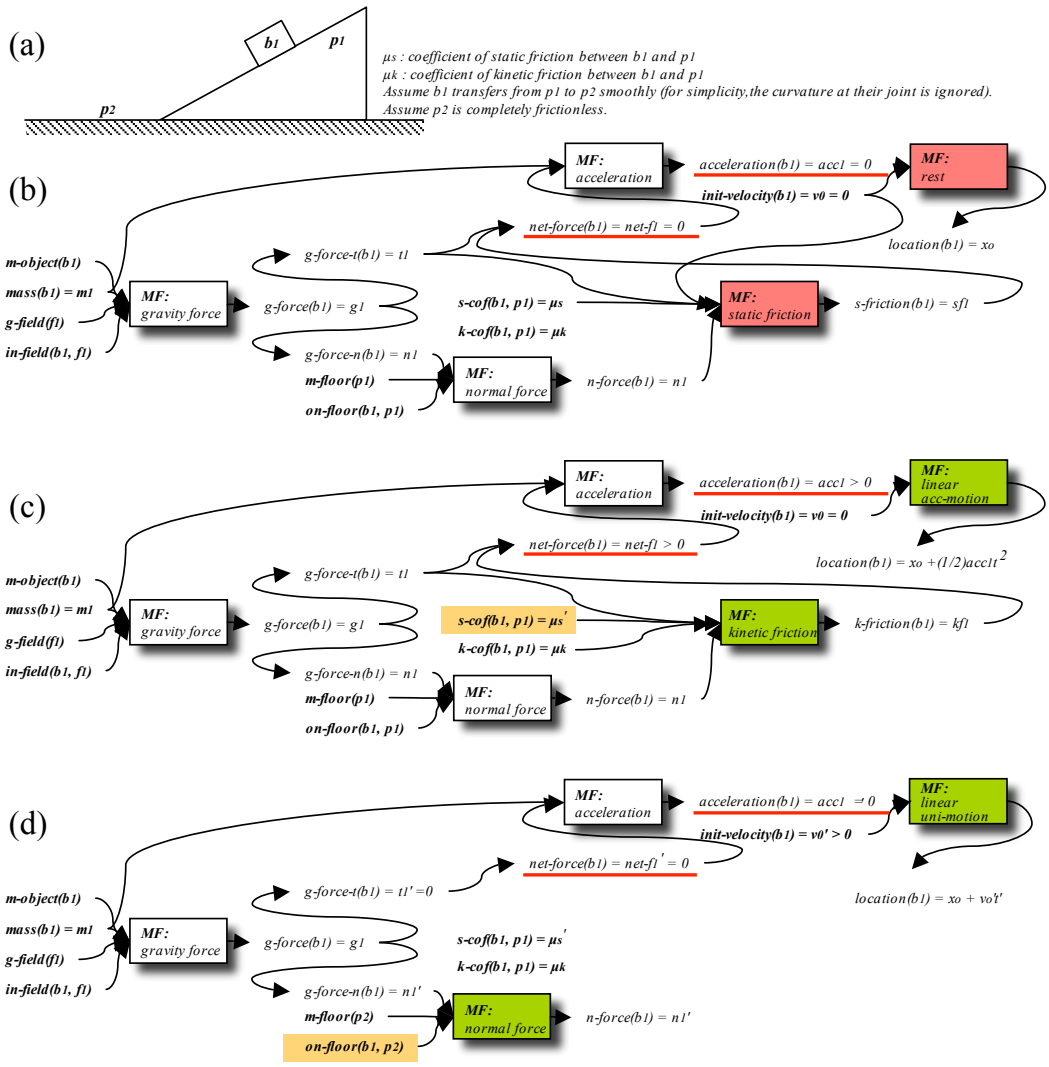


Figure 4: An example of difference between models

of instruction. In Rogoff, B.; and Lave, J. eds. *Everyday Cognition: its development in social context*. Harvard Univ.Press.

Falkenhainer, B.; and Forbus, K.D. 1991. Compositional Modeling: Finding the Right Model for the Job. *Artificial Intelligence* 51: 95-143.

Fischer, G. 1988. Enhancing incremental learning processes with knowledge-based systems. In Mandl, H.; and Lesgold, A. eds. *Learning Issues for Intelligent Tutoring Systems*. Springer-Verlag.

Hirashima, T.; Niitsu, T.; Hirose, K.; Kashihara, A.; and Toyoda, J. 1994. An Indexing Framework for Adaptive Arrangement of Mechanics Problems for ITS. *IEICE Trans. Inf. and Syst.* E77-D(1): 19-26.

Hirashima, T.; Niitsu, T.; Kashihara, A.; and Toyoda, J. 1993. An Indexing Framework for Adaptive Setting of Problem in ITS. In *Proceedings of AIED93*, 90-97.

Horiguchi, T.; and Hirashima, T. 2005. Graph of Microworlds: A Framework for Assisting Progressive Knowledge Acquisition in Simulation-based Learning Environments. In *Proceedings of AIED2005*, 670-677.

Levy, A.Y.; Iwasaki, Y.; and Fikes, R. 1997. Automated model selection for simulation based on relevance reasoning. *Artificial Intelligence* 96: 351-394.

Merrill, M.D. 1999. Instructional Transaction Theory (ITT): Instructional Design Based on Knowledge Objects. In Reigeluth, C.M. ed. *Instructional-Design Theories and Models Vol.II: A New Paradigm of Instructional Theory*, 397-424. Hillsdale, NJ: Lawrence Erlbaum Associates.

Murray, T.; Blessing, S.; and Ainsworth, S. eds. 2003. *Authoring Tools for Advanced Technology Learning Environments*. Kluwer Academic Publishers.

Towne, D.M. 1995. *Learning and Instruction in Simulation Environments*. Educational Technology Publications, Englewood Cliffs, New Jersey.

Towne, D.M.; de Jong, T.; and Spada, H. eds. 1993. *Simulation-*

Based Experiential Learning. Springer-Verlag, Berlin, Heidelberg.
Wenger, E. 1987. *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge*. Morgan Kaufmann.
White, B.; and Frederiksen, J. 1993. ThinkerTools: Causal models, conceptual change, and science education. *Cognition and Instruction* 10: 1-100.
White, B.; and Frederiksen, J. 1990 Causal model progressions as a foundation for intelligent learning environments, *Artificial Intelligence* 42: 99-157.

Qualitative simulation of nonlinear dynamical models of gene-regulatory networks

Liliana Ironi and **Luigi Panzeri**
IMATI-CNR
via Ferrata 1, Pavia (Italy)

Abstract

This paper discusses the work-in-progress of a research effort aiming at the design and implementation of a qualitative simulation algorithm of the dynamics of a specific class of ODE models of Gene-Regulatory Networks (GRN). In such models, characterized by incomplete knowledge of regulation mechanisms and kinetic parameters, regulation is assumed to be threshold-dependent, i.e. only effective above or below a certain threshold. Switch-like behaviors across variable thresholds are properly modeled by steep sigmoid functions the values of which continuously vary from zero to one around the threshold. The ODE models that result from the algebraic combination of such switch-like interaction terms describe both linear and nonlinear GRN dynamics that occur at different time-scales. Qualitative simulation of such kinds of models is a quite hard problem that requires the development of *ad hoc* tailored algorithms. Unlike GNA, that considerably simplifies the problem by approximating threshold-regulated response functions by step functions, we propose a qualitative simulation algorithm that works for continuous models, being the continuity assumption crucial in view of more and more realistic models. The algorithm is grounded on the integration of QR techniques with singular perturbation analysis methods that lay the mathematical basis for dealing with both slow and fast nonlinear dynamics.

Introduction

Due to unprecedented amount of information at genomic level made available, in recent years, by high-throughput experimental technologies, it has become increasingly clear that computational modeling and simulation frameworks are needed to represent, understand and predict the complex dynamics of Gene-Regulatory Networks (GRN). Although, up to now, there is no model that efficiently and accurately represents the gene interactions underlying regulatory mechanisms in their whole complexity, a specific class of ODEs has shown to be adequate to describe the essential features of their dynamics. These models assume that the interactions between variables are threshold-dependent, i.e. the effect of a variable on another one is regulated by a threshold value. Such an assumption is quite reasonable as switch-like behaviors across variable thresholds are well-suited to mathematically represent the effects of the transcription factors on the transcription rates of genes. Although such models allow us to provide detailed description of gene regulatory mechanisms at the molecular level (Glass & Kauffman 1973; Plahte, Mestl, & Omholt 1998), their applicability to predict their quantitative dynamics is rather limited even when the network at hand is very well studied. As a matter of fact, making predictions of the dynamics of specific networks, either from an initial state or in response to environmental stimuli, by exploiting classical numerical approaches is mostly impracticable as precise and quantitative information on (i) the biochemical reaction mechanisms underlying regulatory interactions, and (ii) kinetic parameters and threshold concentrations are currently unknown and not identifiable from data. However, at the current state of knowledge, qualitative predictions of the dynamical properties are not make-shift solutions but rather appropriate to get insight into the functioning of systems not completely understood as molecular interaction networks are.

To this end, the application of generic qualitative simulation approaches (Kuipers 1994), at least in their original form, is not the right solution. The mathematical tools they are grounded on are too much simple to compensate for the lack of complete knowledge. This results in a number of drawbacks, e.g. their inability to upscalability, the exponential growth of the generated behaviors, and the generation of spurious behaviors, that reveal to be particularly serious in predicting nonlinear dynamics of regulatory networks even in the case of networks with a small number of interacting genes. A qualitative study of GRNs dynamics could, in theory, be performed by more sophisticated analytical methods based on the classical theory of qualitative analysis of dynamical systems, and properly adapted to the specific class of models (Glass & Kauffman 1973; de Jong 2002; Plahte, Mestl, & Omholt 1998; Plahte & Kjøglum 2005). But, in practice, given the complexity of the network structures due to the large number of both components and interactions, such kind of analysis is quite hard or even unfeasible to be performed by hand. Thus, the need for the development of qualitative simulation algorithms, based on more sophisticated and adequate mathematical tools, and specifically tailored to capture the network dynamical properties that depend only on the model structure and are invariant for ranges of values of kinetic parameters.

The work, herein presented, is an effort in this direction,

and aims at providing a qualitative simulation algorithm of ODE models of GRN dynamics which works under the assumptions that (i) threshold-dependent regulation mechanisms are modeled by continuous steep sigmoid functions, and (ii) any two genes are never regulated at the same threshold by a certain variable. The sigmoidal-nonlinearities make the simulation problem quite hard to be tackled. But, the assumption that all sigmoids have very high steepness allows us to apply a systematic way of analysis. Let us observe that, due to the switch-like character of the response functions around the thresholds, the GRN dynamics occurs at different time-scales. To be able to deal with both slow and fast nonlinear dynamics we theoretically base our algorithm on a classical singular perturbation analysis method properly adapted to the assumed class of ODEs (Plahte & Kjøglum 2005; Veflingstad & Plahte 2007). Such a method suitably combined with QR key concepts computationally drives, starting from an initial state and constraints that define the parameter space domain, the construction of all possible state transitions along with the sets of symbolic inequalities on parameter values that hold when specific transitions occur.

Related work

Since frameworks for phenomenological modeling of GRNs by ODE equations have been proposed (de Jong 2002; Glass & Kauffman 1973; Plahte, Mestl, & Omholt 1998), a rather significant number of efforts in developing analytical methods for their qualitative study has been made (Glass 1977; Hasty *et al.* 2001; Gouzè & Sari 2003; Plahte & Kjøglum 2005). But, due to the difficulty to perform by hand such kind of analysis, the actual application of these methods has been restricted to toy-examples of scarce biological interest. Pioneering work towards automated qualitative analysis and simulation of GRNs results in a computational tool, called GNA (de Jong *et al.* 2004). GNA circumvents the hard problem of dealing with sigmoidal nonlinear response functions by approximating them with step functions, discontinuous in the threshold hyperplanes. Such an assumption considerably simplifies the analysis as the model results in piecewise-linear equations, but it raises the problem to find a proper continuous solution across the threshold hyperplanes, or, in other words, to seek for generalized solutions of ODEs with discontinuous right-side terms. But, the solution to this problem is not straightforward as (i) there exists in the literature several definitions of generalized solutions, (ii) it is not yet completely understood what are the relationships between different definitions, and then, (iii) it is not clear how to choose the "right" definition for a particular task (Bacciotti 2003). GNA adopts the Filippov approach that results particularly popular and convenient to deal with control problems but it may present drawbacks when applied to approximate the limit solutions of a continuous ODE model: it might find "too many" solutions, and fail to reach all stable solutions. As a consequence, GNA suffers from the same disadvantages that together with a further approximation introduced in the algorithm for computational problems might compromise its soundness and completeness (Dordan, Ironi, & Panzeri).

Therefore, the algorithm we propose aims at both overcoming the limits of GNA and providing a framework that, thanks to the continuity assumption, can be gradually extended to tackle wider and more and more realistic classes of models.

Theoretical background

Singular perturbation analysis: basic ideas

Singular perturbation analysis is a classical approach to study phenomena that occur at different time-scales (Holmes 1995). The dynamics of such phenomena are described by ODEs in which a small parameter multiplies either one of the derivatives or higher order derivative, that is by system equations of the form:

$$\begin{aligned} \epsilon\dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x},\mathbf{y},\epsilon) \\ \dot{\mathbf{y}} &= \mathbf{g}(\mathbf{x},\mathbf{y},\epsilon) \end{aligned} \tag{1}$$

where the dot denotes differentiation with respect to the ordinary time t, $\mathbf{x}(t) \in R^m, \mathbf{y}(t) \in R^l$, $0 < \epsilon \ll 1$, and $\mathbf{f}, \mathbf{g}$ smooth functions of $\mathbf{x}, \mathbf{y}, t$.

Let us indicate Eq. (1) associated with appropriate initial conditions by $\mathcal{M}_\epsilon$, and the same initial value problem where $\epsilon = 0$ by $\mathcal{M}_0$. The system modeled by $\mathcal{M}_\epsilon$, called *full system*, is singularly perturbed if, as $\epsilon \to 0$, the solution of $\mathcal{M}_\epsilon$ identifies a "small" region, called *boundary-layer region*, of non-uniform convergence to the solution of the *reduced system* $\mathcal{M}_0$. The region of uniform-converge of $\mathcal{M}_\epsilon$ to $\mathcal{M}_0$ is called *outer region*.

Singular perturbation methods aim at calculating an approximate solution of $\mathcal{M}_\epsilon$ for $0 < \epsilon \ll 1$, and differ from each other for the way they calculate and combine the boundary-layer solution and outer solution. In outline, the fundamental idea underlying these methods is to calculate local solutions in both boundary-layer and outer region, and combine them to find the global approximate solution. The fast dynamics in the boundary-layer is studied by suitably scaling the time variable, namely $\tau = t/\epsilon$. Then, the full initial value problem turns into the boundary-layer system:

$$\begin{aligned} \mathbf{x}' &= \mathbf{f}(\mathbf{x},\mathbf{y},\epsilon) \\ \mathbf{y}' &= \epsilon\, \mathbf{g}(\mathbf{x},\mathbf{y},\epsilon) \end{aligned} \tag{2}$$

where the prime denotes the derivative with respect to τ. In the limit, the fast dynamics is obtained by solving:

$$\begin{aligned} \mathbf{x}' &= \mathbf{f}(\mathbf{x},\mathbf{y},0) \\ \mathbf{y}' &= 0 \end{aligned} \tag{3}$$

This system has a manifold of stationary points given by $\mathbf{f}(\mathbf{x},\mathbf{y},0) = 0$, called slow-manifold. The reduced system $\mathcal{M}_0$:

$$\begin{aligned} 0 &= \mathbf{f}(\mathbf{x},\mathbf{y},0) \\ \dot{\mathbf{y}} &= \mathbf{g}(\mathbf{x},\mathbf{y},0) \end{aligned} \tag{4}$$

describes the motion in the original time t along those points in the slow-manifold, $\mathbf{x} = \mathbf{x}(\mathbf{y})$, that satisfy suitable hypotheses among those stability (Tikhonov-Levinson theorem, 1952). Then, the outer solution is described by the equation:

$$\dot{\mathbf{y}} = \mathbf{g}(\mathbf{x}(\mathbf{y}),\mathbf{y},0) \tag{5}$$

Taken together, the reduced equation and the boundary-layer solution approximate the solution of $\mathcal{M}_\epsilon$ for small nonzero values of ϵ.

A computational framework for the analysis of GRN dynamics

Experimental and theoretical studies seem to confirm the adequacy of the following specific class of ODEs to describe the essential features of a wide range of regulatory systems, and, in particular, of the complex dynamics of GRNs:

$$\dot{x_i} = f_i(\mathbf{Z}) - \gamma_i x_i \quad i = 1, \ldots, n \tag{6}$$

where the dot denotes time derivative, x_i is the concentration of the i-th gene product, $\gamma_i > 0$ is the decay rate of x_i, $\mathbf{Z}$ is a vector with Z_{jk} as components, and $Z_{jk} = S(x_j, \theta_{jk}, q)$ is a sigmoid function with threshold θ_{jk}.The response, or regulatory, function $S : \mathrm{R}^+ \to [0, 1]$ is a continuous monotonic S–shaped map depending on the parameter q $(0 < q \ll 1)$, that determines the steepness of S around the threshold value θ_{jk}, such that for $q \to 0$ we have $S(x_j, \theta_{jk}, q) = 0$ (respectively 1) when the value of x_j is smaller (greater) than θ_{jk}.

Each x_i, defined in $\Omega_i \subset \mathrm{R}^+$, is associated with n_i thresholds ordered according to $\theta_{ij} < \theta_{ik}$ if $j < k$. The state equations describe the balance between the production process $f_i(\mathbf{Z})$ and the degradation one, herein supposed to be linear. The functions f_i are multilinear polynomials in the variables Z_{jk}, and are frequently composed by algebraic equivalents of Boolean functions. More precisely,

$$f_i(\mathbf{Z}) = \sum_{l=1}^{L_i} \kappa_{il} \prod_{\substack{j=1,n \\ k=1,n_j}} Z_{jk}^{\alpha_{jkl}} \tag{7}$$

where κ_{il} are real values that denote kinetic rate parameters, L_i is the possibly empty number of interactions that synthesize x_i, and, in accordance with the network structure, α_{jkl} assumes value either equal to 1 when Z_{jk} takes part in the l-th interaction or equal to 0 otherwise.

In the present paper we adopt a further assumption that sounds quite realistic:

Assumption $\mathcal{A}$. Every gene product only regulates one gene at each of its thresholds.

Mathematically, this assumption implies that each Z_{jk} only occurs in one equation. This simplifies the calculation of the slow–motion manifold that, otherwise, generally consists in an heavy, nonlinear computational problem.

To exemplify the concepts and the definitions as they are introduced, all through the paper we will consider the ODE model:

$$\begin{aligned} \dot{x_1} &= \kappa_{11}(1 - Z_{11})(1 - Z_{22}) + \kappa_{12}(1 - Z_{21}) - \gamma_1 x_1 \\ \dot{x_2} &= \kappa_{21}(1 - Z_{12}) - \gamma_2 x_2 \end{aligned} \tag{8}$$

where all parameters are strictly positive, and, the response function Z_{jk} is expressed by the standard Hill function $S(x_j, \theta_{jk}, q) = \frac{x_j^{1/q}}{x_j^{1/q} + \theta_{jk}^{1/q}}$, commonly used in the literature.

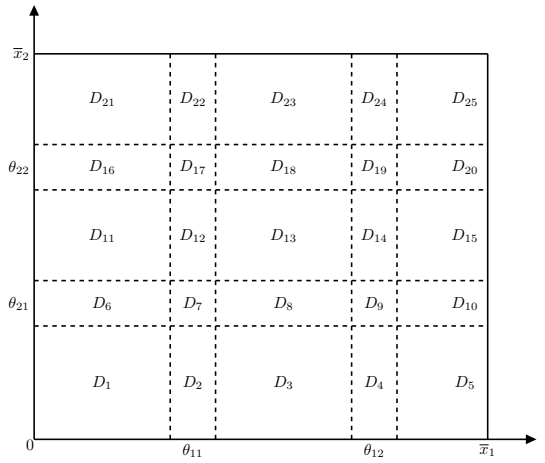


Figure 1: Partition of the phase-plane into regular and switching domains.

A - Regular and Switching Domains. Let us consider the n-dimensional vector of the state variables $\mathbf{x}$ whose domain $\Omega = \Omega_1 \times \ldots \times \Omega_n$ is given by the product of the domains $\Omega_i \subset \mathrm{R}^+$ of each of its component. The ordered set Θ_i of the n_i threshold values θ_{ij} associated with each x_i naturally induces a partition of Ω_i into qualitatively distinct domains. As a matter of fact, instead of the sharp value θ_{ij} we must consider a range of values around it, whose width, $\delta > 0$, is a monotonic function of the steepness parameter q with $\delta(q) \to 0$ for $q \to 0$, and characterizes the domain where the related response function takes values other than zero or one. Let us denote by $\underline{\theta}_{ij}$ and $\overline{\theta}_{ij}$ the values $\theta_{ij} - \delta/2$ and $\theta_{ij} + \delta/2$, respectively. Then, each Ω_i results from the product of open, $(\overline{\theta}_{ik}, \underline{\theta}_{i(k+1)})$, and closed, $[\underline{\theta}_{i(k+1)}, \overline{\theta}_{i(k+1)}]$, intervals $\mathcal{I}_{ik}$[1]. The whole system domain Ω is then partitioned, as showed in Fig. 1, into hyper-rectangles $D = \mathcal{I}_{1l_1} \times \ldots \mathcal{I}_{il_i} \ldots \times \mathcal{I}_{nl_n}, \quad l_i \in \{0, \ldots, n_{i+1}\}$.

In the set Δ of all the domains identified by the partition, we can distinguish the set $\Delta_s \subset \Delta$ of *switching domains* (SD) from the set $\Delta_r \subset \Delta$ of *regular domains* (RD), such that $\Delta = \Delta_s \cup \Delta_r$.

A domain D belongs to Δ_s if one, several, or all variables are at (one of) their thresholds or, equivalently, if it results from the product of at least one closed interval, e.g. D_6, D_{14}, D_{17}. Let $\sigma(D)$, $D \in \Delta_s$, be the switching order of D, i.e.the number of switching state variables, then those ones that assume values in a closed interval in D. A SD with $\sigma = n$ is called a *center*. In the example, the centers are D_7, D_9, D_{17}, and D_{19}.

A domain D belongs to Δ_r if it is an open set, e.g. D_1, D_5, D_{13}, and it is also called *box*.

The network dynamics in each domain $D \in \Delta$ is described by different models: the slow motion in $D_r \in \Delta_r$ is described by linear ODEs whereas the fast motion in $D_s \in \Delta_s$, or equivalently around the thresholds, is described

[1] $k = 0, \ldots, n_i$, where θ_{i0} and θ_{n_i+1} denote 0 and $\overline{x}_i = \max(x_i)$, respectively.

by nonlinear equations. Thus, the need to adopt different analysis strategies of the motion in regular and switching domains.

B - Motion equations in regular domains. In each box D_r, Z_{jk} equals either 0 or 1 in the step function limit. This simplifies Eq. (6) as they reduces to linear equations:

$$\dot{x_i} = \mu_i - \gamma_i x_i, \quad i = 1, \ldots, n \tag{9}$$

where μ_i depends on D_r, and is given by the sum of some κ_{il}. From Eq. (9) we can easily find the *focal point* $\mathbf{x}^* = \{x_j^* = \frac{\mu_j}{\gamma_j}\}$ the trajectories are heading towards. Herein, we assume that focal points do not belong to switching domains. If $\mathbf{x}^*$ belongs to the initial domain D_r, there is a stable point in it, called Regular Stable Point (RSP). Otherwise, the trajectories move to a switching domain adjacent to D_r.

For example, the motion equations (8) in the domain D_{11}, being $Z_{21} = 1$ and $Z_{11} = Z_{12} = Z_{22} = 0$, reduce to:

$$\begin{aligned} \dot{x_1} &= \kappa_{11} - \gamma_1 x_1 \\ \dot{x_2} &= \kappa_{21} - \gamma_2 x_2 \end{aligned} \tag{10}$$

whose trajectories move towards the focal point $\mathbf{x}^* = (\frac{\kappa_{11}}{\gamma_1}, \frac{\kappa_{21}}{\gamma_2})$. If $\mathbf{x}^* \in D_{13}$ the trajectories starting in D_{11} point to $\mathbf{x}^*$. Thus, they escape from D_{11}, and heading towards D_{13}, they first move to D_{12}.

C - Motion equations in switching domains. In a switching domain D_s we distinguish $\sigma(D_s)$ switching variables, $x_s \in [\underline{\theta_s}, \overline{\theta_s}]$, from $n - \sigma(D_s)$ regular ones x_r. For example, in the domain D_{12} ($\sigma(D_{12}) = 1$), x_1 is the switching variable while x_2 is the regular one.

Using singular perturbation analysis as properly adapted to study the system (6) (Plahte & Kjøglum 2005), we can capture the salient features of the nonlinear dynamics in a switching domain D_s, and determine how the trajectories cross it to move towards other domains. In outline, (i) Eq. (6) related to the x_s variables are rewritten into the form (1) through a change of coordinate system, (ii) the boundary-layer and outer solutions are calculated in the new coordinates, and (iii) they are converted back into the usual frame of reference.

Let $\Sigma : \Omega \mapsto [0,1]^n$ be the coordinate transformation that converts the x_s coordinates into the Z_s ones. As under our assumptions, $\frac{\partial Z_s}{\partial x_s} = \frac{1}{q} d_s(Z_s, x_s)$, where d_s is a continuous and limited function, we can write the full system:

$$\begin{aligned} q\dot{Z}_s &= d_s(Z_s, x_s)(f_s(\mathbf{Z_R}, \mathbf{Z_S}) - \gamma_s x_s) \\ \dot{x}_r &= f_r(\mathbf{Z_R}, \mathbf{Z_S}) - \gamma_r x_r \end{aligned} \tag{11}$$

where $\mathbf{Z_S}, \mathbf{Z_R}$ are the vectors of switching and regular variables Z_s and Z_r, respectively. For $q \to 0$, Eq. (11) are of the form (1), and then we study the fast dynamics in the boundary-layer in the scaled time variable $\tau = \frac{t}{q}$:

$$\begin{aligned} Z_s' &= d_s(Z_s, \theta_s)(f_s(\mathbf{Z_R}, \mathbf{Z_S}) - \gamma_s \theta_s) \\ x_r' &= 0. \end{aligned} \tag{12}$$

The solution of the system (12) associated with appropriate initial conditions gives us the boundary-layer solution. As $\mathbf{Z_R}$ is constant in any $D_s \in \Delta_s$, we focus on the switching variables Z_s only, and calculate the slow-manifold of the system (12) that is the set of solutions, for all s, of the stationary equations $Z_s' = 0$. We call *exit point set* (*EP*) the set of stable solutions satisfying the conditions of the Tikhonov-Levinson theorem, and we call *Z-cube* $\mathcal{Z}(D_s) = [0,1]^{\sigma(D_s)}$ the frame of reference where we search for an exit point. Then, under the hypothesis that at least one exit point $\tilde{\mathbf{Z}}_\mathbf{S}$ exists, the reduced equations are obtained by substituting it in the motion equations of regular variables:

$$\dot{x}_r = f_r(\tilde{\mathbf{Z}}_\mathbf{S}, \mathbf{Z_R}) - \gamma_r x_r. \tag{13}$$

The problem (13) is linear, and then, given the initial conditions, the outer solution, that determines how the trajectories move along the x_r directions, is easily calculated.

Remark 1. The location of each exit point is crucial in our analysis as it indicates the next adjacent domains the trajectories are moving towards along the x_s directions.

Let $A(D_s)$ be the set of domains adjacent to D_s, and $\mathcal{D}_s = D_s \cup A(D_s)$. In the limit $q \to 0$, we define a map $\Sigma_{D_s} : \mathcal{D}_s \mapsto \mathcal{Z}(D_s)$ such that the interior of $\mathcal{Z}(D_s)$, $int(\mathcal{Z}(D_s))$, and its boundary are the images of D_s, and $A(D_s)$, respectively. More precisely, the domains $D_k \in A(D_s)$ are mapped into the faces of $\mathcal{Z}(D_s)$ when $D_k \in \Delta_s$ or into its vertices, otherwise. If an exit point exists in the interior of $\mathcal{Z}(D_s)$, and the associated reduced system has a critical point inside D_s then it exists a stable point in D_s, also called Singular Stable Point (SSP).

As an example let us consider the boundary-layer system in D_{12}:

$$\begin{aligned} Z_{11}' &= \frac{Z_{11}(1 - Z_{11})}{\theta_{11}}(\kappa_{11}(1 - Z_{11}) - \gamma_1 \theta_{11}) \\ x_2' &= 0 \end{aligned} \tag{14}$$

Its candidate exit point set *EP* = $\{0, 1, 1 - \frac{\gamma_1 \theta_{11}}{\kappa_{11}}\}$ includes the vertices, and a point in the interior of $\mathcal{Z}(D_{12})$, being $\mathcal{Z}(D_{12})$ the segment $[0,1]$, whose endpoints 0, 1, and its interior are the images of D_{11}, D_{13}, and D_{12}, respectively. Then, D_{11} and D_{13} are possible next traversed domains and D_{12} may contain a stable point.

D - Search for exit points. Let us observe that stationary points always exist on the vertices of $\mathcal{Z}(D_s)$. Then, for a vertex to be an exit point it should fulfill the stability condition. The computational cost of the search for all the other exit points could be quite heavy, but it can be considerably reduced by checking first a necessary condition for the existence of a stationary point on the other elements of $\mathcal{Z}(D_s)$. Let F be a face or the interior of $\mathcal{Z}(D_s)$. In (Veflingstad & Plahte 2007), it has been proved that necessary condition for the existence of a stationary point in F is that the Jacobian matrix $\mathbf{J}_F = (\frac{\partial f_i}{\partial Z_j})$ restricted to the switching variables in F has a complete loop. This holds if and only if there is a non-zero loop involving all variables in $\mathbf{J}_F$, and it can be checked by using concepts from graph theory.

Let be $\tilde{F}$ any elements of $\mathcal{Z}(D_s)$, face, vertex, or interior, where a stationary point $\tilde{\mathbf{Z}}$ is located, and $\mathcal{L}_{\tilde{F}} = \{l : l \in \{1, \ldots, \sigma(D_s)\}, \quad \tilde{Z}_l \in \{0, 1\}\}$. $\tilde{\mathbf{Z}}$ is an exit point if: (i) it is stable, and (ii), if $\tilde{F}$ is on the boundary of $\mathcal{Z}(D_s)$, $Z_l, \forall l \in \mathcal{L}_{\tilde{F}}$ has to head towards $\tilde{F}$. The stability of a candidate exit point $\tilde{\mathbf{Z}}$ is checked by analyzing the spectrum of the Jacobian matrix, and the condition (ii) is verified when the sign of $Z_l'(\tilde{\mathbf{Z}})$, given by $f_l(\tilde{\mathbf{Z}})$, $\forall l \in \mathcal{L}_{\tilde{F}}$, is coherent with the value of $\tilde{Z}_l$, namely $f_l(\tilde{\mathbf{Z}}) > 0$ and $\tilde{Z}_l = 1$ or $f_l(\tilde{\mathbf{Z}}) < 0$ and $\tilde{Z}_l = 0$.

Remark 2. Let us remind that singular perturbation analysis works out in the limit $q \to 0$, but the calculated solution approximates the solution of Eq. (11) for sufficiently small q ($0 < q \ll 1$). Moreover, it can be proved that the Jacobian matrix is stable for $0 < q \ll 1$. This means that the exit points calculated in the limit also hold for sufficiently small q (Ironi, Panzeri, & Simoncini).

Remark 3. Let us observe that, under *Assumption* $\mathcal{A}$, the reduced equations are always independent of the Z_s occurring in the boundary-layer equations, and that the two sets of equations are mutually independent. For this reason, the behavior of the switching variables in a Z-cube is completely independent of the values of regular variables. Then, the study of the motion in a switching domain may be performed by first analyzing the switching variables, and then the regular ones.

A qualitative simulation algorithm

Among the generic qualitative approaches proposed in the literature, QSIM results to be both the most suitable formalism and algorithm to model and simulate models qualitatively abstracted from ODEs (Kuipers 1994). For this reason, the description of the specialized qualitative algorithm we are developing will be mostly given in accordance with the QSIM jargon.

Qualitative value. The qualitative value of each state variable x_i with domain $\Omega_i = [0, \overline{x}_i]$ is described in terms of its quantity space. In our context, the *quantity space* of x_i is defined by the ordered set Θ_i of its n_i threshold symbolic values. The set Θ_i also contains the endpoints of the domain of x_i, namely 0, and $\overline{x}_i$. The partition, induced by the state variable quantity-spaces, of the whole system domain Ω identifies qualitatively distinct hyper-rectangles D that define all possible *system qualitative values*.

Qualitative state. Let $A(D)$ be the set of domains adjacent to $D \in \Delta$. The *qualitative state* of D, $QS(D)$, is defined by all of its adjacent domains D_k towards which a transition from it is possible:

$$QS(D) = \{D_k \mid D_k \in A(D),\ D \to D_k\}$$

Each transition from D identifies a domain next traversed by a system trajectory. More precisely, if we number by i the domain D traversed at time t_i, each $D_k \in QS(D)$ will be traversed by different trajectories at time t_{i+1}.

State transition. Qualitative simulation of network dynamics is achieved by iteratively applying local transition strategies from one domain to its adjacent domains. The possible transitions from any D are determined by different strategies according to whether $D \in \Delta_r$ or $D \in \Delta_s$.

In the case $D \in \Delta_r$, like in traditional QR methods and in GNA (de Jong *et al.* 2004), transitions are determined by the signs of $\dot{x}_i$. As $\dot{x}_i$ are defined by linear expressions, such signs are easily and uniquely determined by exploiting the inequalities that define the parameter space domain, and constrain the RSPs to belong to specific domains.

In the case $D \in \Delta_s$, a sign-based strategy is not practicable as the expressions for $\dot{x}_i$ are nonlinear. A convenient way to proceed is given by singular perturbation analysis: transitions from D towards adjacent D_k are determined by the locations of the exit points in the associated $\mathcal{Z}(D)$ that can be either on (i) the boundary of $\mathcal{Z}(D)$ or in (ii) its interior. Except in the case (ii), the number of exit points may be greater than one, and especially in a qualitative context where knowledge incompleteness on the parameter values is expressed by coarse constraints. Then, in general, the successors of D are not uniquely determined. But, through symbolic computation procedures, it is possible to calculate the set of inequalities, I_j^i, on parameters that hold when a transition from D_i to D_j occurs. Then, each path from D_i to D_j is clearly identified by the 3-tuple $\langle D_i, D_j, I_j^i \rangle$.

Qualitative behavior. A finite sequence of paths, where each path is clearly both linked and consistent with its predecessor and successor, defines a *qualitative behavior*:

$QB = \langle D_0, I_0 \rangle, \langle D_0, D_1, I_1^0 \rangle, \ldots \langle D_k, D_i, I_i^k \rangle, \ldots \langle D_F, I_F \rangle$, where D_0 is the initial domain, and D_F either contains a stable fixed point or identifies a cycle, i.e it is an already visited domain. I_0 is the initial set of inequalities that defines the parameter space domain, and I_F the set of inequalities on parameter values associated with D_F.

Qualitative simulation. Starting from an initial domain D_0 and a set, I_0, of symbolic inequalities on parameter values, qualitative simulation generates all possible state transitions, and represents them by a directed tree rooted in D_0, $\mathrm{BT}(D_0)$, where the vertices correspond to D_i, and the arcs, labeled by the inequalities I_j^i, to the transitions from D_i to D_j. Each branch in $\mathrm{BT}(D_0)$ defines a qualitative trajectory from D_0, that occurs when the values of parameters satisfy its related inequalities. Then, although, from a strictly computational point of view, the tree representation is a little bit less convenient than the graph one, it is much preferable as it enables an easier, more direct, and less ambiguous interpretability of the simulation outcomes.

Given as input, (i) n symbolic state equations of the form (6); (ii) n quantity spaces $\Theta_i = \{\theta_{ij}\}$ of the state variable x_i; (iii) $D_0 \in \Delta$; (iv) a set of symbolic inequalities I_0 on parameters defining a parameter space domain PSD_0, the algorithm provides as output $\mathrm{BT}(D_0)$. Its main steps are outlined in the following:

1. *Calculate* the qualitative state $QS(D_i)$ of the current domain D_i, or equivalently the possible transitions from D_i.
2. *Determine constraints* I_k^i on parameters for each path $e_k = D_i \to D_k$, where $D_k \in QS(D_i)$.
3. *Append* $\langle D_i, D_k, I_k^i \rangle$ to $\mathrm{BT}(D_0)$ if I_k^i are consistent with

the initial constraints I_0, and *mark* D_i as visited domain.

4. *Repeat* from step 1 for each not visited D_k.

Step 1 is the core of the overall algorithm, and it requires two separate algorithms to implement the different strategies adopted according to whether $D_i \in \Delta_r$ or $D_i \in \Delta_s$. Both algorithms calculate the conditions on parameters I_k^i that should hold for a possible transition from D_i to D_k. A transition from D_i to D_k actually occurs, and then $D_k \in QS(D_i)$, only if the set I_k^i is consistent with the set I_0. Let us define I_k^i *consistent* with I_0 when it defines a not empty parameter space domain PSD_k^i such that $PSD_k^i \subseteq PSD_0$. Furthermore, as both the algorithms involve the calculation of the relative positions of two regions we define the relative position of D_1 with respect to D_2, indicated by $V(D_1, D_2) = \{v_j\}_{j=1}^n$ where $v_j \in \{-1, 0, 1\}$, by the comparison of the intervals defining D_1 and D_2, where D_1 and $D_2 \in \Delta$.

Transition from a regular domain

The algorithm in charge of the construction of the possible paths from regular domains is, in principle, similar to that one proposed by GNA, but it is more informative as it calculates the I_k^is. As $\delta(q) \to 0$ for $q \to 0$, for the sake of simplicity, we indicate $D_i \in \Delta_r$ by the product $D_i = \prod_{j=1}^n (\theta_{ji}, \theta_{j(i+1)})$ where $(\theta_{ji}, \theta_{j(i+1)})$ denotes the interval of x_j in D_i. In outline, the algorithm performs the following steps:

1- *Calculate $A(D_i)$ and state equations in D_i.* The algorithm calculates the set $A(D_i)$, the symbolic state equations in D_i (9), and its focal point $\mathbf{x}^*$. As an example, let us consider the domain D_{11}: $A(D_{11}) = \{D_6, D_7, D_{12}, D_{16}, D_{17}\}$, the state equations (10) have a stationary solution $\mathbf{x}^* = (\frac{\kappa_{11}}{\gamma_1}, \frac{\kappa_{21}}{\gamma_2})$.

2- *Calculate I_k^i and possible transitions.* $\forall D_k \in A(D_i)$, the algorithm calculates the set of inequalities on parameters I_k^i that need to be fulfilled to have a transition from D_i to D_k. As in D_i all the equations (9) are linear, and all the trajectories head towards a focal point $\mathbf{x}^*$ in a regular domain, such inequalities are calculated by imposing that the signs of state variable rates match the relative position of D_k with respect to D_i. Let $V(D_k, D_i) = \{v_j\}_{j=1}^n$ be the relative position of D_k with respect to D_i. I_k^i, initialized to I_0, is updated, $\forall j \in \{1, \ldots, n\}$, with either the inequality $(\frac{\mu_j}{\gamma_j} > \theta_{j(i+1)})$ if $v_j = 1$ or $(\frac{\mu_j}{\gamma_j} < \theta_{ji})$ if $v_j = -1$. Thus, if the calculated inequality set defines a not empty parameter space domain $PSD_k^i \subseteq PSD_0$ then a transition towards D_k is possible and the qualitative state $QS(D_i)$ is updated accordingly. As an example, let us define I_0 as follows:

$$I_0 : (\frac{\kappa_{11} + \kappa_{12}}{\gamma_1} > \theta_{11}) \quad \wedge \quad (\theta_{21} < \frac{\kappa_{21}}{\gamma_2} < \theta_{22}) \quad (15)$$

Then, transitions from D_{11} in Fig. 1 are possible under the following conditions on parameters:

$$\begin{array}{lll} I_6^{11}: & \dot{x_2} < 0 \Rightarrow (\frac{\kappa_{21}}{\gamma_2} < \theta_{21}) & \text{to go to } D_6 \\ I_{12}^{11}: & \dot{x_1} > 0 \Rightarrow (\frac{\kappa_{11}}{\gamma_1} > \theta_{11}) & \text{to go to } D_{12} \\ I_7^{11}: & \dot{x_1} > 0, \dot{x_2} < 0 \Rightarrow I_6^{11} \wedge I_{12}^{11} & \text{to go to } D_7 \\ I_{16}^{11}: & \dot{x_2} > 0 \Rightarrow (\frac{\kappa_{21}}{\gamma_2} > \theta_{22}) & \text{to go to } D_{16} \\ I_{17}^{11}: & \dot{x_1} > 0, \dot{x_2} > 0 \Rightarrow I_{12}^{11} \wedge I_{16}^{11} & \text{to go to } D_{17} \end{array}$$

Among the inequalities given above, only I_{12}^{11} is not in disagreement with I_0. Thus, a possible transition from D_{11} towards D_{12} occurs when $I_{12}^{11} \wedge I_0$ holds. In other words, $QS(D_{11}) = \{D_{12}\}$.

3- *Check the existence of a* RSP *in D_i.* A stable point RSP exists in D_i, i.e. $D_i \in QS(D_i)$, if $\widetilde{PSD} \subseteq PSD_0$ and $\widetilde{PSD} \neq \emptyset$, where $\widetilde{PSD}$ is a parameter space domain defined by the set of inequalities $(\theta_{ji} < \frac{\mu_j}{\gamma_j} < \theta_{j(i+1)})$ $\forall j \in \{1, \ldots, n\}$.

Transition from a switching domain

Let $D_i \in \Delta_s$ be defined by the $\sigma(D_i)$ switching variables x_s with their values around θ_s and by the $n - \sigma(D_i)$ regular ones, x_r. In a switching domain, the nonlinear dynamics is characterized by fast and slow motions, respectively associated with x_s and x_r that are independently calculated. Let us reindex the variables x_j, Z_j such that the switching variables come first, and proceed first with the construction of the fast motion.

A - Fast motion. The study of the fast dynamics is performed in $\mathcal{Z}(D_i)$ in the scaled time, and aims at localizing the set of exit points in $\mathcal{Z}(D_i)$ rather than at detailing the dynamics within it. Such points clearly identify the next adjacent domains the trajectories are moving towards from D_i along the x_s directions. To this end, the algorithm proceeds as follows:

1- *Calculate the boundary-layer equations in D_i.* The algorithm symbolically calculates the boundary-layer equations (12) in the Z variables, and defines the mapping $\Sigma_{D_i} : \mathcal{D}_i \to \mathcal{Z}(D_i)$ that states a correspondence between D_i and its adjacent domains D_k with the interior and the elements on the boundary of $\mathcal{Z}(D_i)$. Let $\mathcal{F}$ be the set of both the faces and the interior of $\mathcal{Z}(D_i)$: its generic element $F = \Sigma_{D_i}(D)$, $D \in \Delta_s$ is either a face of $\mathcal{Z}(D_i)$ when $D \in A(D_i)$ or its interior when $D = D_i$.

To exemplify, let us consider the switching domain D_7, which is characterized by fast motion only as both variables are switching in it. In D_7 the boundary-layer system is given by:

$$\begin{aligned} Z'_{11} &= \frac{Z_{11}(1 - Z_{11})}{\theta_{11}}(\kappa_{11}(1 - Z_{11}) + \kappa_{12}(1 - Z_{21}) - \gamma_1\theta_{11}) \\ Z'_{21} &= \frac{Z_{21}(1 - Z_{21})}{\theta_{21}}(\kappa_{21} - \gamma_2\theta_{21}) \end{aligned} \quad (16)$$

The set $\mathcal{F}$ has five elements, the four faces F_k corresponding to D_2, D_6, D_8, D_{12}, and the interior of $\mathcal{Z}(D_7)$ that correspond to D_7. Moreover, the vertices of $\mathcal{Z}(D_7)$ are the

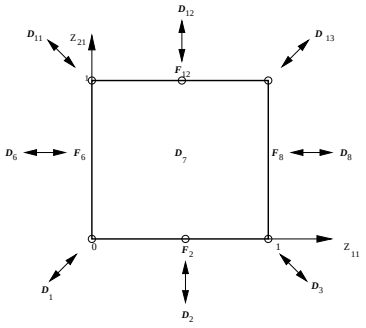


Figure 2: Correspondence between the domains in the phase-plane and the elements of $\mathcal{Z}(D_7)$. The candidate exit-points are denoted by an empty circle.

images, through the mapping Σ, of the adjacent regular domains D_1, D_{11}, D_3, and D_{13}.

2- *Search for stationary points.* Let us denote by EP the set of stationary points, initially made up of the vertices of $\mathcal{Z}(D_i)$. The set of the candidate exit points EP is updated by the possible stationary points on each element of $\mathcal{F}$, that under the *Assumption* $\mathcal{A}$ contains at most one stationary point. To this end, the algorithm symbolically calculates, $\forall F \in \mathcal{F}$, the Jacobian matrix $\mathbf{J}_F$, obtained by removing, $\forall i \in \mathcal{L}_F$, the i-th rows and columns from the Jacobian matrix of the system, and by computing its elements on F.

The Jacobian matrices associated with the elements of $\mathcal{F}$ in the example above are:

$$\mathbf{J}_{F_7} = \begin{pmatrix} -\kappa_{11} & -\kappa_{12} \\ 0 & 0 \end{pmatrix}; \qquad \mathbf{J}_{F_2} = (-\kappa_{11});$$
$$\mathbf{J}_{F_6} = (0); \qquad \mathbf{J}_{F_8} = (0); \qquad \mathbf{J}_{F_{12}} = (-\kappa_{12})$$

As the presence of a non-zero loop is a necessary condition for the existence of a stationary point, the algorithm first searches for a non-zero loop involving all variables in $\mathbf{J}_F$: in case, it symbolically calculates the stationary point on F, and updates accordingly the set of candidate exit points EP.

In the example, only $\mathbf{J}_{F_2}$ and $\mathbf{J}_{F_{12}}$ have a non-zero loop. Then, the algorithm looks for the stationary state on F_2 and F_{12}: $\tilde{\mathbf{Z}}^{\mathbf{2}} = (1 + \frac{\kappa_{12}}{\kappa_{11}} - \frac{\gamma_1\theta_{11}}{\kappa_{11}}, 0)$ and $\tilde{\mathbf{Z}}^{\mathbf{12}} = (1 - \frac{\gamma_1\theta_{11}}{\kappa_{11}}, 1)$. Finally, the exit point candidate set is updated with the points $\tilde{\mathbf{Z}}^{\mathbf{2}}$ and $\tilde{\mathbf{Z}}^{\mathbf{12}}$ (Fig. 2).

3- *Calculate I_k^i and possible transitions by checking stability of stationary points.* The inequality set I_k^i, initialized to I_0, is calculated for each candidate exit point $\tilde{\mathbf{Z}}^{\mathbf{k}} = \{\tilde{Z}_s^k\} \in EP$ by requiring that each point fulfills stability conditions. In addition, for those $\tilde{\mathbf{Z}}^{\mathbf{k}}$ located on elements of $\mathcal{F}$, I_k^i is further constrained by the inequalities on parameters that impose $0 < \tilde{Z}_s^k < 1$ for each $\tilde{Z}_s^k \notin \{0, 1\}$. The algorithm checks the stability conditions (i) by analyzing the spectrum of the Jacobian matrix $\mathbf{J}_F$, and (ii) by imposing conditions on the sign of $Z_l'(\tilde{\mathbf{Z}}^{\mathbf{k}})$, given by $f_l(\tilde{\mathbf{Z}}^{\mathbf{k}})$, $\forall l \in \mathcal{L}_F$. The latter condition is easily checked as it is of the form $(f_l(\tilde{\mathbf{Z}}^{\mathbf{k}}) > 0)$ if $\tilde{Z}_l = 1$ and $(f_l(\tilde{\mathbf{Z}}) < 0)$ if $\tilde{Z}_l = 0$, while the former one is

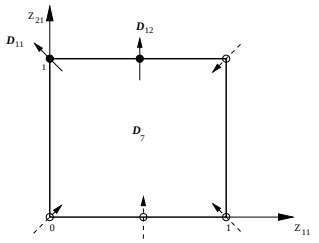


Figure 3: $\mathcal{Z}(D_7)$ and the exit-points denoted by a filled circle. The empty circles denote unstable stationary points that correspond to possible entrance points to the domain.

checked by using concepts from graph theory, and the usual definition of stability based on the sign of the eigenvalues of $\mathbf{J}_F$. Due to *Assumption* $\mathcal{A}$, $\mathbf{J}_F$ has just one element per row and per column. Then, reordering the variables leads to a matrix $\mathbf{J}_F$ block-structured, where each block is a permutation matrix associated with a sub-loop. It follows that the characteristic equation $|J_F - \lambda I| = 0$ is:

$$\prod_{i=1}^{m} (\lambda^{l(i)} + L_i) = 0 \tag{17}$$

where the roots of the equation above, λ_i, are the eigenvalues, m is the number of sub–loops of $\mathbf{J}_F$, l_i is the i–th sub–loop, $l(i)$ is the length of l_i and L_i is the loop product of l_i. As stability is guaranteed when the eigenvalues of $\mathbf{J}_F$ have not positive real part, we exclude the case $l(i) > 2$. Then, $\tilde{\mathbf{Z}}$ is stable if: (i) $\mathbf{J}_F$ has no blocks with dimension strictly greater than 2; (ii) in blocks with $l(i) = 1$, $L_i = b_i < 0$; (iii) in blocks with $l(i) = 2$, the product of the non-zero elements is negative.

The stable points located on $\mathcal{Z}(D_i)$ clearly identify the set of all possible exit domains, i.e. those domains towards which a transition from D_i is possible. Such domains are easily calculated by the algorithm by applying the map Σ^{-1} to each element of $\mathcal{Z}(D_i)$ that contains an exit point. Let us observe that the remaining unstable stationary points in EP are possible entrance points to D_i.

Going back to the example, both exit points $\tilde{\mathbf{Z}}^{\mathbf{2}}$ and $\tilde{\mathbf{Z}}^{\mathbf{12}}$ fulfill the stability condition (i) as $-\kappa_{11} < 0$ and $-\kappa_{12} < 0$. The condition (ii) on variable $Z_l, l = 2$ requires that :

$$I_{s,2}: \quad f_2(\tilde{\mathbf{Z}}^{\mathbf{2}}) < 0 \quad \Rightarrow \quad \kappa_{21} - \gamma_2\theta_{21} < 0 \tag{18}$$
$$I_{s,12}: \quad f_2(\tilde{\mathbf{Z}}^{\mathbf{12}}) > 0 \quad \Rightarrow \quad \kappa_{21} - \gamma_2\theta_{21} > 0 \tag{19}$$

The inequality $I_{s,12}$ defined by (19) is compatible with (15), but the inequality $I_{s,2}$ is not. Then, $\tilde{\mathbf{Z}}^{\mathbf{2}}$ is removed from the exit point set. To be an exit point $\tilde{\mathbf{Z}}^{\mathbf{12}}$ must satisfy the condition:

$$I_{(0,1),12}: \; 0 < \tilde{Z}_1^{12} < 1 \quad \Rightarrow \quad (\kappa_{11} > \gamma_1\theta_{11})$$

Finally, $\tilde{\mathbf{Z}}^{\mathbf{12}}$ is an exit point if I_{12}^7, defined by $I_0 \wedge I_{s,12} \wedge I_{(0,1),12}$, holds.

As for vertices in the example, the stability condition is fulfilled in the point $\tilde{\mathbf{Z}}^{\mathbf{11}} = (0, 1)$ that corresponds to the vertex defined as image of D_{11} by the map Σ.

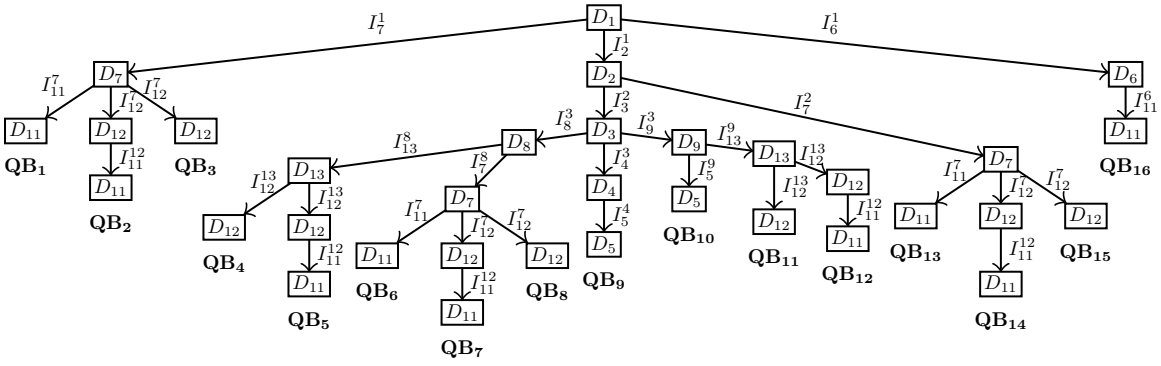


Figure 4: Behavior tree rooted in D_1.

Finally, the only *exit domains* are D_{12}, D_{11}, and then $QS(D_7) = \{D_{12}, D_{11}\}$ (Fig. 3).

B - Slow motion. The slow dynamics of regular variables x_r is studied in the normal time in the usual frame of reference, and it is reconstructed from the reduced system through the same symbolic procedure given for regular domains.

Locating SSPs in D_i. Let us consider the general case when the domain model is characterized by both switching and regular variables. The motion towards a SSP, that occurs in a sliding mode along a stable point in the slow-manifold of the boundary-layer system, is described, in the normal time, by the reduced system. A stable stationary point exists in D_i, i.e. $D_i \in QS(D_i)$, if a stable point exists in the interior of $\mathcal{Z}(D_i)$, and the regular variable rates are *zero* in a point inside D_i. The set $I_{r,i}$ of inequalities that checks the latter condition are defined by $\theta_{ji} < \frac{\mu_j}{\gamma_j} < \theta_{j(i+1)}$ $\forall j \in \{\sigma(D_i)+1, \ldots, n\}$. Let us observe that, due to *Assumption* $\mathcal{A}$, any $\mathcal{Z}(D_i)$ can at most contain one internal stable point. This occurs if all variables in $\mathcal{Z}(D_i)$ are involved in a loop.

As an example, let us consider again the domain D_{12}, and check if it contains one SSP. In D_{12} the boundary-layer system is given by Eq. (14), and the stationary state $\tilde{\mathbf{Z}}^{\mathbf{12}} = 1 - \frac{\gamma_1\theta_{11}}{\kappa_{11}}$ in the interior of $\mathcal{Z}(D_{12})$ fulfills the stability conditions when it holds $I_0 \wedge I_{(0,1),12}$, being $I_{(0,1),12}$ the inequality set that constrains it to belong to the interval (0,1).

Finally, the stability condition $I_{r,12} : (\theta_{21} < \frac{\kappa_{21}}{\gamma_2} < \theta_{22})$ on the regular variable x_2 is provided by the reduced equation: $\dot{x_2} = \kappa_{21} - \gamma_2 x_2$. Thus, a SSP exists in D_{12} when parameter values satisfy the set of inequalities $I_0 \wedge I_{(0,1),12} \wedge I_{r,12}$.

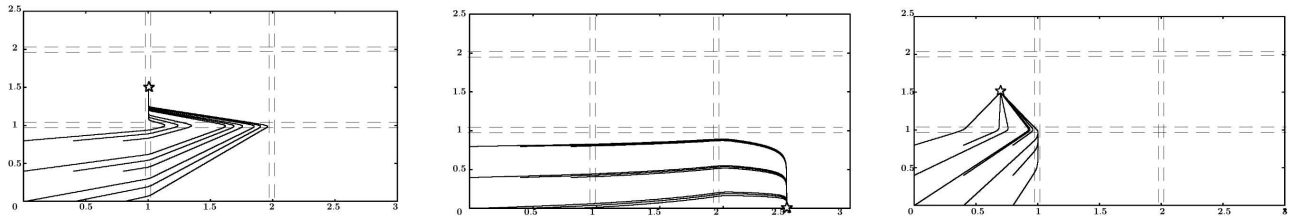
Results

The algorithm is currently under implementation. To illustrate the type of output, let us consider the results of a simulation of the ODE model (8) starting from D_1 with the parameter space defined by the inequalities I_0. The algorithm builds the behavior tree showed in Fig. 4, and calculates the inequalities on parameters, listed in Fig. 5(b), that are associated with each path in $\mathrm{BT}(D_1)$. As $n = 2$, a representation in the phase plane of the trajectories described by the tree, possibly filtered as explained below, is also given (Fig. 5(a)). Three reachable stable states, located in D_{11}, D_{12} and D_5, are identified by the final leaf of each branch in BT. As $D_{12} \in \Delta_s$, one of them is a SSP whereas the others are RSPs. These stable states are reached by different predicted qualitative behaviors, each of them occurring under specific constraints on parameters. For example, the trajectory QB_{16} starting from D_1, crossing D_6, and reaching a RSP in D_{11} is allowed when the inequalities I_6^1, I_{11}^6 and I_{11}, consistent with I_0, hold.

The qualitative simulation outcomes are numerically confirmed. In Fig. 6 we report some of the numerical simulations performed under different conditions. Fig. 6(a) shows trajectories, abstracted by QB_4 and QB_{11}, and characterized by a sliding motion along θ_{11} towards the SSP in D_{12}. The trajectories moving towards the RSP in D_5 (Fig. 6(b)) are abstracted by QB_9, and those moving towards the RSP in D_{11} (Fig. 6(c)) are abstracted by QB_1 and QB_{16}.

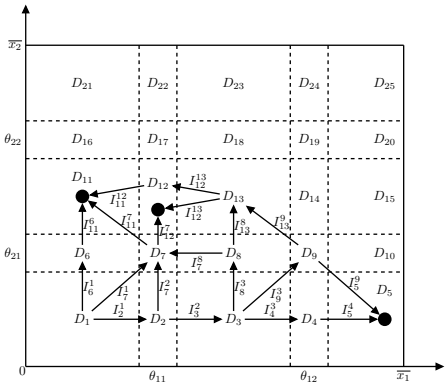
Soundness. The algorithm guarantees that the behavior tree captures all of the sound behaviors for values of q sufficiently small. The determination of a symbolic upper bound, $\overline{q}$, depending on parameters would better characterize the terms of validity of the generated behaviors. However, for a formal proof of soundness we need to prove that assuming stability instead of asymptotic stability does not affect the main results in singular perturbation analysis. Both problems are currently under study, and close to be solved.

Completeness. At present, the algorithm may generate spurious behaviors. There is a twofold explanation for that. First, we have not yet performed a thorough analysis with respect to entrance-exit transition, or in other words, we have not yet solved (i) the problem of identifying the only admissible connections between entrance and exit points. Moreover, singular perturbation analysis is a "local" procedure that works quite well in a quantitative context but that needs, in a qualitative context, to be supported by a "global" criterion when local paths are combined to produce a specific trajectory. For example, the behavior QB_2 is spurious as



(a) (b) (c)

Figure 6: Phase space plots of the numerical simulations performed with different parameter sets and initial conditions taken on an uniform grid of points in D_1. Common parameter values are: $\theta_{11} = \theta_{21} = 1$, $\theta_{12} = \theta_{22} = 2$, $q = 0.01$, $\kappa_{21} = 1.5$, $\gamma_2 = 1$. Other parameters are: (a) $\kappa_{11} = 2.5, \kappa_{12} = 2.5, \gamma_1 = 1$; (b) $\kappa_{11} = 25, \kappa_{12} = 2.5, \gamma_1 = 10$; (c) $\kappa_{11} = 0.7, \kappa_{12} = 0.7, \gamma_1 = 1$.



(a)

I_{12}^7, I_{12}	$I_0 \wedge (\frac{\kappa_{11}}{\gamma_1} > \theta_{11})$
I_3^2	$I_0 \wedge (\frac{\kappa_{12}}{\gamma_1} > \theta_{11})$
$I_5^9, I_9^3, I_4^3, I_5^4$	$I_0 \wedge (\frac{\kappa_{12}}{\gamma_1} > \theta_{12})$
$I_{11}, I_{11}^7, I_{11}^{12}$	$I_0 \wedge (\frac{\kappa_{11}}{\gamma_1} < \theta_{11})$
I_7^2	$I_0 \wedge (\frac{\kappa_{12}}{\gamma_1} < \theta_{11})$
I_5	$I_0 \wedge (\theta_{12} < \frac{\kappa_{12}}{\gamma_1} < \overline{x}_1)$
$I_2^1, I_6^1, I_7^1, I_8^3$ $I_{11}^6, I_7^8, I_{13}^8, I_{12}^{13}, I_{13}^9$	I_0

(b)

Figure 5: (a) Phase space representation of trajectories described by BT after filtering; • denotes a stable state. (b) Inequalities calculated by the algorithm.

I_{12}^7 is not consistent with I_{11}^{12}. Similarly, QB_7 and QB_{14} are spurious. We are quite confident that the automatic analysis of the consistency of the whole sequence of inequalities that characterizes a behavior together with the solution of problem (i) will allow us to filter out all spurious solutions, and to prove the completeness of the algorithm.

Conclusion and future work

The qualitative simulation algorithm we propose works for models of GRNs with continuous sigmoid response functions. The continuity assumption makes the simulation problem hard to be tackled but it is crucial in view of the realization of tools that can be gradually extended to tackle more and more realistic models. The algorithm is grounded on a set of symbolic computation algorithms that carry out the integration of qualitative reasoning techniques with singular analysis perturbation methods: the former techniques allow us to cope with uncertain and incomplete knowledge whereas the latter ones lay the mathematical groundwork for a sound and complete algorithm capable to deal with regulation processes that occur at different time-scales.

As for symbolic calculus, the algorithm requires to tackle complex tasks, such as: (i) update an inequality set with an another one; (ii) check the consistency of two sets of inequalities I_1 and I_2; (iii) solve systems of equations; (iv) find cycles in the Jacobian matrix. As for (iii), the original equations are multilinear in Z_s, but due to *Assumption* $\mathcal{A}$ they assume a linear form in the boundary layer, and then they can be straightforward solved and analyzed for stability. Also the solution of problems (i) and (ii) benefits from *Assumption* $\mathcal{A}$ as the inequalities are always linear. Then, thanks to the *Assumption* $\mathcal{A}$, and to algorithms proposed both by the literature and common symbolic computation package, such as Mathematica (Wolfram 2003), the tasks (i)-(iii) are simplified and feasible. As for the task (iv), it is performed by using cycle–detection algorithms and tools of matrix graph theory (Gross & Yellen 2006).

The characterization of the paths from one domain to the next ones by sets of inequalities constraining the model parameters is quite new in the field of qualitative simulation, as for both general-purpose and specifically tailored algorithms. Such a strategy may reveal quite useful in the def-

inition of a "global criterion" that allows us to distinguish sound behaviors from spurious ones by requiring that the sets of inequalities that label the local paths in a specific trajectory are consistent with each other. Both the definition of such a criterion and its implementation are not a trivial task, especially from a computational point of view. As for algorithm completeness, another essential methodological and computational issue to be deepened deals with the definition of the transition map that states the *proper* connection of the entrance points to the exit points associated with a switching domain. Moreover, the complex nonlinearities of the models we are interested in require to design methodological and computational methods to deal with possible aspects of the model dynamics that we have ignored herein, such as *limit cycles*.

Acknowledgement

The authors gratefully acknowledge Olivier Dordan, Erik Plahte and Valeria Simoncini for the useful discussions on the different methodological aspects and mathematical problems related to the work described in this paper.

References

Bacciotti, A. 2003. Some remarks on generalized solutions of discontinuous differential equations. *Int. Journal of Pure and Applied Mathematics* 10(3):257–266.

de Jong, H.; Gouzé, J. L.; Hernandez, C.; Page, M.; Sari, T.; and Geiselmann, J. 2004. Qualitative simulations of genetic regulatory networks using piecewise linear models. *Bulletin of Mathematical Biology* 66(2):301–340.

de Jong, H. 2002. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology* 9(1):67–103.

Dordan, O.; Ironi, L.; and Panzeri, L. Some critical remarks on GNA. *in preparation*.

Glass, L., and Kauffman, S. A. 1973. The logical analysis of continuous, nonlinear biochemical control networks. *Journal of Theoretical Biology* 39(1):103–129.

Glass, L. 1977. Global analysis of nonlinear chemical kinetics. In Berne, B., ed., *Statistical Mechanics, Part B: Time Dependent Processes*, 311–349. Plenum Press, New York.

Gouzè, J. L., and Sari, T. 2003. A class of piecewise linear differential equations arising in biological models. *Dynamical systems* 17:299–316.

Gross, J., and Yellen, J. 2006. *Graph Theory and its Applications*. New York: Chapman & Hall/CRC Press.

Hasty, J.; McMillen, D.; Isaacs, F.; and Collins, J. 2001. Computational studies of gene regulatory networks: *In numero* molecular biology. *Journal of Computational Biology* 2:268–279.

Holmes, M. 1995. *Introduction to Perturbation Methods*. Berlin: Springer.

Ironi, L.; Panzeri, L.; and Simoncini, V. Matrix perturbation analysis for qualitative simulation of the dynamics of gene-regulatory networks. *in preparation*.

Kuipers, B. 1994. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. Cambridge, MA: MIT Press.

Plahte, E., and Kjøglum, S. 2005. Analysis and generic properties of gene regulatory networks with graded response functions. *Physica D: Nonlinear Phenomena* 201(1-2):150–176.

Plahte, E.; Mestl, T.; and Omholt, S. W. 1998. A methodological basis for description and analysis of systems with complex switch-like interactions. *Journal of Mathematical Biology* 36(4):321–348.

Veflingstad, S. R., and Plahte, E. 2007. Analysis of gene regulatory network models with graded and binary transcriptional responses. *Biosystems* 90:323–339.

Wolfram, S. 2003. *The Mathematica Book*. Wolfram Media.

Using Qualitative Reasoning in Building Ubiquitous Computing System

Hyeon-Kyeong Kim

Department of Information Science and Telecommunication, Hanshin University
411 Yangsan-dong, Osan-si, Gyeonggi-do, 447-791, Korea
hkim@hs.ac.kr

Abstract

A key problem in building ubiquitous computing system is providing context aware and adaptive services to users by cooperation among various computing objects. In this paper, we show how an integration of qualitative reasoning and multi-agent based architecture can be used to provide context aware and adaptive services for operation of physical system in ubiquitous computing environment. Differential qualitative analysis is used to propose appropriate services once an abnormal operation of a given physical system is detected. Causal dependencies generated from a qualitative model combined with primitive tasks are traced to restore the given physical system to normal state. This idea has been implemented and tested on several tasks including the operations of incineration plant and pyrolysis reactor.

Introduction

Recently, computing-device-rich and communication-resource-rich environment has introduced a new computing paradigm called ubiquitous computing. The goal of ubiquitous computing is to provide seamless services to users by cooperation between various computing objects. A key factor in building ubiquitous computing system (UCS) is intelligence that can provide self-configuring, self-repairing and adaptive abilities (Roman et al, 2002; Ranganathan, 2005). We believe that qualitative reasoning (QR) will play an important role in providing the intelligence in UCS. We expect that QR can provide a way for breadth, robustness and natural user interaction. Since QR has been developed to capture both everyday reasoning and expert reasoning, it can provide a firm ground for breadth of system. By robustness, we mean the ability to solve a given problem even with partial knowledge. QR, that solves problems by using qualitative knowledge, already provides a good possibility (Klenk etal, 2005). Lastly, natural interaction with user is an important feature for UCS since it can enhance active user participations as active computing objects in cooperation. Qualitative nature of QR is expected to support this feature.

We are exploring how QR can support intelligent services for a given task in ubiquitous computing environment. Unlike previous work in task execution framework in ubiquitous computing that workflow and primitive tasks are predefined, our framework accomplishes a given task by tracing causal dependencies generated for the task (Ranganathan, 2005). We are focusing on the operation of physical system. In this paper, we show our work by describing a UCS for operation of incineration plant (Kim & Kim, 2007).

Incineration Process Domain

Lately, incineration has drawn attention because of its reduction efficiency of solid wastes and technical stability. A lot of incineration plants have been installed in Korea during the last two decades. However, incineration was blamed to generate air pollutants such as dioxins. Basically it consists of 1st incinerator, 2nd incinerator and flue gas treatment train. Given solid waste and air, it finally produces gases through combustion and thermal destruction of incompletely combusted gas. Ideally if the working condition is controlled, this process should not produce air pollutants. But it is impossible in real world. What we can do is to control parameters of system to avoid undesired conditions as much as possible.

While many efforts have been made to develop complete quantitative incineration process model, complicated thermodynamic nature of the process has made it difficult to build such a model. Although such a model has been developed, there are still difficulties to get complete and precise input data, i.e., complete data about solid waste supplied. In addition to this, it is not easy for non-expert to interpret numerical results.

Problem Solving with a Qualitative Model

A qualitative model provides a basis for problem solving. The expert knowledge about incineration process is often imprecise and of qualitative nature. Our qualitative incineration process model captures the qualitative knowledge of the experts. It is built based on qualitative

process theory (Forbus, 1984). Figure 1 shows a part of causal dependencies generated from a qualitative incineration model.

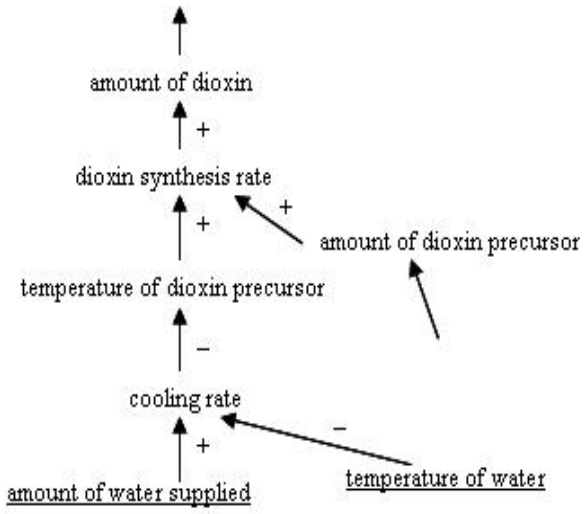


Figure 1: A part of causal dependencies of incineration process

We are focusing on the operation of physical system. Our system concentrates on maintaining normal conditions. Once an abnormal condition occurs, our system tries to restore to a normal state. It implies that system tries to the related quantity to change (e.g., to increase or to decrease) to a normal state. Differential qualitative analysis (DQA) is an inference technique that propagates the effects of a perturbation through causal dependencies (Weld, 1988). We use DQA backwards to find appropriate services for a given task. Problem solver works backwards from the top-level goal to sub-goals by tracing qualitative influences through the causal dependencies. For example, as shown as Figure1 a way to decrease the amount of dioxin is to decrease dioxin synthesis rate. And one way to decrease the rate is to decrease the temperature of dioxin precursor. And the temperature is decreased by increasing cooling rate. The cooling rate is increased by increasing the amount of water supplied. The rate is also increased by decreasing the temperature of water. Then the task to increase the cooling rate is achieved by primitive tasks to control water supplier device and water temperature control device, respectively.

In the figure, underlined parameters represent primitive tasks that are controlled by operating corresponding devices. Services in our framework are primitive tasks performed by devices. To achieve a task, reasoning agent regress back to the primitive tasks through causal relations.

UCS for Operation of Physical System

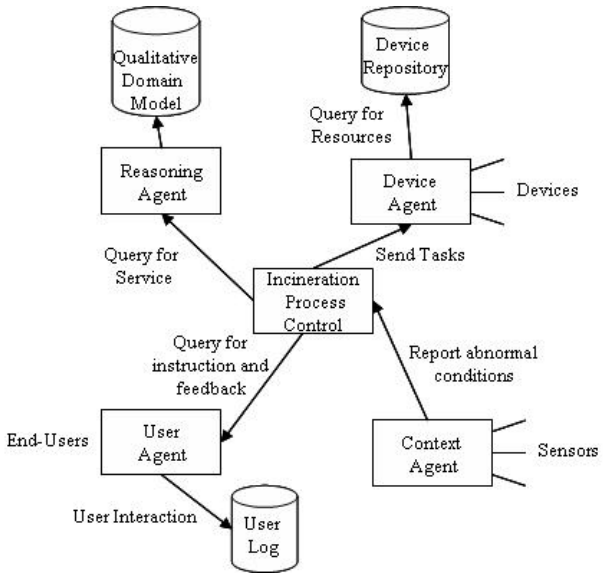


Figure 2: Architecture

Figure 2 shows the overall architecture for our system. In this section, we describe our system by using operation of incineration plant. The system (IPC: Incineration Process Control) consists of multiple agents that execute tasks by cooperation among them. We assume devices and sensors are connected to our framework. The following describes each agent shortly.

Context Agent: Context agent monitors the chosen plant. It collects data about quantities from the sensors attached to the plant, detects deviations from expected values of quantities: either a higher value than expected or a lower value than expected. Once a deviation is detected, this fact is reported to IPC. For example, if the concentration of dioxin is higher than normal value, this is reported to IPC.

Reasoning Agent: Reasoning agent finds solutions, i.e., proper services to change higher or lower value to expected value. It tries to make the quantity with a higher value decrease and to make the quantity with a lower value increase. It figures out the necessary service by applying DQA through causal dependencies generated from qualitative model. Service consists of primitive tasks operated by devices such as "increase water supply by operating water supplier".

Device Agent: Once proper service is suggested by reasoning agent, IPC requests to device agent for relevant devices to perform primitive tasks as directed. Device repository maintains a list of all entities such as incineration plants and their devices. Devices are the entities that can change the values of physical parameters such as air supplier and thermostat.

User Agent: User agent provide user interface. The explanation regarding the execution of the task is given to users and users provide feedback through user agent.

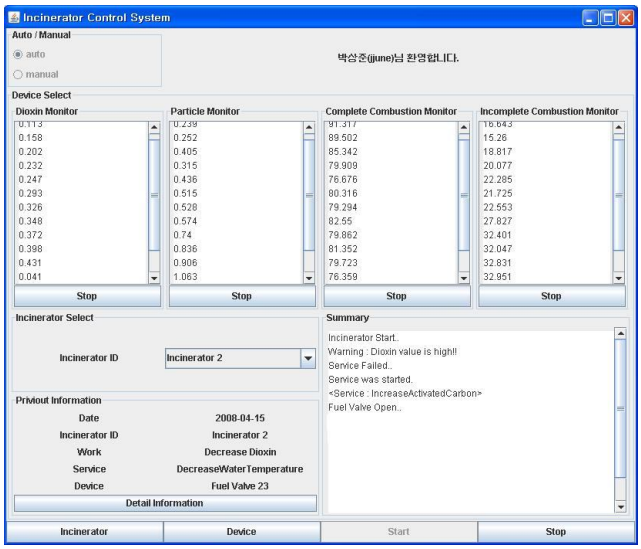


Figure 3: Screenshot of user interface

Figure 3 shows screenshot of user interface in the middle of execution of IPC. The top four windows show the values of quantities monitored by corresponding sensors. Since dioxin level is reported as higher than expected in the figure, reasoning agent finds proper services to lower the level. The window at the right bottom displays explanation regarding the perturbation of the quantity and the execution of the services performed by device agent.

Our system has been implemented in Java. Qualitative incineration model and reasoning agent have been implemented by using Jess. Jess is a rule engine for the Java platform. Agent communications has been implemented with JADE. JADE is Java agent Development Framework that supports developing multi-agent system.

Discussion

This work shows our effort to combine QR with multi-agent framework for building an intelligent UCS. We are exploring how QR can provide context aware and adaptive services in ubiquitous computing environment. We believe that UCS is a good application domain where QR can show its strengths. We plan to keep exploring the possibilities.

Acknowledgement

This research is supported by Foundation of ubiquitous computing and networking project (UCN) Project, the Ministry of Knowledge Economy(MKE) 21st Century Frontier R&D Program in Korea and a result of subproject UCN 08B3-S2-10M.

References

Forbus, K. 1984. *Qualitative Process Theory*. Artificial Intelligence

Jess. http://herzberg.ca.sandia.gov/jess/

JADE. http://jade.tilab.com/

Klenk, M. et al. 2005. *Solving Everyday Physical Reasoning Problems by Analogy using Sketches*. AAAI

Kim, H. and Kim, S. 2007. *Multi-agent Based Incineration Process Control System with Qualitative Model*. 10th Pacific Rim Int'l Workshop on Multi-Agents

Ranganathan, A.. 2005. *A Task Execution Framework for Autonomic Ubiquitous Computing*. PhD thesis, University of Illinois at Urbana-Champaign

Roman, M. et.al, 2002. *Gaia: A middleware Infrastructure to Enable Active Spaces*. IEEE Pervasive Computing

Weld, D. 1988. *Comparative Analysis*. Artificial Intelligence.

Learning Modeling Abstractions via Generalization

Matthew Klenk, Scott E. Friedman, Kenneth D. Forbus

Qualitative Reasoning Group, Northwestern University
21331 Sheridan Rd, Evanston, IL 60208 USA
{m-klenk, friedman, forbus}@northwestern.edu

Abstract

In domains with everyday scenarios, an important aspect of model formulation concerns moving from broad descriptions to the technical abstractions necessary for effective problem-solving. We present a method for learning how to make abstraction decisions from experience via analogical generalization. Specifically, we generalize abstraction decisions from worked examples, abstracting away irrelevant information. When faced with a new situation, our method compares the entities in the situation with the generalizations, and makes its decision by using the best match. We argue that the similarity score from the comparison is an effective heuristic for judging the quality of the modeling decision. Using textbook physics problems, we show that our method can make accurate abstraction decisions, and that these decisions improve as the system gains experience.

Introduction

One of the important contributions of qualitative reasoning has been formalizing the process of model formulation (cf. Falkenhainer & Forbus 1991; Nayak 1994; Rickel & Porter 1994). Most model formulation work has focused on ascertaining what levels of detail and which perspectives should be used in a model, given a particular task. In general, model formulation research has ignored the problem of computing structural descriptions, *i.e*. moving from the broad set of concepts used in everyday life to a concise, technical vocabulary of abstractions that can be used effectively for problem-solving. We use the term *participant abstraction* to refer to the type of a participant in a domain theory, and the term *scenario entity* to refer to an entity within the everyday domain scenario. This work addresses how decisions about participant abstractions can be learned. Specifically, we use analogical techniques from structure-mapping theory (Gentner 1983) to decide how to represent everyday entities in a scenario model.

The typical method for making these participant abstraction decisions is as follows. Given a scenario and a domain theory, one can use the type of each scenario entity in an ontology to determine its appropriate participant abstraction in the domain theory. For example, in the ResearchCyc[1] ontology, the collection `Coin` is a specialization of the collection `PartiallyTangible`. Consequently, we could write a rule stating that a `PartiallyTangible` should be considered a `PointMass` in a model. This rule-based approach is problematic for several reasons. First, these rules would contain false positives (e.g. a `Lake`, which is a `PartiallyTangible`, should not be considered a point mass in most situations). Second, participant abstraction decisions are very contextual. While a coin falling off a building could be considered a `PointMass`, the same coin spinning on a table, in a rotational mechanics problem, should be a viewed as an object with extent. Accounting for this necessary contextual information greatly increases the complexity of such rules. As noted by Falkenhainer and Forbus (1991), modeling rules are very domain specific; that is, for each new domain a knowledge engineer will have to construct a new set of rules.

We propose an alternative method that learns from examples the necessary connections between everyday scenario entities and participant abstractions to construct scenario models. Our method uses psychological simulations of analogical processing to learn these participant abstraction decisions.

This paper uses physics problem solving to demonstrate our method, but we believe it is applicable across a wide range of domains. We begin by summarizing the analogical processing components we use and our representations of the physics domain. Next, we describe how participant abstraction decisions can be learned from examples, using generalization. We present experimental results demonstrating the effectiveness of our method. Finally, we close with a discussion of related work and future work.

Background

Our approach to learning participant abstraction decisions utilizes the SEQL generalization model (Kuehne *et al*. 2000). SEQL constructs generalizations incrementally via analogical comparison using SME, the Structure-Mapping Engine (Falkenhainer *et al*. 1989). For this work, the

[1] http://research.cyc.com/ - a large scale effort to formalize commonsense knowledge

generalizations are formed over example participant abstraction decisions from physics problems. We begin with an overview of the analogical processes utilized in our method, and then we describe the participant abstractions of our domain theory and physics representations.

Analogical Processes: SME and SEQL

We use Gentner's (1983) structure-mapping theory, which postulates that analogy and similarity are computed via structural alignment between two representations (the *base* and *target*) to find the maximal structurally consistent match. For maximality, structure-mapping uses the principle of *systematicity*: mappings that are highly interconnected and contain deep chains of higher order relations are preferred.

SME simulates analogical matching. It takes as input two structured representations (base and target). It produces one or more *mappings* that describe how the two representations can be aligned. A mapping includes *correspondences* that link items (entities and relations) in one representation with items of the other, a *structural evaluation score* which reflects the quality of the entire mapping, and a set of *candidate inferences* that are conjectures about the target created by projecting partially mapped base expressions.

SEQL simulates analogical generalization. It maintains a list of generalizations and exemplars, which are structured representations, called a *generalization context*. It takes as input a sequence of new examples. Given a new example, SME is used to compare it to the existing generalizations. When the structural evaluation score is above the *assimilation threshold*, the example is assimilated into that generalization by keeping the common overlapping structure. If the example is not assimilated into an existing generalization, it is compared against the exemplars. Again, if it is sufficiently similar to another exemplar, a new generalization is created from the common structure. Otherwise, the new example is added to the list of exemplars.

Physics problem-solving and QP theory

de Kleer's (1977) pioneering work emphasized the importance of modeling in solving physics problems. Given a domain theory and a physics problem, a problem-solver must make a number of modeling decisions to arrive at the correct solution. Consider a problem where a ball is dropped off the top of a building. The ball should be considered a point mass, and the falling event should be considered a constant linear acceleration event. These are examples of participant abstraction decisions and are the focus of this paper. Solving this problem requires additional modeling decisions, including assuming that the event occurs on Earth. As noted in related work, analogy may be useful for learning how to make these types of decisions as well. In this paper, these other modeling decisions are made via hand-coded rules, so we can focus exclusively on participant abstraction decisions here. Determining which abstraction to apply, given problem scenarios whose entities can range over tens of thousands of possible categories, is quite challenging.

Our physics domain theories consist of *encapsulated histories* (Forbus 1984) that represent physics equations. Encapsulated histories, unlike model fragments, permit constraints to be placed on the duration of events and time intervals. The encapsulated histories for our physics domain theory include participant abstractions such as `PointMass` and `ConstantTranslationAccelerationEvent`. These abstractions are not used within the scenario descriptions of physics problems; rather, the scenario entities are encoded as real-world objects such as `Automobile` and `Driving`. Moving from the real-world entities to the technical language for problem-solving is one kind of *simplifying assumption* (Falkenhainer & Forbus 1991).

```
(def-encapsulated-history
  VelocityByTime-1DConstantAcceleration
  :participants
 ((theObject :type PointMass)
  (theEvent  :type
             ConstantTranslationAccelerationEvent))
 :conditions
 ((primaryObjectMoving theEvent theObject))
 :consequences
 ((equationFor VelocityByTime
  (mathEquals
    (AtFn (Speed theObject) (EndFn theEvent))
      (PlusFn
          (AtFn (Speed theObject)
            (StartFn theEvent))
      (TimesFn
          (AtFn (Acceleration theObject) theEvent)
          (Time-Quantity theEvent)))))))
```

Figure 1: Encapsulated history definition

Figure 1 shows the definition for the encapsulated history representing the equation $v_f = v_i + at$, velocity as a function of time. The two participants, `theObject` and `theEvent`, must satisfy their type constraints, `PointMass` and `ConstantTranslationAccelerationEvent`, respectively. Furthermore, the conditions of the encapsulated history must be satisfied in order to instantiate it and conclude its consequences. In this example, `theObject` must be the object moving in `theEvent` for the encapsulated history to be instantiated.

The method we describe in this paper learns how everyday entities in problems should be modeled in terms of the abstractions used in the domain theory.

Representing Physics Problems and Examples

The representations used in this work are in CycL, the predicate calculus language of the ResearchCyc knowledge base (Matuszek *et al.* 2006). We use a subset of the ResearchCyc KB, consisting of 33,000+ concepts, and

13,000+ relations, plus our own extensions for QP theory (Forbus 1984) and problem-solving strategies. Consequently, objects, relations, and events that appear in physics problems such as "rotor," "car," and "driving" are predefined in the ontology. This reduces the degree of tailorability in our experiments.

All the problems used in this work were taken from a common physics textbook (Giancoli 1991). We represent the problems and examples as cases, consisting of predicate calculus facts. Consider the following physics problem:

> Suppose a ball is dropped from a 70m tower. How far will it have fallen after 3 seconds?
> *Example problem 2-9, p. 30.*

This problem is represented in our system as a case of 19 facts, a subset of which is shown in Figure 2. There are five entities in the problem: the top of the tower, the tower, the ball, the dropping event, and the 3-second interval. The facts in Figure 2 pertain to the ball's motion during the dropping event, the description of the time interval, and the query of the problem.

```
...
(objectStationary (StartFn Drop-2-10) Ball-2-10)
(primaryObjectMoving Drop-2-10 Ball-2-10)
(directionOfTranslation Fall-2-10 Down-Directly)
(objectTopSide Tower-2-10 Top-2-10)
(fromLocation Drop-2-10 Top-2-10)
(temporallyCooriginating Drop-2-10 Interval-2-10)
(valueOf (Time-Quantity Interval-2-10)
         (SecondsDuration 3))
(querySentence Gia-Query-2-10
  (valueOf
     (DistanceTravelled Ball-2-10 Interval-2-10)
     Distance-2-10))
```

Figure 2: Part of example problem 2-9 representation

One common learning method physics students use is to solve problem sets and compare their answers to worked solutions. This technique motivates the feedback we provide our system. Worked solutions are neither deductive proofs nor problem-solving traces produced by our solver. The worked solution for this example problem consists of five steps:

1. Categorize the problem as a constant acceleration linear mechanics problem
2. Assume that the acceleration of the ball ($a = 10\ m/s^2$)
3. Instantiate the distance by velocity time equation ($d = v_i t + .5at^2$)
4. Because the ball is stationary at the start of the drop infer that its velocity is zero ($v_i = 0$ m/s)
5. Solve the equation for d ($d = 45$ m)

The entire worked solution for this problem consists of 38 facts. The third step is most relevant to the goals of this paper – the instantiation of the distance by the velocity time equation. This step depends upon two abstraction decisions, one for the ball and one for the dropping event, as illustrated in Figure 3. Next we describe how we build generalizations from these example decisions and apply the learned knowledge to new problems.

```
(StepUses Gia-2-10-WS-Step-3
   (abstractionForObject Ball-2-10 PointMass))
(StepUses Gia-2-10-WS-Step-3
   (abstractionForObject Drop-2-10
      ConstantTranslationAccelerationEvent))
```

Figure 3: Worked solutions indicate appropriate participant abstractions for problem entities

Learning Participant Abstraction Decisions

The primary contribution of this work is our method for learning how to make decisions about participant abstractions for problem entities described in everyday terms. Our method is best understood in two stages: generalization and execution. First, we generalize examples of participant abstraction decisions. Then, when faced with a problem, our method uses analogies between the entities in the problem and the generalizations to make participant abstraction decisions. These decisions allow our solver to instantiate the necessary encapsulated histories to solve the problem.

Generalization of Participant Abstractions

We create generalizations at the granularity of the participant abstraction. As such, we contextualize the generalizations such that all examples of a given participant abstraction are considered together. We achieve this with generalization contexts. Each context has an *entry pattern* that exemplars must satisfy to be generalized within. The entry patterns used here reflect the various participant abstractions. Figure 4 depicts the four generalization contexts after generalizing decisions from eight worked solutions.

Our system populates the generalization contexts with exemplars generated from worked solutions. Exemplars are created for each participant abstraction within the worked solutions and generalized within the appropriate contexts. For example, in the worked solution from Figure 3, there are two statements indicating participant abstractions. The statement `(abstractionForObject Ball-2-10 PointMass)` signals that an exemplar case should be constructed, including all statements that mention the entity `Ball-2-10` in the problem plus the `abstractionForObject` statement. Since the worked solution contains a second participant abstraction, the system generates a separate exemplar in the same manner for the entity `Drop-2-10`. Next, these exemplars are added to their appropriate generalization contexts as indicated by Figure 4 and generalized via SEQL as described above.

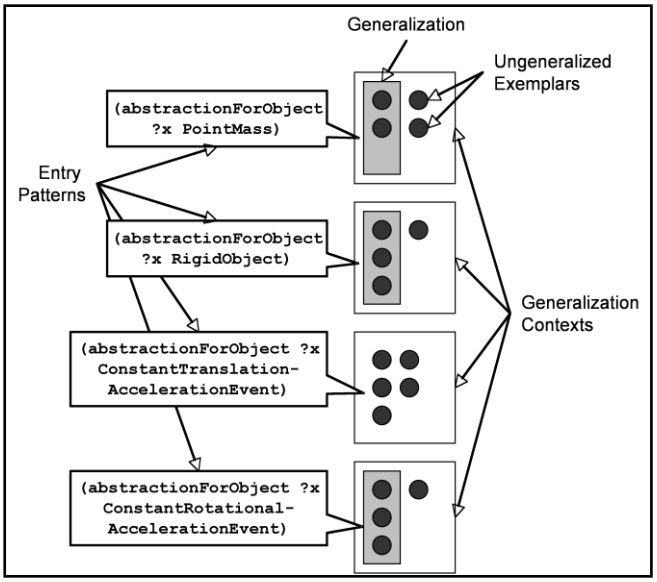


Figure 4: Example generalization contextualization

This allows the system to maintain several contexts simultaneously, each representing a participant abstraction with its own lists of generalizations and ungeneralized exemplars.

As new worked solutions are made available, our method builds participant abstraction examples and adds them to the appropriate generalization contexts. Therefore, our method learns incrementally by refining and extending its generalizations.

Making Participant Abstraction Decisions

Given a problem, a domain theory, and contextualized generalizations of participant abstractions, our method uses analogy to determine if and how entities in the problem should be included in the model. In addition to making the modeling decision, our method returns a *confidence* value (0-1) as a heuristic for confidence in the decision.

The algorithm listed in Figure 5 is performed on every entity in the problem. The process consists of three steps: building a case around the entity, comparing it against the best match from each generalization context, and deciding which, if any, abstraction is appropriate for the entity.

Our method begins by building an *entity case* from the entity. As in building the worked solution exemplars, we include all facts in the problem that mention the entity. The system then compares this case to each generalization context.

From each generalization context, our method identifies the generalization or exemplar with the highest structural evaluation score via SME comparison with the entity case. The systematicity principle implemented in SME means that matches with deeper relational structures have higher structural evaluation scores; therefore, the best mapping for

1) Given entity, e, from problem, P
   a) Build entity case, ec, with each fact in P mentioning e
2) For each Generalization Context gc_i
   a) Compare ec with each exemplar and generalization within gc_i
   b) Use the best matching exemplar or generalization as the base of an analogy with ec
   c) If a candidate inference of this match includes a fact of the form: `(abstractionForObject` e gc_i`)`
      i) Return the normalized structural evaluation score for this match as the confidence for this generalization context
      ii) Otherwise, return 0
3) Select as the participant abstraction for e from the generalization with the highest confidence
   a) If all generalization context score 0, do not make a participant abstraction for e

Figure 5: Participant abstraction decision algorithm

a generalization context is not necessarily the largest, but the one with the most relational structure.

The confidence value of the match is computed by analyzing several aspects of the match. First, our method analyzes the candidate inferences of the match between the best match and the entity case. Because every case in the generalization context has an `abstractionForObject` fact, we search the mapping for a corresponding candidate inference in the target (entity case) under consideration. If there is no such candidate inference, the confidence for this abstraction is zero. Otherwise, the confidence value is the SME structural evaluation score normalized against a self-match of the exemplar or generalization. Normalization is necessary for comparing confidence values across generalization contexts. Normalizing against the best match means the maximum confidence score approaches one as the entire exemplar or generalization, aside from the `abstractionForObject` statement, participates in the mapping.

The confidence values are compared, and the system identifies the generalization context that generated the highest confidence value. The participant abstraction represented by this generalization context is selected as the abstraction for the entity. If the highest confidence value is zero, the entity is not considered a participant in the model.

Evaluation

Our evaluation focuses on exploring the following questions. First, is our method able to make accurate participant abstraction decisions? Second, does our method's performance improve as examples are added to the system? Finally, does the confidence value provide a

useful heuristic in determining the accuracy of a participant abstraction for a particular problem entity?

Method

Our materials include five linear kinematics problems and five rotational kinematics problems. In these problems, there are 34 entities, of which 21 should be modeled as one of four different participant abstractions: `PointMass`, `LinearConstantAccelerationEvent`, `RigidObject`, `RotationalConstant-AccelerationEvent`. To evaluate the effect of learning, we created four conditions based upon the size of the training set (2, 4, 6, and 8). To ensure that each generalization context has at least one exemplar, each training set consists of an equal number of problems from linear and rotational kinematics. Using the worked solution for each training set problem, we added participant abstraction exemplars to the appropriate generalization contexts. The remaining problems were used for testing. That is, for each entity in each problem our method selected a participant abstraction based upon the generalizations created by the training set. We evaluated every possible combination of problems for the training sets in each trial (size 2=25 trials, size 4=100 trials, size 6=100 trials, and size 8=25 trials).

For each decision, we compare the result of our method to the desired result, as indicated by the worked solutions. There are five possible results:

1. Correct: The entity was a model participant and identified correctly.
2. Correctly Ignored: The entity was not a model participant and was not identified as one.
3. Extraneous: The entity was not a model participant, and our method selected an abstraction.
4. Wrong: The entity was a model participant, but was identified as the wrong abstraction.
5. Failed: The entity was a model participant, but was not identified as any abstraction by our method.

Correct and correctly ignored answers are considered successful modeling decisions. Extraneous answers result in more participants to consider when formulating the model, but should not cause errors when solving the problem. In the worst case, additional encapsulated histories will be instantiated resulting in valid but irrelevant equations for the problem-solver to consider. Wrong and failed answers are errors, as they provide the rest of the model formulation process with incorrect information.

Results

As the number of trials varies by the size of the training set and the number of entities per trial depends on the problems in the test set, each condition has a different number of total participant abstraction decisions. Therefore, we report the frequency of each decision type as a percentage of the total decisions made in Table 1.

These results support our hypothesis that our method is able to learn to make participant abstraction decisions.

Training Set Size (# of decisions)	2 (680)	4 (2040)	6 (1360)	8 (170)
Correct	72%	74%	75%	76%
Correctly Ignored	17%	16%	15%	14%
Extraneous	6%	7%	8%	8%
Wrong	4%	2%	1%	.5%
Failed	0%	0%	0%	0%

Table 1: Participant abstraction decision results

With two worked solutions, the system made successful inferences (correct + correctly ignored) 89% of the time, extraneous inferences 6% and incorrect inferences (wrong + failed) 4% of the time. Furthermore, these results support the learning hypothesis because the number of incorrect decisions decreases down to 0.5% as the number of worked solutions in the training set increases to eight.

Figure 6 contains a graph of our method's mean confidence values for each of the inference categories. Correctly ignored and failed decisions always have a confidence value of 0; consequently, they are not shown. The confidence values are a useful discriminator. The correct answer values are significantly different from the irrelevant and wrong answers ($p < .001$). Additionally, our method's confidence values for correct classifications is significantly higher ($p < .001$) with eight worked solutions than with two, supporting our learning hypothesis.

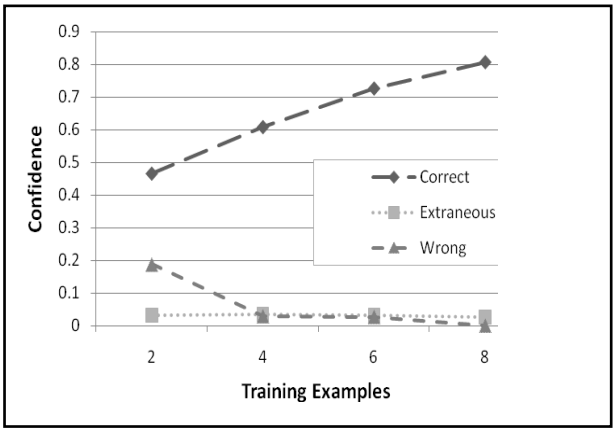


Figure 6: Confidence by answer type and number of examples

Discussion

These results indicate that our method is effective for making participant abstraction modeling decisions. Our method not only makes these decisions, but also returns a confidence score, permitting additional reflection during the model formulation process. Furthermore, our method has a learning component, such that its decisions and confidence estimates improve with experience. We can explain these results by noting that SEQL generalizations abstract away the aspects of the exemplars that are not shared across scenarios. As such, this focuses the participant abstraction decision on the appropriate

```
Generalization:
  (Automobile :genent0)
  (primaryObjectMoving :genent1 :genent0)
  (valueOf
   (AtFn (Acceleration :genent0) :genent1)
   (MetersPerSecondPerSecond :genent2))
  (abstractionForObject :genent0 PointMass)
Exemplar:
  (PassengerAirplane Jet-2-19-P)
  (TurbojetPropelledAircraft Jet-2-19-P)
  (primaryObjectMoving TakeOff-2-19-P Jet-2-19-P)
  (objectStationary
    (StartFn TakeOff-2-19-P) Jet-2-19-P)
  (valueOf  (AtFn (Speed Jet-2-19-P)
                  (EndFn TakeOff-2-19-P))
            (MetersPerSecond 80))
  (querySentenceOfQuery Gia-Query-2-19-P
   (valueOf (AtFn (Acceleration Jet-2-19-P)
                  TakeOff-2-19-P)
          Acceleration-2-19-P))
  (abstractionForObject Jet-2-19-P PointMass)
```

Figure 7: Generalization abstracts away unnecessary facts, highlighting important relations

relational structure in the problem representation. The few failure cases within our results are due to extraneous relational structure within the generalization contexts because the system has not seen enough examples.

This behavior is more evident when we compare a generalization and an exemplar from the experiment, both of which are illustrated in Figure 7. This generalization contains four facts and three generalized entities. `:genents` are entities that have been abstracted by SEQL. The generalization contains an entity that is an automobile, which is the primary object moving in some event with a known acceleration. On the other hand, the exemplar contains seven concrete facts about a jet plane taking off, some of which may complicate the modeling decision. For example, the `objectStationary` fact provides distracting relational structure that could align erroneously with facts in the entity case and result in incorrect modeling decisions. As generalizations are formed from additional examples, however, our method is better able to extract and preserve the relational structure important for making modeling decisions.

Related Work

As noted previously, the majority of model formulation work has focused on ascertaining the levels of detail and perspectives that should be used in a model, given a particular task (cf. Falkenhainer & Forbus 1991; Nayak 1994 Rickel & Porter 1994). A notable exception is Flores and Cerda's (2000) work in analog electronics, which formalized a number of equivalent circuit configurations as rewrite rules to simplify circuit schematics in a human-like way. While these systems perform well in the domains in which they were designed, the goal of this work is to learn how to make modeling decisions in new domains based upon examples. By focusing on learning, we believe our approach will be applicable in a wide variety of domains.

An alternative to these rule-based approaches is analogical model formulation (Klenk *et al.* 2005; Klenk & Forbus 2007). Motivated by the observation that engineers frequently use analogies with their experiences in formulating new models (Falkenhainer 1992), analogical model formulation allows an agent to make a number of modeling decisions about a situation, described in everyday terms, based upon explanations of similar situations. In this work, our method learns how to make participant abstraction decisions via generalization. These generalizations allow the learned knowledge to be applied more generally than in analogical model formulation.

Conclusion & Future Work

This paper presents a method for learning participant abstraction decisions from examples via generalization. We present results from an evaluation in which participant abstraction decisions were learned and applied in the physics domain.

This represents a significant step towards building systems that learn how to model situations from examples. While our results demonstrate the utility of generalizing at the granularity of the model participant decision, we plan on extending this method to other modeling decisions. For example, we plan to explore how generalization could be used to learn situation-appropriate simplifying or operating assumptions (e.g. ignoring friction, laminar flow, or elastic collisions). We also plan to investigate generalization at the level of physical processes or encapsulated histories, perhaps accelerating the model formulation process with experience.

The ability to leverage previously understood domains when faced with new domains is an important frontier for AI research. We plan to incorporate this method for learning domain specific modeling decisions into our Domain Transfer via Analogy (DTA) framework (Klenk & Forbus 2007), which uses multiple cross domain analogies to transfer domain theories between areas of physics. Transferring the modeling knowledge encoded in these generalizations is an important direction for transfer learning research.

References

de Kleer, J. 1977. Multiple representations of knowledge in a mechanics problem solver, pp.299–304. *Proc. IJCAI-77.*

Falkenhainer, B. 1992. Modeling without amnesia: Making experience-sanctioned approximations. *Proceedings of QR02.*

Falkenhainer, B. and Forbus, K. 1991. Compositional modeling: finding the right model for the job. *Artificial Intelligence* 51:95–143.

Falkenhainer, B., Forbus, K. and Gentner, D. 1989. The Structure-Mapping Engine: Algorithm and examples. *Artficial Intelligence.* 41.

Forbus, K. 1984. Qualitative process theory. *Artificial Intelligence*

Flores, J. and Cerda, J. 2000. Efficient modeling of linear circuits to perform qualitative reasoning tasks. *AI Communiations*, 13(2) 125-134.

Gentner, D. 1983. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*.

Giancoli, D. 1991. Physics: Principles with Applications. 3rd Edition. Prentice Hall.

Klenk, M. & Forbus, K. 2007. Learning domain theories via analogical transfer. *Proceedings of Qualitative Reasoning Workshop.* Aberystwyth, UK.

Klenk, M. & Forbus, K. 2007. Measuring the level of transfer learning by an AP physics problem-solver. *Proceedings of Association for the Advancement of Artificial Intelligence (AAAI-07).* Vancouver, Canada.

Klenk, M., K. Forbus, E. Tomai, H. Kim, and B. Kyckelhahn. 2005. Solving everyday physical reasoning problems by analogy using sketches. *Proceedings of the American Association for Artificial Intellegence (AAAI-05).* Pittsburgh, PA.

Kuehne, S.E., Forbus, K.D., Gentner, D., & Quinn, B. (2000). SEQL: Category learning as progressive abstraction using structure mapping. *Proceedings of CogSci 2000*, August.

Matuszek, C., J. Cabral, M. Witbrock, and J. DeOliveria. An Introduction to the Syntax and Content of Cyc. 2006. *Proceedings of AAAI-06 Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering.* Standford, CA.

Nayak, P. 1994. Causal approximations. *Artificial Intelligence* 70:277–334.

Rickel, J. and Porter, B. 1994. Automated modeling for answering prediction questions: selecting the time scale and system boundary, pp. 1191–1198. *Proc. AAAI-94*.

Online Model-Based Diagnosis of Production Systems

Lukas Kuhn, Johan de Kleer
Palo Alto Research Center
3333 Coyote Hill Road, Palo Alto, CA 94304 USA
{lkuhn,dekleer}@parc.com

Abstract

This paper extends model-based diagnosis (MBD) (Reiter 1987; de Kleer & Williams 1987) to systems which convert, move and process material. Examples of such systems are printers, refineries and food processing plants. Such plants present two challenges to model-based diagnosis: (1) the plant may process 100s-1000s of items per minute so retaining full details of behavior of all past objects is impractical, and (2) complex multi-way interactions can occur among components operating on the same object. We address the first challenge by synopsizing past behavior in a data structure of fixed size. We address the second challenge by introducing the notion of interaction fault which represents the situation where a set of components operating on the same object damage the object even though each component alone produces no noticeable damage. Introducing interaction faults is much simpler than introducing fine-grained models of component-object interactions. We demonstrate the approach on a highly redundant printer.

Introduction

Most existing approaches to model-based diagnosis presume all information flow in a system as signals. They are good for modeling systems that can be directly modeled as ODEs such as in is characterized by system dynamics (Shearer, Murphy, & Richardson 1971). However, most real world systems transport and modify materials. For example, a refinery converts one kind of fuel into another with different characteristics, a printer converts blank paper to paper with marks on it, and a General Mills plant converts wheat and cardboard into boxes containing donut-shaped objects (Cheerios). Such systems need to reason about both the attributes of the stuff (e.g., voltage, current, pressure) and their properties (e.g., wrapped candy bar, unwrapped candy bar, partially assembled automobile).

Plants present two challenges to model-based diagnosis: (1) the plant processes 100s-1000s of items per minute so retaining full details of behavior of all past objects is impractical, and (2) complex multi-way interactions can occur among components operating on the same object. This paper outlines an integrated approach to both challenges. First, we synopsize the results in a single fixed-size data structure. Necessarily some information will be lost and we rely on high throughput rates to "make up" for any information so lost. Second, we do not explicitly model the details of component-object-component behavior, but instead introduce a new generic fault category of an interaction fault which specifies some misbehavior has occured, but omitting details on exactly how.

We draw many of our examples from a prototype printer illustrated in Figure 1 ((Fromherz, Bobrow, & de Kleer 2003) provides more background).

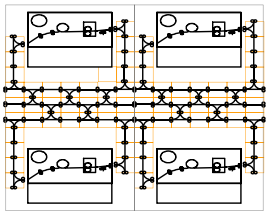


Figure 1: Model of PARC's prototype highly redundant printer. It consists of 4 printers (large rectangles). Sheets enters on the left and exit on the right.

Refineries, printers, manufacturing lines all run continuously. They are expensive to halt so minor problems are ignored or compensated for by later manual processing. Unlike simple qualitative envisionments of one signal propagated through the system, we intend to address a continuous movement with large number of objects being processed at any moment. The closest analog to this type of qualitative reasoning is the parts-of-stuff ontology of (Collins & Forbus 1987). Although more difficult to analyze, continous operation has the advantage that it is possible to gather a great deal of observations quickly and cheaply.

The fact that objects are being processed by the system introduces a whole new set of fault types. For example, we will often see situations where component A operates correctly stand-alone, and component B operates correctly stand-alone, yet fail when they both operate on the same object. Consider a food processing line for candy bars. There are multiple components wrapping and boxing candy bars. It may be that component A leaves a tiny rip which is of no consequence for the consumer, but boxing component B has a small protrusion such that the rip sometimes catches and

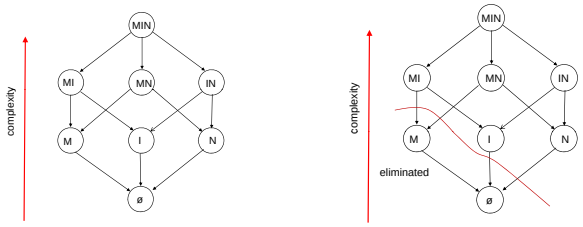
destroys the candy bar. We call such faults interaction faults: A and B are perfectly operational individually but will not work correctly if A and B both process the same candy bar. The classical model-based diagnosis approach would be to consider both components A and B as faulted, but that is not useful for the technician. The line can be restored to full operation by either removing the protrusion on B or repairing A. There is no need to replace both A and B. Such faults occur in digital circuits as well: Gates A and B may not work well together as both may be "late". Replacing either A or B with one having an average gate delay restores the circuit to full functioning.

A technician reasons about a system at multiple levels of abstraction. A technician will make the simplest assumptions possible to diagnose a system and only when those assumptions yield a contradiction will he/she choose a more detailed model. We adopt the meta-diagnosis abstraction framework of (de Kleer 2007). In this approach, the meta-assumptions (of the modeling approach itself) are treated as assumptions in a model-based diagnosis engine. The system always picks one particular diagnosis as the current abstraction level.

Example: Simplest Meta-Diagnosis Failing

For simplicity, we presume that if components are persistent faulty they will always manifest bad behavior. This assumption can also be a meta-assumption, but this makes the examples too complicated.

The three initial meta-assumptions we make are: (1) the system does not have multiple faults "M" (vs. single fault), (2) the fault is not intermittent "I" (vs. persistent), (3) the fault is not interactive "N." This corresponds to the bottom node of Figure 2(a).



(a) M indicates multiple faults; I indicates intermittent faults; N indicates interaction faults

(b) After the minimal conflict $AB_a(I) \vee AB_a(N)$.

Figure 2: Meta-Diagnosis lattice

Consider only the components A, B, C. Suppose we observe the following plans:

time	plan	observation	conclusion
1	A,B	fail	$\neg M$ exonerates C
2	B,C	success	$\neg I$ exonerates B, C
3	A	success	$\neg I$ exonerates A

A plan $p_i = [c_1, c_2, \ldots, c_n]$ is a sequence of components involved in an execution. Plan 1 (A, B) fails, therefore if the system does not contain a multiple fault, one of A or B must be faulted and C cannot be faulted. Plan 2 (B, C) succeeds, therefore given the system is not intermittent B and C must be functioning correctly. Plan 3 (A) succeeds so A cannot be faulted. At this point no single fault, non-intermittent, non-interaction faults exist. This results in the meta-conflict:

$$AB_a(M) \vee AB_a(I) \vee AB_a(N).$$

Analysis must consider retracting one of these three meta-assumptions. Consider multiple faults. Plan 2 exonerates B, C and plan 3 exonerates A with no dependence on the single fault assumption. Therefore, the meta-conflict is:

$$AB_a(I) \vee AB_a(N).$$

Figure 2(b) illustrates the resulting meta-diagnosis lattice.

The system can contain either an intermittent fault or an interaction fault. For example, component A can be intermittently failing, producing a bad output at time 1 and a good output at time 3. The system can also contain an interaction fault. For example, the system can contain the interaction fault $[AB]$. An interaction fault is one in which both components might individually be working correctly, but produce faulty behavior when combined. We use $[...]$ to indicate the interaction fault which occurs only when all of the components operate on the same object. Plan 1 is the only plan in which A and B co-occur, therefore the interaction fault explains all symptoms.

Meta-Inferences

As in conventional model-based diagnosis, a tentative diagnosis is represented by the set of failing components. When a plan p succeeds the following inferences can be drawn:

- If there are no intermittent faults ($\neg AB_a(I)$), then every component mentioned in the plan is exonerated.
- If there are interaction faults ($AB_a(N)$), then every diagnosis containing a interaction fault which contains only components from p is exonerated.

When a plan p fails the following inferences can be drawn:

- Every diagnosis not containing a component in p is exonerated.

Initially, all subsets of components can be diagnoses. With the introduction of interaction faults, any combination of components can also be a fault. Therefore, if a system consists of n components, there are $O(2^{2^n})$ possible diagnoses (Eiter & Gottlob 1995).

Figure 3 shows a fraction of the diagnosis lattice for a simple system with components three components: $\{A, B, C\}$. For simplicity we assume non-intermittent faults, but multiple and interaction faults are allowed. Consider the prior example again. Plan 1 which used A, B produced a failure. By the preceeding rules, C alone cannot explain the symptom, neither can $[AC]$, $[BC]$ or $[ABC]$. The only minimum-cardinality diagnoses are $\{A\}$, $\{B\}$ and $\{[AB]\}$. The successful plan 2 exonerates B and C. Therefore any diagnosis which contains B or C is exonerated. In addition, any diagnosis containing the interaction fault $[BC]$ is exonerated. Finally, when Plan 3 is observed to succeed, A is exonerated.

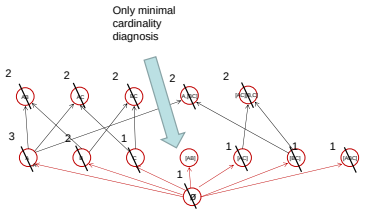


Figure 3: Fragment of diagnosis lattice for the simple 3 component system A, B, C. Includes multiple and interaction faults. The numbers indicate which plan eliminates that diagnosis.

The only minimum cardinality diagnosis which explains the symptoms is the interaction fault $[AB]$.

The very large size of this diagnosis lattice prompts a new diagnostic algorithm more akin to what a technician would use when diagnosing the system. It is also much more efficient for on-line diagnosis.

Diagnostic Algorithm

In this section we present a new diagnostic algorithm which differs from classical model-based diagnostic algorithms in significant ways. The new diagnostic algorithm maintains set of mutually exclusive sets: diagnostic foci, good components, bad components and unknown components. Intuitively, each diagnostic focus represents a set within which we are sure there is a fault. Technicians will typically explore one focus at a time. As the printer or manufacturing line runs continuously there are far too many observations to record in detail. Therefore, the current foci together with the set of bad components and unknown components combined with a fixed size buffer will represent the *entire* state of knowledge of the faultedness of system components. Some information from prior observations will be discarded. The algorithm we describe may take more observations to pinpoint the true fault(s), but it will never miss faults.

Multiple Faults Case

In what follows we discuss an algorithm for diagnosing multiple simultaneous faults. Our algorithm allows variable amount of observation data (which can be obtained throughout execution of plans) to be retained.

A **system** Sys is a tuple $< C, P, Z >$ where:

- C is the set of all components.
- P is a list of plans. A plan $p_i = [c_1, c_2, \ldots, c_n]$ is a sequence of components involved in the plan.
- Z is a list of observations. A observation $z_i \in \{f, s\}$ is associated with plan p_i. We denote a plan failure as f and a normal plan execution as s.

A **state of knowledge** SK is a tuple $< g, b, x, DF >$ where:

- $g \subseteq C$ is the set of good components.
- $b \subseteq C$ is the set of bad components.
- $x \subseteq C$ is the set of unknown components which are not under suspicion.
- DF is the set of diagnosis foci. A diagnosis focus $df_i \subseteq C$ is a set of suspected components with at least one faulted component in it.

Algorithm 1: Multiple (Interaction) Faults Algorithm with Memory

```
foreach p_j : P do
    if !multipleFaults(p_j, z_j) then
        memorize(p_j, z_j, memorysize);
    else
        evaluateMemorizedPlans();
```

Algorithm 1 executes Procedure 2 (or for the interaction fault case Procedure 4) for each plan and observation pair. The algorithm updates the *entire* state of knowledge of the faultedness of system components. We focus on high throughput systems (100s-1000s/min) and therefore the algorithm we describe may take more observations to pinpoint the true fault(s), but it will never miss faults. We include a memory extension to mitigate the loss of diagnosis information. There are two cases in which the evaluation of an observation could lead to information loss: (1) two intersecting plans fail due to different faults or, (2) a failing plan intersects two diagnosis foci. In the first case we might not know at evaluation time if two intersecting plans fail because of the same fault or two different faults and therefore we keep the plan to later re-evaluate it. In the second case we can not extract any information before we reduce the diagnostic foci until the failing plan intersects only one diagnosis foci. Note that this might not be possible. Failing plans of either case can be helpful if they are re-evaluated later. Note that we are able to configure the memory size to address memory limitations.

Let Sys be a simple system with five components $C = \{A, B, C, D, E\}$. Again we assume that we are able to execute any combination of components as a plan. Suppose component B and D are faulted.

In Table 1 we show for each time step t the *entire* state of knowledge.

Again note that every component will be a member of exactly one of the sets of the current SK. Consider the sequence of plans illustrated by Table 1. Plan 1 $(ABCDE)$ fails. Therefore we focus on the fact that one of $\{A, B, C, D, E\}$ is faulted. Plan 2 (ABC) fails. Therefore, we narrow the focus to the fact that one of $\{A, B, C\}$ is faulted, and we don't know anything about $\{D, E\}$. Plan 3 (ADE) fails. Therefore the focus narrows to A, while there may be a fault in $\{B, C\}$ (But the scope is still $\{A, B, C\}$). Plan 4 (A) succeeds. Therefore, A is exonerated. At this point we backtrack and move the focus to $\{B, C\}$. Plan 5 (ADE) fails. Therefore, given that A is exonerated, we can introduce a new focus on the fact that one of $\{D, E\}$ is faulted. Plan 6 (AC) succeeds. Therefore, C is exonerated and B is the only component left in focus 1. Therefore we know B is faulted. We close focus 1. Plan 7 (ADC) fails. Therefore, given that A, C are exonerated, D is faulted. We close focus 2 and move the remaining components (here E)

Function `multipleFaults` (*plan* p_j, *obs* z_j)

```
if z_j == f then
    rp_j = p_j − g;
    if rp_j ∩ b = ∅ then
        if |rp_j| == 1 then
            b = b ∪ rp_j;
            foreach df_i : DF do
                if df_i ∩ rp_j ≠ ∅ then
                    x = (x ∪ df_i) − b;
                    DF.remove(df_i);
        else
            if rp_j ∩ ⋃_k df_k = ∅ then
                df_new = rp_j;
                x = x − rp_j;
            else
                foreach df_i : DF do
                    if rp_j ∩ df_i ≠ ∅ ∧ rp_j ≠ df_i then
                        if rp_j − df_i ⊆ x then
                            if |rp_j| < |df_i| then
                                x = (x ∪ df_i) − rp_j;
                                df_i = rp_j;
                            else
                                return false;
                        else
                            return false;
else
    g = g ∪ p_j;
    x = x − g;
    foreach df_i : DF do
        df_i = df_i − g;
        if |df_i| == 1 then
            b = b ∪ df_i;
return true;
```

in the unknown set. Plan 8 (ACE) succeeds, thus ACE is exonerated.

Multiple Interaction Faults Case

Definition: Let $X = \{x_1, \ldots, x_n\}$ be a set of elements.

- $P(X)$ is the power set over X, e.g. $X = \{x_1, x_2\} \leftrightarrow P(X) = \{\{\}, \{x_1\}, \{x_2\}, \{x_1, x_2\}\}$.
- $\overline{X} \equiv P(X)$ represents the power set of X.
- $\{\overline{Y}\} \sqcup \{\overline{X}\} \equiv \begin{cases} \{\overline{Y}\} & : \text{if } X \subseteq Y \\ \{\overline{X}\} & : \text{if } Y \subseteq X \\ \{\overline{Y}, \overline{X}\} & : \text{otherwise} \end{cases}$
- $\overline{P}(X) \equiv \bigcup_{Y \subseteq X} \overline{Y}$ is the set of all power sets over all possible subsets of X .
- $E(X)$ is the set of all individual components mentioned in X, e.g. $X = \{\{a, b, c\}, \{a, d, e\}, \{g\}\} \leftrightarrow E(X) = \{a, b, c, d, e, g\}$.

A **system** Sys is a tuple $< C, P, Z >$ as defined in the multiple fault case.

A **state of knowledge** SK is a tuple $< g, b, x, DF >$ where:

- $g \subseteq \overline{P}(C)$ represents all global good diagnosis candidates. A diagnosis candidate is a set of components that can cause a failure. Let $X \subseteq C$ be a set of components, than $\overline{X} \in \overline{P}(C)$ represents all diagnosis candidates $dc \in P(X)$.
- $b \subseteq P(C)$ is the set of bad diagnosis candidates. $\{A, [DE]\}$ denotes that A and the diagnosis candidate $[DE]$ (interaction fault) are bad.
- $x \subseteq P(C)$ is the set of unknown diagnosis candidates which are not under suspicion.
- DF is the set of diagnosis foci. A diagnosis focus df_i is a tuple $< su_i, lg_i >$ where:
  - $su_i \subseteq P(C)$ is the set of suspected diagnosis candidates in the diagnosis focus df_i.
  - $lg_i \subseteq \overline{P}(C)$ represents all local (relevant) good diagnosis candidates.

t	p	z	g	b	x	df_1	df_2
0					ABCDE		
1	ABCDE	f				ABCDE	
2	ABC	f			DE	ABC	
3	ADE	f			DE	ABC	
4	A	s	A			BC	
5	ADE	f	A			BC	DE
6	AC	s	AC	B			DE
7	ADC	f	AC	BD	E		
8	ACE	s	ACE	BD			

Table 1: System with five components $C = \{A, B, C, D, E\}$ where B and D are faulted.

Consider the following example. Let Sys be a simple system with five components $C = \{A, B, C, D, E\}$. Suppose component B and D are faulted. In Table 2 we show walk through the example.

t	p	z	g	b	df_1		df_2	
					su_1	lg_1	su_2	lg_2
1	ABCDE	f			ABCDE			
2	ABC	f			ABC			
3	ADE	f			ABC			
4	A	s	$\overline{A}$		BC	$\overline{A}$		
5	ADE	f	$\overline{A}$		BC	$\overline{A}$	DE	$\overline{A}$
6	AC	s	$\overline{AC}$		B	$\overline{AC}$	DE	$\overline{A}$
7	ADC	f	$\overline{AC}$	D	B	$\overline{AC}$		
8	ACE	s	$\overline{ACE}$	D	B	$\overline{AC}$		
9	B	s	$\overline{ACE}, \overline{B}$	D	[AB][BC]	$\overline{AC}, \overline{B}$		
10	AB	f	$\overline{ACE}, \overline{B}$	[AB],D				

Table 2: System with five components $C = \{A, B, C, D, E\}$ where $[AB]$ and D are faulted.

Consider the sequence of plans illustrated by Table 2. Plan 1 $(ABCDE)$ fails. Therefore we focus on the fact that one of $\{A, B, C, D, E\}$ is faulted. Plan 2 (ABC) fails. Therefore, we can narrow the focus to the fact that one of $\{A, B, C\}$ is faulted, and we don't know anything about $\{D, E\}$. Plan 3 (ADE) fails. Therefore the focus narrows to A, while there may be a fault in $\{B, C\}$ (But the scope is still $\{A, B, C\}$). Plan 4 (A) succeeds. Therefore, A is exonerated. At this point we backtrack, move the focus to $\{B, C\}$ and keep A as a local (relevant) good ($\{\overline{A}\}$). The scope is now $\{B, C\}$. Plan 5 (ADE) fails. Therefore, given that A is exonerated, we can introduce a new focus on $\{D, E\}$, but we keep A as a local (relevant) good ($\{\overline{A}\}$). Plan 6 (AC) succeeds. Therefore, A, C, AC are exonerated, denoted as $\overline{AC}$.

The new global goods are $\{\overline{AC}\}$, because $\{\overline{A}\} \sqcup \{\overline{AC}\} = \{\overline{AC}\}$. We update the local (relevant) goods in focus 1 to AC, because A, C, AC are relevant to focus 1. B is the only diagnosis candidate left in focus 1. Plan 7 (ADC)fails. Therefore, given that A, C are exonerated, D is faulted, because it is a minimal diagnosis candidate. We close focus 2. Plan 8 (ACE)succeeds, thus A, C, E, AC, AE, CE, ACE are exonerated, denoted as $\overline{ACE}$. The new global goods are $\{\overline{ACE}\}$, because $\{\overline{AC}\} \sqcup \{\overline{ACE}\} = \{\overline{ACE}\}$. Plan 9 (B) succeeds, thus B is exonerated, denoted as $\overline{B}$. The new global goods are $\{\overline{ACE}, \overline{B}\}$, because $\{\overline{ACE}\} \sqcup \{\overline{B}\} = \{\overline{ACE}, \overline{B}\}$. At this point we know that the diagnosis candidates A, B, C, AC relevant to focus 1 are goods. Therefore generate all minimal diagnosis candidates form the local goods $\{[AB], [BC]\}$ and move the focus to them. Plan 10 (AB)fails. Therefore, given that A, B are exonerated, $[AB]$ is faulted, because it is a minimal diagnosis candidate.

Function *multiInteractFaults(plan p_j, z_j)* descripes the algorithm for multiple interaction faults in more detail.

Functioncandidates and minimalCandidates

```
minimalCandidates(Set<Comps> C, P̄(Set<Comps>) PC)
Beginn
    CA = candidates(C, PC);
    minCar = |E(CA)|;
    MCA = ∅;
    foreach ca_i : CA do
        if |ca_i| = minCar then
            MCA = MCA ∪ ca_i;
        if |ca_i| < minCar then
            MCA = ∅;
            MCA = MCA ∪ ca_i;
    return MCA;
Ende
candidates(Set<Comps> C, P̄(Set<Comps>) PC)
Beginn
    CA = P(C);
    foreach pc_i : PC do
        CA = CA − pc_i;
    return CA;
Ende
```

Conclusions

This paper is a first step towards an integrated qualitative diagnostic approach to systems which process material such as manufacturing lines and printers. It presents a novel algorithm for diagnosing multiple interaction faults which is far more memory efficient than the traditional model-based algorithms. The overall approach is similar to how technicans address troubleshooting.

References

Collins, J. W., and Forbus, K. D. 1987. Reasoning about fluids via molecular collections. In *AAAI*, 590–594.

de Kleer, J., and Williams, B. C. 1987. Diagnosing multiple faults. *Artificial Intelligence* 32(1):97–130. Also in: *Readings in NonMonotonic Reasoning*, edited by Matthew L. Ginsberg, (Morgan Kaufmann, 1987), 280–297.

de Kleer, J. 2007. Modeling when connections are the problem. In *Proc 20th IJCAI*, 311–317.

Eiter, T., and Gottlob, G. 1995. The complexity of logic-based abduction. *Journal of the ACM* 42(1):3–42.

Fromherz, M.; Bobrow, D.; and de Kleer, J. 2003. Model-based computing for design and control of reconfigurable systems. *The AI Magazine* 24(4):120–130.

Reiter, R. 1987. A theory of diagnosis from first principles. *Artificial Intelligence* 32(1):57–96.

Shearer, J. L.; Murphy, A. T.; and Richardson, H. H. 1971. *Introduction to System Dynamics*. Reading, MA: Addison Wesley.

Function multiInteractFaults (*plan p_j, obs z_j*)

```
if z_j == f then
    CA_j = candidates(p_j, g);
    if CA_j ∩ b == ∅ then
        if |CA_j| == 1 then
            b = b ∪ CA_j;
            foreach df_i : DF do
                if CA_j ∩ su_i ≠ ∅ then
                    DF.remove(df_i);
        else
            MCA_j = minimalCandidates(p_j, g);
            if CA_j ∩ ⋃_k su_k == ∅ then
                foreach c ∈ g do
                    lg_new = lg_new ⊔ overline(E(c) ∩ p_j);
                su_new = MCA_j;
            else
                foreach df_i : DF do
                    if MCA_j ∩ su_i ≠ ∅ ∧ MCA_j ≠ su_i then
                        if MCA_j ∩ ⋃_{k,k≠i} su_k == ∅ then
                            if |MCA_j| < |su_i| then
                                foreach c ∈ g do
                                    lg_i = lg_i ⊔ overline(E(c) ∩ p_j);
                                su_i = MCA_j;
                            else
                                return false;
                        else
                            return false;
else
    g = g ⊔ overline(p_j);
    foreach df_i : DF do
        lg_i = lg_i ⊔ overline(E(lg_i) ∪ E(su_i)) ∩ p_j;
        su_i = minimalCandidates(su_i, lg_i);
        if |su_i| == 1 then
            CA_{lg_i} = candidates(E(lg_i), lg_i);
            if |CA_j| == 1 then
                b = b ∪ CA_j;
                foreach df_i : DF do
                    if CA_j ∩ su_i ≠ ∅ then
                        DF.remove(df_i);
            else
                su_i = minimalCandidates(E(lg_i), lg_i);
return true;
```

Supporting Conceptual Knowledge Capture Through Automatic Modelling: A Preliminary Progress Report

Jochem Liem and **Hylke Buisman** and **Bert Bredeweg**
Human Computer Studies Laboratory, Informatics Institute, Faculty of Science,
University of Amsterdam, The Netherlands. Email: {jliem,bredeweg}@science.uva.nl, hbuisman@gmail.com

Abstract

Building qualitative models is still a difficult and lengthy endeavour for domain experts. This paper discusses progress towards an automated modelling algorithm that learns Garp3 models based on a full qualitative description of the system's behaviour. In contrast with other approaches (Bridewell et al. 2008; Bratko and Šuc 2004), our algorithm attempts to learn the causality that explains the system's behaviour. The algorithm achieves good results when recreating four well-established models.

Introduction

In this paper we focus on the ground work required to advance towards an automated modelling program. The input is considered to have a qualitative representation, i.e. a state graph that represents the possible situations that can emerge from a system, and the values of the quantities in each situation. Furthermore, the input is assumed to have no noise nor any inconsistencies. The completed algorithm is envisioned to support researchers in articulating their conceptual understanding. As such it will help to establish theories that explain the phenomena provided as input data.

QR Model and Simulation Workbench: Garp3

The automatic model building algorithm is implemented in Garp3[1] (Bredeweg et al. 2006). Garp3 allows modellers to represent their knowledge about the structure and the important processes in their system as *model fragments*, which can be considered formalisations of the knowledge that applies in certain general situations.

Next to model fragments, different *scenarios* can be modelled. These represent specific start states of a system. Garp3 can run simulations of models based on a particular scenario. The result of such a simulation is a state graph, in which each state represents a particular possible situation of the system, and the transitions represent the possible ways a situation can change into another.

The simulation engine takes a scenario as input, and finds all the model fragments that apply to that scenario. The consequences of the matching model fragments are added to the scenario to create a state description from which new knowledge can be inferred such as the derivatives of quantities. Given the completed state description, the possible successor states are inferred. The complete state graph is generated by applying the reasoning to the new states.

In Garp3 the structure of a system is represented using *entities* (objects) and *configurations* (relations). For example, a lion hunting on a zebra would be represented as two entities (lion and zebra) and a configuration (hunts).

Quantities represent the features of entities and agents that change during simulation. A quantity has a magnitude and a derivative, which represent its current value and trend. The magnitude and derivative are each defined by a quantity space that represents the possible values the magnitude and the derivative can have. Such a quantity space is defined by a set of alternating *point* and *interval* values.

We use $M_v(Q_1)$ to refer to the current value of the magnitude of a quantity. $M_s(Q_1)$, the sign of the magnitude, indicates whether the magnitude is positive, zero or negative ($M_s(Q_1) \in \{+, 0, -\}$). $D_v(Q_1)$ refers to the current value of the derivative of a quantity, which has a value from the predefined derivative quantity space ($D_v(Q_1) \in \{-, 0, +\}$). $D_s(Q_1)$ refers to the current sign of a derivative. Note that the predefined values of derivatives completely correspond to the possible signs of the derivative.

Causality

Garp3 explicitly represents causality using indirect and direct influences. Direct influences are represented as $Q_1 \overset{I_+}{\rightarrow} Q_2$. Influences can be either positive (as above) or negative. The positive influence will increase $D_v(Q_2)$ if $M_s(Q1) = +$, decrease it if $M_s(Q1) = -$, and have no effect when $M_s(Q1) = 0$. For a negative influence, it is vice versa.

The indirect influences, called *proportionalities*, are represented as $Q_1 \overset{P_+}{\rightarrow} Q2$. Similar to influences, proportionalities can be either positive or negative. The positive proportionality will increase $D_v(Q_2)$ if $D_s(Q_1) = +$, have no effect if it is stable, and decrease if it is below zero. For a negative proportionality, it is vice versa.

Other Behavioural Ingredients

Other behavioural ingredients in Garp3 are operators, inequalities, value assignments and correspondences. Opera-

[1] http://www.garp3.org

tors (+ and -) are used to calculate the magnitude value of quantities (e.g. $Q_1 - Q_2 = Q_3$, to indicate $M_v(Q_1) - M_v(Q_2) = M_v(Q_3)$). Inequalities can be placed between different model ingredient types: (1) magnitudes ($M_v(Q_1) = M_v(Q_2)$), (2) derivatives ($D_v(Q_1) < D_v(Q_2)$, (3) values $Q_1(point(Max)) = Q_2(point(Max))$, (4) operator relations ($M_v(Q_1) - M_v(Q_2) < M_v(Q_3) - M_v(Q_4)$, (5) combinations of the 1, 2, 3 and 4 (although only between either magnitude or derivative items). Value assignments simply indicate that a quantity has a certain qualitative value ($M_v(Q_1) = Q_1(Plus)$). Finally, correspondences indicate that from certain values of one quantity, values of another quantity can be inferred. There are quantity correspondences ($Q_1 \overset{Q_{qs}}{\leftrightarrow} Q_2$) and value correspondences ($Q_1(Plus) \overset{Q_v}{\rightarrow} Q_2(Plus)$), which can both be either directed or undirected. The value correspondence indicates that if $M_v(Q_1) = Q_1(Plus)$, $M_v(Q_2) = Q_2(Plus)$. If the value correspondence is bidirectional, the reverse inference is also possible. Quantity correspondences can be considered a set of value correspondences between each consecutive pair of the values of both quantities. There are also inverse quantity space correspondences ($Q_1 \overset{Q_{qs}^{-1}}{\leftrightarrow} Q_2$) that indicate that the first value in Q_1 corresponds to the last value in Q_2, the second to the one before last, etc.

Algorithm Requirements and Approach

Assumptions and Scoping

The goal of the automatic model building algorithm is to take a state graph and a scenario as input, and generate the model that provides an explanation for the behaviour. Our approach focusses on the generation of causal explanation. Several assumptions are made to scope the work. In further research these assumptions can be alleviated. Firstly, input is assumed to have no noise or inconsistencies. Secondly, the state graph is assumes to be a full envisionment of the system's behaviour.

The second assumption is that a model can be build using a single model fragment. From a causal explanation point of view, it is reasonable to assume that influences and proportionalities never disappear, but that their effects are only nullified when quantities become zero or stable.

Thirdly, the algorithm is focussed on causal explanation and less on structure. Therefore, the entity hierarchy is assumed known.

Input and Output

The algorithm takes a complete state graph as input, which includes (1) the quantity names, (2) the quantity spaces, (3) the magnitudes and derivatives of the quantities in different states, (4) the observable inequalities, and (5) the state transitions. Furthermore, the algorithm is provided with the scenario that should produce the state graph, which consists of: (1) the entities, agents and assumptions involved, (2) structural information about the configurations between them, (3) the quantities and their initial values, and (4) the inequalities that hold in the initial state.

The output of the algorithm is one or more Garp3 qualitative models that explain (are consistent with) the input that can be immediately simulated.

Algorithm Design Approach

Since the semantics of model ingredients are formally defined, one would assume that it is clear how each ingredient manifests itself in the simulation results of a model. Otherwise, how would the implementation of a simulation engine have been possible? However, in practice, it is hard even for expert modellers to pinpoint the model ingredients that are responsible for certain (lack of) behaviour. This has several reasons. Firstly, a large set of inequalities are derived during qualitative simulation, of which the implications (other inequalities) are difficult to foresee. Secondly, the engine has a lot of intricacies (such as second order derivatives) which makes simulation results hard to predict. Thirdly, the branching in the state graph that results from ambiguity is difficult for people to completely envision.

For these reasons, an iterative algorithm design approach is chosen. Well-established models are ordered by complexity, and attempts are made to generate them using their own output. Each of the models requires a different (and increasingly large) set of considerations that must be dealt with.

The models chosen are Tree and Shade, Communicating Vessels, Deforestation, Population Dynamics and a set of other even more complex models[2]. Tree and Shade is the least complex model, containing only a few quantities, and causal dependencies, and no conditions, causal interactions, inequalities or operator relations. Communicating vessels is more complex, as it contains causal interactions, an operator, and inequalities. The deforestation model is different from the previous models as it contains many clusters linked to each other by proportionalities. Population dynamics is again more complex, due to the large amount of quantities, interactions and conditions.

Causality and Clusters

Causal Paths Important for the algorithm is the concept of *causal paths*. These are series of quantities connected by influences and proportionalities. A causal path is defined as a set of quantities that starts with an influence, and is followed by a arbitrary number of proportionalities. For example: $Q_1 \overset{I_+}{\rightarrow} Q_2 \overset{P_+}{\rightarrow} \ldots \overset{P_-}{\rightarrow} Q_{n-1} \overset{P_+}{\rightarrow} Q_n$. A quantity that has no proportionalities leading out of it ends the causal path. If a quantity has more than one proportionality leading out of it, multiple causal paths can be defined.

Since each influence represents the causal effect of a process, a causal path can be seen as the cascade of effects of a process. Given this perspective, certain successions of causal relations become unlikely. For example the causal path $Q_1 \overset{I_+}{\rightarrow} Q_2 \overset{I_+}{\rightarrow} Q_3 \overset{P_-}{\rightarrow} Q_4 \overset{I_+}{\rightarrow} Q_5$ would imply there are many active processes with short or no cascading effects.

Direction of Causality An important issue in scientific enquiry is the problem of correlation and causality. This

[2]The models are available at http://www.garp3.org

issue appears when trying to derive causal relations from the state graph. For example, $D_s(Q_1) = D_s(Q_2)$ can be an caused by $Q_1 \overset{P_+}{\rightarrow} Q_2$, $Q_2 \overset{P_+}{\rightarrow} Q1$, or even $Q_3 \overset{P_+}{\rightarrow} Q_1$ and $Q_3 \overset{P_+}{\rightarrow} Q_2$. Another example of this is in the communicating vessels model. Ideally, a model capturing the idea of a contained liquid would distinguish between Volume, Height and Bottom pressure, and have a particular causal account ($Volume \overset{P_+}{\rightarrow} Height \overset{P_+}{\rightarrow} Bottom\_pressure$). However, from the model's behaviour this causality may not be derivable, e.g. when the width of the containers doesn't change. As a result, the unique role of the quantities involved can only be inferred when the required variation for that is apparent in the input state-graph. Therefore, it is considered the modeller's responsibility to provide simulation examples which will allow the algorithm the make these critical distinctions. However, it can be considered the responsibility of the tool to indicate to the modeller that the causality between certain sets of quantities cannot be derived, and that examples showing these differences should be provided.

Clusters The algorithm makes use of a specific subset of causal paths called *clusters*. We define clusters as groups of quantities that exhibit "equivalent" behaviour. More specifically, a set of quantities constitute a cluster if their values either correspond ($Q_1 \overset{Q_{qs}}{\leftrightarrow} Q_2$) or inversely correspond ($Q_1 \overset{Q_{qs}^{-1}}{\leftrightarrow} Q_2$) to each other. Additionally, the corresponding derivatives should be equal ($D_v(Q_1) = D_v(Q_2)$), while inversely corresponding derivatives should be each other's inverse ($D_v(Q_1) = -D_v(Q_2)$).

A further constraint is that the corresponding quantities (not inverse) in a cluster must be completely equivalent. Therefore, $M_v(Q_1) = M_v(Q_2)$ must always hold. If an inequality holds between two quantities, they are considered not to belong to the same cluster.

During implementation it became obvious that clusters are not meaningful when quantities within a cluster belong to different entities. The reason for this originates from the idea of 'no function in structure'. Clusters involving multiple entities would integrate causality across individual structural units, which is undesired. Therefore, clusters can only contain quantities that belong to the same entity.

Quantities cannot be a member of more than one cluster. If Q_1 and Q_2 are in a cluster, and Q_1 and Q_3 are in a cluster, then Q_1, Q_2 and Q_3 must be in the same cluster. After all, if Q_1 and Q_2 have equivalent behaviour, and Q_1 and Q_3 have equivalent behaviour, by transitivity Q_2 and Q_3 have to exhibit equivalent behaviour.

Minimal Covering

The key requirement of the model building algorithm is that it explains the input behaviour. However, a second requirement is that the algorithm does not contain redundant dependencies. That is, the algorithm should return the minimal set of dependencies that explains the behaviour.

Two dependencies are considered *substitutionary* if they have the same effect on the simulation result (i.e. removing one of them would have no effect, however removing both would). *Complementary* dependencies are responsible for different aspects of the behaviour, and both have to be present to explain the data. The aim is to create an algorithm that is minimally covering, i.e. it should only contain complementary dependencies.

Algorithm

Finding Naive Dependencies

The goal of this step is to find (non-interacting) dependencies that are valid throughout the entire model (i.e. are not conditional). These causal relations are called *naive dependencies*, and provide the basis for the rest of the algorithm.

Consistency Rules Naive dependencies are identified using consistency rules. Each pair of quantities is checked using these rules to determine which of them potentially holds throughout the state graph. These rules make use of $M_v(Q_x), M_s(Q_x), D_v(Q_x), D_s(Q_x)$ of each quantity in a pair, and inequalities that hold between them. These statements are referred to as the *state information* of a quantity.

The consistency rules are derived from the semantics of the causal dependencies (see Section on Garp3). Examples of rules (that should hold throughout the state graph) are:

$$Q_1 \overset{I_+}{\rightarrow} Q_2 \text{ if } M_s(Q_1) = D_s(Q_2) \tag{1}$$

$$Q_1 \overset{I_-}{\rightarrow} Q_2 \text{ if } M_s(Q_1) = -D_s(Q_2) \tag{2}$$

$$Q_1 \overset{P_+}{\rightarrow} Q_2 \text{ if } D_s(Q_1) = D_s(Q_2) \tag{3}$$

$$Q_1 \overset{P_-}{\rightarrow} Q_2 \text{ if } D_s(Q_1) = -D_s(Q_2) \tag{4}$$

$$Q_1(V_x) \overset{Q_v}{\leftrightarrow} Q_2(V_y) \text{ if } M_v(Q_1) = Q_1(V_x) \implies M_v(Q_2) = Q_2(V_y) \tag{5}$$

$$Q_1 \overset{Q_{qs}}{\leftrightarrow} Q_2 \text{ if } \forall V_n (Q_1(V_n) \overset{Q_v}{\leftrightarrow} Q_2(V_n)) \tag{6}$$

Redundancy The set of dependencies that are found contain a lot of redundancy, i.e. many dependencies are substitutionary. For example, in the communicating vessels model $height \overset{P_+}{\rightarrow} pressure$, can be substituted by $pressure \overset{P_+}{\rightarrow} height$. The remainder of the algorithm selects the correct substitutionary groups, and uses the selected naive dependencies to derive more complex dependencies.

Determining Clusters

This step tries to determine clusters within the set of naive dependencies. The algorithm searches for quantities belonging to the same entity that exhibit equivalent behaviour, and tries to expand these candidate clusters by adding other quantities. Quantities are only added if they exhibit behaviour equivalent to the quantities already contained in the candidate cluster. If no more quantities can be added to a candidate cluster, the algorithm searches for other candidate clusters. By only considering models composed of clusters, the space of possible models is significantly reduced.

The validity of the candidate clusters is checked by determining if there is overlap between the clusters. All clusters that overlap are removed. An alternative would be to only remove clusters until no more overlap is present. However,

in practice no situations were encountered where this was desirable. An example of a found cluster is volume, height and pressure in the communicating vessels model. Note that these clusters are still missing influences (their actuators), these are determined later in the algorithm.

Generating Causal Paths

This step returns the possible causal orderings within clusters based on the cluster and naive dependencies sets. For each cluster a valid causal ordering is returned. Through backtracking other possible orderings are generated.

The quantities in a cluster can be either connected in a linear fashion ($Q_1 \overset{P_+}{\rightarrow} Q_2 \overset{P_+}{\rightarrow} Q3$) or using branching ($Q_1 \overset{P_+}{\rightarrow} Q_2$ and $Q_1 \overset{P_+}{\rightarrow} Q3$). The algorithm prefers linear branching, as branching does not often occur in practice. Additionally, the reduction of possible models is a significant advantage.

Another constraint that reduces the number of possible models is requiring clusters that belong to entities of the same type to have the same causal ordering. For example, if for one container $Volume \overset{P_+}{\rightarrow} Height \overset{P_+}{\rightarrow} Pressure$, than for other containers the same causal ordering must hold.

Actuating Clusters

The goal of the actuating clusters step is to connect clusters by identifying cluster actuations. This step takes the set of clusters with established causal orderings and the naive dependencies as input.

Clusters can either be actuated by another cluster, or act as an actuator itself. Furthermore, clusters can be connected by propagating an actuation. In a model, each cluster should take part in at least one of these kind of relations such that all clusters are related in a way. Otherwise, the model would include two separate non-interacting subsystems.

When one cluster actuates another, there is an influence relation between the two. Actuations are the most important form of connecting clusters, since these connections are the cause of change in the system. They are also the easiest to detect, due to the specific way influences manifest themselves in the state information. For this reason, actuations by influences are identified first. Two types of actuations though influences are distinguished: (1) *equilibrium seeking mechanisms* (ESM) and (2) *external actuators*.

Equilibrium Seeking Mechanisms ESMs are better known as *flows*, and are common in qualitative models. Flows cause two unequal quantities to equalize. The flow in the communicating vessels model has a non-zero value when the pressures in the two containers are unequal. The flow changes the volume of the containers, and thus the pressures to equalize. An ESM holds under the following two conditions: (1) $Q_1 = Q_2 - Q_3$, where $Q_1 \in C_1, Q_2 \in C_2, Q_3 \in C_3$, where the C's are clusters, and (2) $Q_4 \overset{I_-}{\rightarrow} Q_5$ and $Q_4 \overset{I_+}{\rightarrow} Q_6$, where $Q_4 \in C_1, Q_5 \in C_2, Q_6 \in C_3$. Note that in many cases $Q_1 = Q_4$, such as in the communicating vessels model.

Finding Calculus Relations The algorithm reduces the search space of finding ESMs using four constraints. Firstly, all quantities involved in the operator should be in different clusters (C_1, C_2 and $C3$ are unequal). Secondly, the set of naive dependencies should at least contain one influence from Q_1 (to serve as an actuation). Thirdly, both Q_2 and Q_3 would be at the end of the causal paths within their cluster, as in most cases this is the most meaningful interpretation. Finally, Q_2 and Q_3 are required to be of the same type, as only things of the same type can be subtracted.

External Actuators External actuators are causes of change more at the edges of the system compared to ESMs. To identify external actuators, the algorithm considers the influences in the naive dependencies that are not part of an ESM. Again, the minimal covering principle is applied to keep the number of dependencies to a minimum. As a result a cluster will never have more than one incoming actuation.

An actuation is only considered between C_1 to C_2 if the set of naive dependencies contains influences between each possible pair of quantities, such that $\forall Q_x \in C_1, \forall Q_y \in C_2 (Q_x \overset{I_+}{\rightarrow} Q_y)$. This removes the influences in the set of naive dependencies that are consistent with the behaviour by chance.

Alternative actuations are returned through backtracking. In the future, actuations may be chosen based on the structure of the system, as causal relations are more likely to occur parallel to structurally related entities.

Feedback A common pattern in qualitative models is feedback, which is a proportionality originating from the end of a causal path to the quantity actuating the causal path. Feedbacks are simply added if the naive dependencies contain one. The algorithm always adds feedback at the end of causal paths, since this is what happens in the investigated models. However, it could be the case that feedbacks from halfway a causal chain are also possible.

Linking Clusters by Propagation

This step connects the clusters that have not yet been connected through proportionalities, based on the naive dependencies. As with clusters, the causal ordering of the clusters cannot be distinguished. Therefore all possibilities are generated. Furthermore, the same design choices as with finding causal paths within clusters have been made. Only linear orderings of clusters are allowed (i.e. no branching).

Setting Initial Magnitudes

An influence has no effect if the magnitude of the quantity from which it originates is unknown. Therefore this step assigns initial values to quantities. Note that this step first generates a set of candidate assignments. When a value can be derived in another way than through assignment, it is removed from the set of value assignment candidates.

There are six ways to assign initial magnitudes. Firstly, if a value assignment for the quantity is present in the scenario, it requires no initialisation. Secondly, if the magnitude can be derived through a correspondence, the value is known. Thirdly, the result of a minus operator can be derived if an

inequality between its arguments is known. Based on the possible magnitudes of the result this inequality can be derived. Either this inequality is present in the scenario, or multiple inequalities should be made assumable by adding them as conditions in multiple model fragments. Garp3 automatically assumes unprovable values and inequalities if they are conditions in model fragments. Note that generating the conditional inequalities is currently beyond the scope of the algorithm, as it involves adding model ingredients to multiple model fragments. Fourthly, it is possible that a certain magnitude holds everywhere throughout the state graph. In this case, a value assignment is added as a (conditionless) consequence. Fifthly, a value could hold under certain conditions. However, this would require a value assignments with a conditional inequalities in separate model fragments. Therefore, it is currently beyond the scope of the algorithm. Finally, multiple model fragments could be created in which the magnitudes are present as conditions. Garp3 will generate the different states that would result by assuming each of the values. As with the conditional value assignments, having value assignments as conditions in multiple model fragments is currently beyond the scope of the algorithm.

Dependency Interactions

This step identifies dependency interactions (influences or proportionalities) based on the input behaviour. Dependency interactions are detected in the same way as naive dependencies, i.e. using a set of consistency rules. Interactions are not found as naive dependencies, as the individual dependencies are not consistent with the *entire* state graph (as an interaction results in more behaviour than a single dependency).

The algorithm assumes that the interaction consists opposing dependencies, such as birth vs. death and immigration vs. emigration.

Results[3]

The *tree and shade model* is successfully modelled by the algorithm. It returns two models, representing both possible directions of causality between Size and Shade. The initial magnitude assignment correctly finds a conditionless value assignment on Growth rate. The simulation results of these models are equivalent to that of the original model.

The dependencies of the *communicating vessels model* are correctly found. The algorithm returns 6 models; one for each possible causal ordering of amount, height and pressure. The algorithm also correctly identifies the ESM-based actuations of the clusters, by properly finding the min operator. Furthermore, all necessary causal dependencies and correspondences are identified. Model fragments that allow the assumption of initial values are missing (due to the fact that the algorithm generates a single model fragments). Adding an inequality between the pressures of the containers in the scenario allows the model to simulate without problems.

The *deforestation model* (containing entities 'Woodcutters', 'Vegetation', 'Water', 'Land' and 'Humans') is successfully modelled, including setting initial magnitudes using conditions. The simulation is equivalent to that of the original model. The causal ordering does differ, as it does not capture the branching of the causal paths in the original model. The resulting model however, is not considered wrong by experts, and is arguably better than the original. Over 2000 models are returned when generating all possible results, due to the many possible causal orderings.

The *population dynamics model* generates the correct models for the open and closed population scenarios. However, the initial values are not set.

The algorithm does not yet give correct results for the *heating/boiling*, *R-Star* and *Ants' Garden* models. For the heating model this is due to inequalities that hold under specific conditions, which are not taken care of in the algorithm. The R-Star and Ants' Garden are large models that resulted from specific research projects. As such, these models are an order of magnitude more complex than the other models. It is therefore not surprising that the algorithm in its current form cannot cope with them.

[3]For the models and references go to http://www.garp3.org

Conclusions & Future Work

This paper presents preliminary work towards an algorithm that automatically determines a Garp3 qualitative model, using an enumeration of all possible system behaviour as input. The algorithm uses consistency rules to determine the causal dependencies that hold within the system. Using the concept of clusters the search space is significantly reduced. Accurate results are generated for a set of well-established models. The results seem to suggests that it is possible to derive causal explanations from the behaviour of a system, and that model building support through an automatic model building algorithm is viable.

There are several algorithm improvements planned. The first improvement is to have a generalised representation for the ambiguity within and between clusters. That is, have a single representation for the complete model space. For simulation purposes an arbitrary instantiation can be chosen, as each one has an equivalent result. Secondly, the algorithm has to be improved to be able to create multiple model fragments in order to deal with conditional model ingredients. Thirdly, means have to be developed to be able to compare generated state graphs with the desired state graph.

References

Bratko, I., and Šuc, D. 2004. Learning qualitative models. *AI Mag.* 24(4):107–119.

Bredeweg, B.; Bouwer, A.; Jellema, J.; Bertels, D.; Linnebank, F.; and Liem, J. 2006. Garp3 - a new workbench for qualitative reasoning and modelling. In Bailey-Kellogg, C., and Kuipers, B., eds., *20th International Workshop on Qualitative Reasoning (QR-06)*, 21–28.

Bredeweg, B.; Salles, P.; Bouwer, A.; Liem, J.; Nuttle, T.; Cioaca, E.; Nakova, E.; Noble, R.; Rios Caldas, A. L.; Yordan, U.; Varadinova, E.; and Zitek, A. 2008. Towards a structured approach to building qualitative reasoning models and simulations. *Ecological Informatics* 3(1):1–12.

Bridewell, W.; Langley, P.; Todorovski, L.; and Džeroski, S. 2008. Inductive process modeling. *Machine Learning* 71:132.

A Theory of Depiction for Sketches of Physical Systems

Kate Lockwood, Andrew Lovett, Ken Forbus, Morteza Dehghani and Jeff Usher

Northwestern University, Qualitative Reasoning Group
2145 Sheridan Road Room L359, Evanston Illinois
{kate, andrew-lovett, forbus, morteza, usher}@northwestern.edu

Abstract

Complex spatial and physical concepts are often communicated using diagrams. For many qualitative reasoning tasks, it is necessary that computers understand diagrams in much the same way as their human collaborators. Here we describe some preliminary work on basic diagram interpretation based on common depiction conventions. Using a combination of semantic and qualitative spatial information we are able to distinguish relevant regions and edges in sketched diagrams using the CogSketch sketch understanding system.

Introduction

Complex physical concepts are often communicated using a combination of text and diagrams. Often the diagram illustrates the physical arrangement of a system and accompanying text or captions describe a process that happens in that system. For example, consider Figure 1 below taken from *Sun Up Sun Down* (Buckly, 1979) an introductory solar energy text.

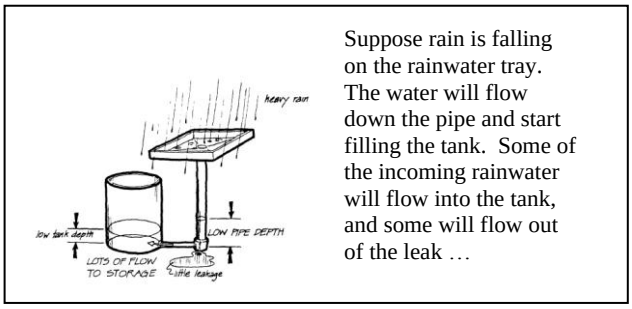


Figure 1. An example of a diagram and accompanying text from a solar energy textbook

The caption provides a description of a process, the filling of a tank with rainwater, while the diagram provides an illustration of the physical layout of the system (tank, pipe, etc).

Parsing the qualitative information in the diagram is easy for people, but quite complicated for software. For example people can both recognize the diagram as a full system and refer to its individual components like "the water in the tank". Part of this flexibility is due to our familiarity with diagrams and their depiction conventions. Another source of flexibility is our knowledge about how things like tanks and pipes and water work. We are able to leverage both types of knowledge when looking at a diagram.

To make intelligent systems that can reason and communicate using diagrams, they must understand diagrams in ways similar to their human users. Additionally, systems need to be able to understand not just polished textbook diagrams, but incomplete, messy user-sketched diagrams. For intelligent conceptual design aids and intelligent tutoring systems, being able to understand informally sketched concepts is paramount.

Traditional diagram and sketch understanding work is far from exhibiting the human-like flexibility needed in a diagram understanding system. Most sketch systems are focused on recognition. First, they segment images into lines and then combine the lines into larger objects. Objects are then matched against a library of known shapes, with the best match considered as the interpretation for the object. Such systems can work well in tightly constrained domains that have a small number of distinct symbols such as family trees (Alvarado & Davis, 2004), circuit and diagrams (Alvarado, Oltmans, & Davis, 2002), x,y plots (Futrelle, 1990) and maps (Reiter & Mackworth, 1989). Unfortunately, they do not translate well to diagrams like Figure 1.

In this paper we are proposing a more integrated approach to diagram interpretation using CogSketch, an open-domain sketch understanding system. In our model, we use a combination of semantic information about the objects and qualitative spatial relationships between them to infer relevant depiction conventions. Specifically, our goal is to correctly assign concepts to *edges* and *regions* in a diagram, consistent with depiction conventions.

The CogSketch approach is based on two insights: (1) In most human-to-human sketching, recognition is a catalyst, not a requirement. People use language to explain their sketches; we provide interface tools for providing functionally similar ways to *conceptually label* glyphs in a sketch. (2) Many of the conceptually relevant relationships in sketches are *qualitative*. For example, in the diagram in Figure 1 the specific details of the objects depicted does not matter, what matters is the qualitative relationships: how the objects are connected, what the level of the water is in the tank is relative to the placement of the leak, etc. Our approach is to model human visual and geometric processing of the ink in a sketch, combined with formal representations of conceptual knowledge drawn from a large-scale knowledge base, to provide *open-domain* sketch understanding abilities. This is very important for building intelligent systems for open-ended,

domains, such as engineering design, where the set of possible objects is extremely broad.

The rest of this paper describes our method for modeling this flexible interpretation of depiction conventions within CogSketch. First we review CogSketch. Next we describe the spatial extent problem. Then we describe how we combine semantic and geometric information to interpret a sketch, using a detailed example. We finish with related work and future work.

Sketching

All diagrams are sketched using CogSketch[1]. CogSketch is an open-domain sketch understanding system built on the nuSketch architecture (Forbus, Ferguson, & Usher, 2001). In CogSketch, each object drawn is represented by a *glyph*. A glyph contains both the actual ink drawn by the user and a *conceptual label.* The conceptual label is supplied by the user and is tied to a concept in the underlying knowledge base. Currently we are using a subset of the ResearchCyc[2] knowledge base (including over 30,000 concepts). Users can also supply a name with which to refer to the glyph. Names can be any natural language string. For example, Figure 2 shows a screenshot of a diagram drawn in CogSketch. In this diagram, the cylinder in the sketch is labeled as a `WaterTank` using the concept from ResearchCyc and is named "tank". This allows the user to refer to the tank simply as "tank". Likewise, if there were multiple tanks, they could each be given different identifying names. In CogSketch, users determine what ink belongs to a glyph by clicking a button at the beginning and end of drawing each glyph. All the ink drawn between button presses is part of the glyph.

Conceptual labeling allows CogSketch to truly be domain-independent and allows us to operate in domains without clear drawing conventions. All sketch understanding work must strike a balance between constraints on the user and the depth of interpretation that is possible. While labeling glyphs does require more work by the user, in return they gain freedom from recognition errors and the ability to be supported by more in-depth reasoning. Aside from manual segmentation, we place no other restrictions on how users draw each glyph. For example, they can use as many strokes as they like, connected or not, and can take as long as they like. This contrasts with a common practice in multimodal interfaces of using constraints such as time-outs and pen-up events to automatically infer segmentation. For our users, who are often thinking hard about what they are drawing, time-outs and pen-up constraints are poor segmentation signals and quite annoying to them.

CogSketch computes a variety of spatial relationships automatically, including the RCC-8 qualitative topology (Cohn, 1996) relationships and connected and contained groups of glyphs (see (Forbus, Tomai, & Usher, 2003) for details). The digital ink itself is also available in subsequent processing, re-sampled into constant-spaced intervals from the original time-stamped pen events.

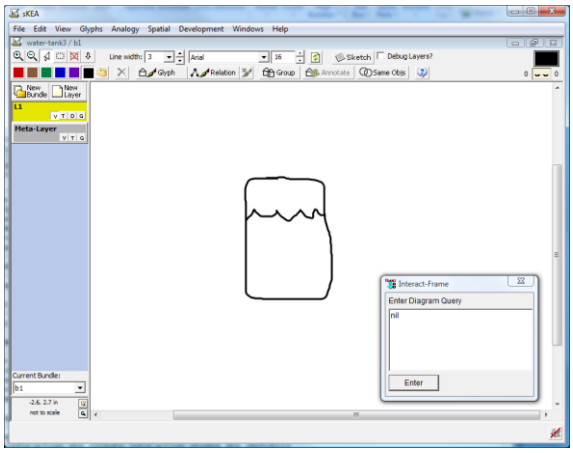


Figure 2. A screen shot of CogSketch showing a sketch of a tank of water.

Conceptual Segmentation

We define the task of *conceptual segmentation* to be the assignment of conceptual interpretations to regions and edges within the sketch. As noted above, conceptual labeling of ink is necessary, but not sufficient, for solving this problem. Consider the sketch in Figure 2 above showing a tank filled with water. We will use this example as an illustration throughout this paper. This sketch consists of two glyphs: one closed polygon representing the tank, and one line representing the water. Figure 3 illustrates.

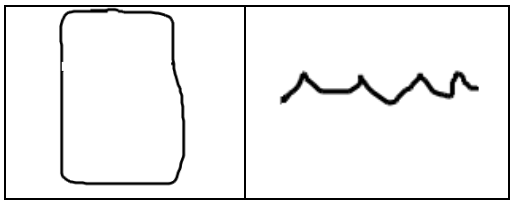


Figure 3. The two glyphs that make up the sketch in Figure 2. The glyph on the left is the tank and the glyph on the right is water.

If we simply use the conceptual labels, the system would think that the object water in the sketch was the edge created by the water glyph when in fact it is the area

1 CogSketch is publicly available at http://spatiallearning.org/projects/cogsketch_index.html
2 http://research.cyc.com/

inside the tank underneath the water glyph. The situation gets even more complex in a sketch like that in Figure 4 below. Here again, the water glyph is a single line, but this line is discontinuous and spans to different tank glyphs. The system would also need to infer that the pipe (another individual glyph) is also filled with water even though the pipe glyph does not touch the glyph representing the water.

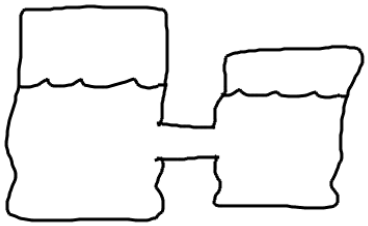


Figure 4. A two tank system as sketched in CogSketch

One way to address this would be to require users to draw following specific conventions – for example, have them trace around the inside of all of the tank/pipe glyphs so that the water glyph was one continuous closed shape. However, while we could institute that constraint, it only addresses this specific situation, and adding new constraints to address every new situation is untenable. Additionally, requiring users to trace the full outline of the water still leaves the situation ambiguous. The system still doesn't have a way to figure out if the user intended just the outline to represent water, or all of the space contained by the outline. For example, consider the two sketches in Figure 5 below.

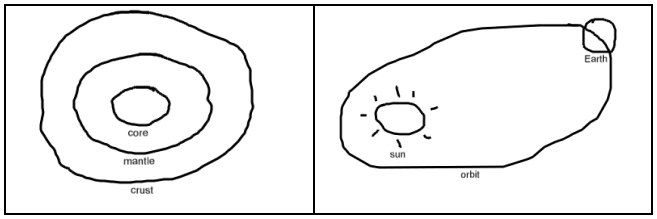


Figure 5. Two sketches, one of the layers of the Earth and another of a planet orbiting the sun.

Both sketches contain an outer ellipse. In the sketch on the left it represents the crust of the earth, and in the sketch on the right it represents the orbit of Earth around the sun. The interpretation for the two ellipses is different. In the sketch on the left the convention is that everything between the outer ellipse and the next ellipse is the stuff that makes up the crust. By contrast, in the sketch on the right, the orbit is actually just the edge represented by the glyph itself. This is why we need a combination of semantic and geometric information in order to make a correct interpretation. There are two parts to our interpretation process - the gathering of semantic information and the segmentation of the image.

We are interested in using the fact that we know what we are drawing and we know about how things are typically drawn – *depiction conventions* – to automatically derive the correct conceptual segmentation of the sketch. We test its segmentations by asking it to highlight the region or edge in a sketch representing a specific entity. If the correct area is highlighted, we conclude that the system has correctly interpreted that portion of the sketch.

Using semantic information for depiction reasoning

Once the appropriate glyph is identified, we access the conceptual label(s) provided by the user. In our example, the glyph being considered is labeled with the concept `Water` from the ResearchCyc KB. Knowing what the glyph represents helps us figure out how to interpret the diagram correctly. For example, ResearchCyc has 335 facts about water. This includes information about its role in the ResearchCyc ontology and, especially important for our purposes, some linguistic knowledge about the term.

Backchaining rules are used to ascertain whether a concept needs a region versus a polyline to depict it. For example, a concept might contain information that, linguistically, the word referring to it is a mass noun or a count noun. Mass nouns refer to entities that can be viewed as spatially flexible pieces of stuff, such as liquids and powders, whose boundaries are highly constrained by containment relationships. The concept `Water` is linguistically a mass noun, and consequently the system infers that it requires a volume to depict it.

Figure 6 below shows an outline of the process we use to determine the correct depiction for a glyph using both the conceptual label and the ink. This figure shows the algorithm as it is currently implemented, as we expand the number and type of diagrams that we interpret, the algorithm will be further refined. In the first step, the conceptual label is accessed and the knowledge base is queried to determine which category the entity belongs to: (1) a mass noun or entity that subclasses from the Cyc concept `TangibleStuffCompositionType` (2) an entity that subclasses from `Path-Spatial` (3) or a physical object.

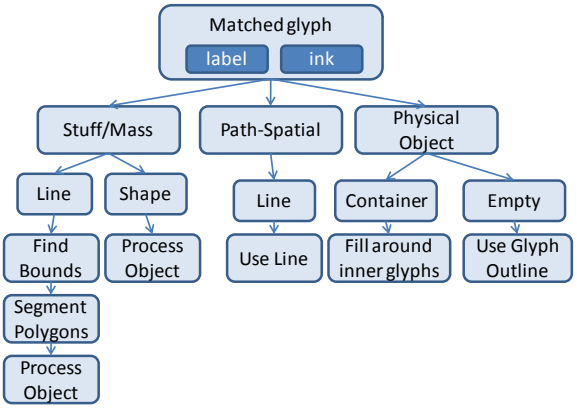


Figure 6. Outline of the spatial extent identification algorithm as it is currently implemented

Inferring the geometry of depiction

Once the system has inferred the conceptual category for a glyph, it attempts to find or construct the appropriate geometric entity. For the water/tank example (an instance of the stuff/mass path through Figure 6) it starts by classifying the geometric properties of the ink for the glyph, determining if it is a line or a region. For example, the glyph representing water in Figure 5 is a polyline, not a region. Since the depiction of water requires a region, the system has more work to do. If the user had drawn the water by tracing out a region inside the tank, then the system would be satisfied with the glyph itself as the geometric entity.

The next step is to determine if there are other glyphs which can help constrain the extent of the object. In this example, the tank glyph constrains the extent. We find such glyphs by looking for RCC8 relationships, i.e., glyphs for which the water is either TPP or NTPP (i.e., Tangential Proper Part or Non-Tangential Proper Part). When these relationships hold, between the tank glyph and the water glyph, we then do a follow-up check to see if the water intersects (within a threshold) both sides of the tank.

Once we have both glyphs (the water and the tank) we need to find the region representing the part of the tank where the water is found. This is accomplished by combining the ink from the two glyphs and segmenting the ink into edges and *edge cycles*. Edges are identified by segmenting the ink at places where one line intersects another, or where there is a clear corner along a line. Edge cycles are identified by finding minimal closed cycles among the edges. In the current example, CogSketch identifies two edge cycles, one representing the area in the tank above the water and the other representing the area in the tank below the water.

For stuff/mass nouns, the system assumes the user has drawn the uppermost edge of the object, and that the object descends from there to fill the container below it. Thus, in the current example, the system looks for a cycle such that glyph for water overlaps with the top of the cycle, while the rest of the cycle is made up of points from the tank glyph. If an appropriate cycle is found, it is identified as the region that the user is looking for, and it is then converted to a polygon and processed like a physical object.

Physical objects (the third path in Figure 6) are checked to see if they contain other glyphs (containment is one of the spatial relationships computed automatically by CogSketch). If the glyph has other objects inside of it, the algorithm as currently implemented assumes that the correct segmentation for the glyph is the space around the inner objects. This is the correct interpretation for situations like the layers of the earth, or bubbles in soda. Figure 7 shows the results of the query "mantle" in a sketch of the layers of the Earth.

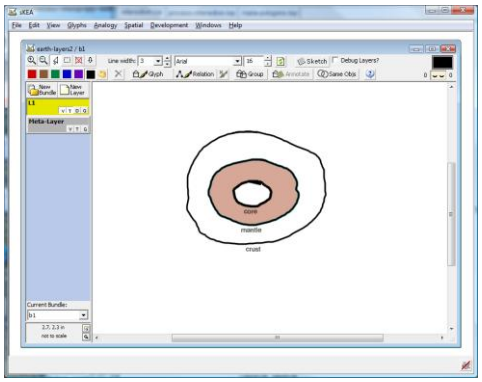


Figure 7. The results from the user query "mantle" in a sketch of the layers of the Earth.

If a physical object has no interior glyphs, the whole area of the glyph is considered the correct depiction and it is highlighted in the diagram. Figure 8 shows the results of our system on the water and tank example when queried for "water".

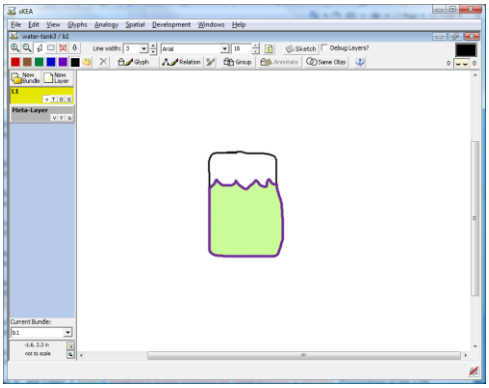


Figure 8. A screen shot of CogSketch showing the results from the user query "water". The shaded are represents the region that the system infers is water.

This approach easily extends to other, more complex situations. In Figure 9 the sketch is composed of four glyphs: tank1 (the tank on the left), a pipe, tank2 (the tank on the right), and one glyph representing the water. Since our algorithm for locating cycles of edges is flexible enough to find cycles over multiple glyphs, the two tank problem is easily handled.

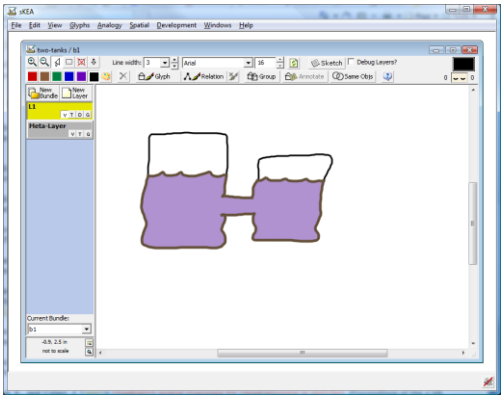


Figure 9. Another result for the query "water". In this case the sketch contains four separate glyphs {tank1, tank2, pipe, water}.

We are also able to handle situations where there are several glyphs that are conceptually labeled as mass nouns, even if they are drawn similarly. In Figure 10 the sketch is a tank with both oil and water in it. When queried for "oil" our system is able to easily identify the extent of the area representing oil. Situations like this would be particularly tricky for template based systems since both oil and water are drawn with similar glyphs. Also, while the wavy line is typical of a convention used to indicate liquid in a sketch, it is by no means a standardized symbol.

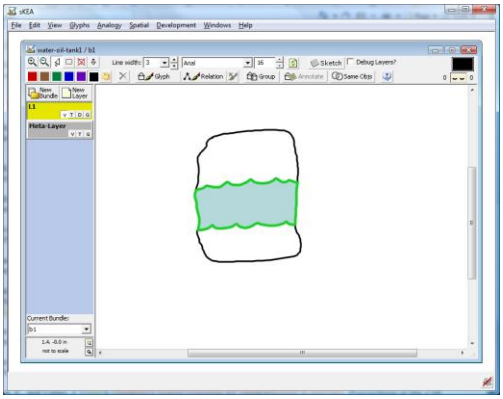


Figure 10. In this example, the system is able to easily discriminate between the region representing "oil" and that representing "water" using the same techniques.

The current algorithm for physical objects has been sufficient for all of the diagrams that we have considered in this paper, however, when a glyph is a container it isn't always the case that you want just the space around the interior glyphs. For example, consider a glass of water with a straw in it. When you are determining the spatial extent of the water, it actually also covers the area occupied by the straw. We are extending the spatial extent algorithm to account for situations like this by further examining the objects in container/contained groups. This is another example where we will need to combine conceptual information from the KB with spatial information from the ink to identify the correct spatial extent.

The processing for an entity that has been determined to be an instance of a Path-Spatial proceeds much like the processing of a mass noun, by first checking to see how the object is drawn in the sketch. Here we will refer back to the solar system/orbit example from Figure 5. In this case, the system checks to see if the path is represented by a single line, like the orbit in the sketch. This suggests that the points on the line make up the conceptual entity. The other option, of course, is that a path is depicted by multiple lines or polygons such as a drawing of a railroad track or road. This condition is not currently being handled by our system, but is in the process of being added.

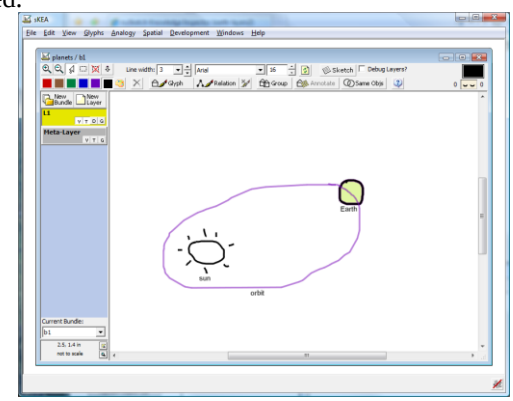


Figure 11. A screenshot showing the spatial extent identified for "orbit" and "Earth" in a simplified drawing of the solar system. Even though both objects are drawn similarly, conceptual information provides clues as to their different interpretations.

Compound Queries

Often the parts of a diagram that need to be referred to are more complex than just "water". For example, when doing problems in physics or chemistry, it may be useful to be able to refer to the water in one part of the apparatus only. Our system also handles queries of the form *<object> <relation> <object>*. Information about

relations from ResearchCyc is used to understand the semantics of such queries. Figure 12 illustrates the result for the query "water in tank1". The analysis is essentially that of Figure 9 with the additional specification of "in tank1" leading to the intersection of the water polygon and the tank1 polygon.

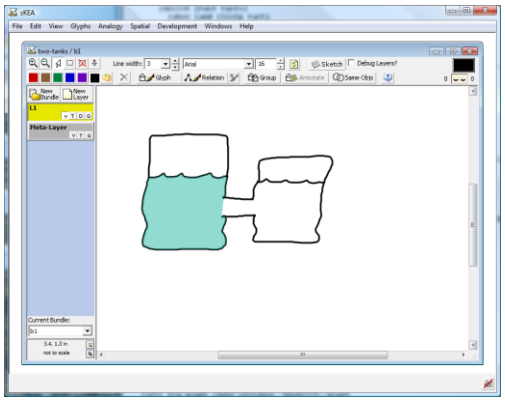


Figure 12. Screenshot showing the result of the query "water in tank1"

Related Work

The division of scene elements into edges and regions in sketches was explored in the Mapsee program of Reiter and Mackworth (Reiter & Mackworth, 1989). They proposed a logical framework for depiction that formalized the mapping between images and scenes of simple maps containing roads, rivers, shores (represented as edges in the images) and water and land (represented by regions in the images). They identified a set of six visual relations ({tee, chi, bounds, closed, interior, and exterior}) and provided axioms and constraints which combined these visual primitives and mapped them to the scene elements (roads, rivers, etc). Like Mapsee, we are concerned with modeling how conceptual entities are depicted. However, Mapsee was designed for one domain, i.e., maps, and its axioms map visual elements directly to interpretations in that domain. By contrast, our model works through an intermediate distinction – regions versus edges – and performs reasoning over a large-scale, off-the-shelf knowledge base to identify depiction constraints. Their task was fundamentally one of image interpretation, recognizing unlabelled lines as map elements, whereas our task starts with conceptually labeled ink.

Alvarado and colleagues (Alvarado & Davis, 2004; Alvarado, Oltmans, & Davis, 2002) describe a multi-domain sketch recognition engine. Their systems use a hierarchical shape description language where low level shape description (circles, arrows, etc) are defined once in a domain-independent fashion. Then a separate set of rules ties a given shape to a domain specific interpretation (e.g. an arrow represents a child link in a family tree diagram). This approach can work well in a very tightly constrained domain with a small number of differentiated symbols (family trees, circuit diagrams, etc.) Unfortunately, it does scale to the more open-domain, unconstrained types of sketches that we are concerned with.

There could be advantages to incorporating some carefully restricted low-level shape recognition to our depiction reasoning, to identify common elements (e.g., arrows). For example, in a physics system, it might be useful to automatically recognize arrows and interpret them as forces while leaving the types of objects that those forces can act on unconstrained given the wide variety of physical objects in the world.

We believe that recognition is not very important for the sketch understanding tasks we are focused on. Unlike sketches in engineering design, where later versions will need to be imported to a formal CAD system, the sketches produced for student assessments are meant to be short lived. Also, while the amount of detail can vary greatly, much of it is superfluous to the pedagogical goals of the assignment and is not important for the overall interpretation of student understanding.

Conclusions and Future Work

We have described how to use a combination of semantic and geometric information to identify one type of depiction convention in sketched diagrams. Our interpretation process closely couples semantic and geometric information to reason about depiction conventions and to use those conventions to segment the sketch into meaningful regions and edges.

Our work on depiction conventions is motivated by several projects. Creating a platform for sketch-enabled educational software is one of our long-range goals. Another is the use of sketches in multimodal knowledge capture. For example, diagrams in educational materials are accompanied by explanatory text. We are creating a system that learns from sketched diagrams plus accompanying simplified English text. Being able to correctly interpret how entities in the diagram are depicted is essential for integrating knowledge across modalities.

We are also interested in studying depiction conventions which are widely used, but not domain or situation dependent. For example, call-outs and cut-aways are two conventions that are used across disciplines which have important implications for how diagrams (and the spatial relations in them) should be interpreted.

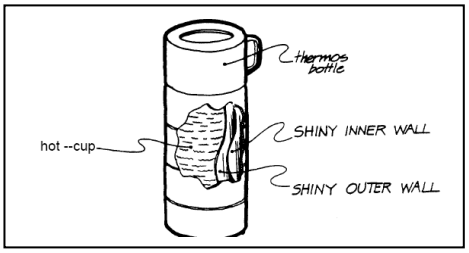


Figure 13. Example of a cut-away in a diagram

CogSketch is free and available online (the online version comes bundled with OpenCyc, as opposed to ResearchCyc which was used for this work). As more people download and use CogSketch, we are hoping to amass a large library of sketches. This library will enable us to more thoroughly survey the conventions used in sketched diagrams. It will also provide a corpus of labeled sketches that we hope will be useful to us and to others in the sketch understanding community.

Acknowledgements

This work was supported by a grant from the Office of Naval Research and by the National Science Foundation under Grant No. SBE-0541957, The Spatial Intelligence and Learning Center.

References

Alvarado, C., Davis, R. (2004). Sketchread: a multi-domain sketch recognition engine. *Proceedings of the 17th annual acm symposium on user interface software and technology*.

Alvarado, C., Oltmans, M., Davis, R. (2002). A framework for multi-domain sketch recognition. *Proceedings of aaai spring symposium on sketch understanding*.

Buckley, S. (1979). *Sun Up to Sun Down*. New York: McGraw Hill.

Cohn, A. (1996) Calculi for Qualitative Spatial Reasoning. In *Artificial Intelligence and Symbolic Mathematical Computation*, LNCS 1138, eds: J Calmet, J A Campbell, J Pfalzgraph, Springer Verlag, 124-143.

Forbus, K., Ferguson, R., & Usher, J. (2001). Towards a computational model of sketching. *IUI'01*. January 14-17, 2001. Santa Fe, New Mexico.

Forbus, K., Tomai, E., and Usher, J. (2003). Qualitative spatial reasoning for visual grouping in sketches. *Proceedings of the 17th International Workshop on Qualitative Reasoning*, Brasilia, Brazil, August.

Futrelle, R. P. (1990). Strategies for Diagram Understanding: Object/Spatial Data Structures, Animate Vision and Generalized Equivalence. In *10th ICPR* (pp. 403-408): IEEE Press.

Reiter, R. and Mackworth, A.K. (1989). A Logical Framework for Depiction and Image Interpretation. *Artificial Intelligence*, 41, 125-155.

Building and Comparing Qualitative Descriptions of Three-Dimensional Design Sketches

Andrew Lovett **Morteza Dehghani** **Kenneth Forbus**
{andrew-lovett@, morteza@cs., forbus@}northwestern.edu
Qualitative Reasoning Group, Northwestern University

Abstract

We describe a method for constructing qualitative structural descriptions of hand-drawn sketches of 3D objects. We use visual grouping and segmentation operations to extract edges and surfaces, and use line labeling with an extension of Malik's (1987) junction catalog to identify three-dimensional features in order to construct an orientation-invariant symbolic representation. These symbolic representations can be used to identify corresponding surfaces and edges in two different sketches drawn in different perspectives of the same object. The comparison process uses the Structure-Mapping Engine, with additional sketch-specific matching constraints. We evaluate our techniques with a sketch recognition task, using drawings of 12 objects from an engineering design textbook.

Introduction

Representing and reasoning about human-drawn sketches presents an interesting problem for AI. Sketches are a promising input modality for intelligent systems: people can often draw an object or spatial layout more easily than they can describe it. However, every person's drawing style is different, and most of us are not skilled artists. This makes accurate interpretation of sketches a difficult problem. We have argued that one key to reasoning about sketches intelligently is the use of qualitative spatial representations (Forbus et al., 2001). The detailed, quantitative description of ink is laden with accidental information, whereas a qualitative representation of key features can concisely summarize the information that was meant to be conveyed.

Qualitative representations are particularly important for tasks where sketches are compared. For example, a system comparing two users' sketches of a bucket must contend with differences in width and orientation of the bucket's sides, as well as the presence or absence of water. Qualitative representations of edges and relationships between edges can help a system identify commonalities in the sketches, such as the presence of an ellipse at the top, two straight edges along the sides, and a straight or curved edge at the bottom.

We use CogSketch (Forbus et al., 2008), a publically available sketch understanding system[1], to automatically derive qualitative spatial relations between objects in a sketch, as well as between edges within an object. Sketches are compared using the Structure-Mapping Engine (SME) (Falkenhainer, et al. 1986), a computational model of similarity and analogy. CogSketch and SME have been used together to accomplish several spatial reasoning tasks, including answering geometric Miller Analogy Test questions (Tomai et al., 2005), matching human performance on a subset of the Raven's Progressive Matrices, a visually-based intelligence test (Lovett et al., 2007b), and sketch recognition (Lovett et al., 2007a). In the sketch recognition task, the system was able to recognize sketches of 8 household objects, including a bucket and an oven, after being trained on only 2-6 example sketches of each object. In contrast, sketch recognition systems that rely on quantitative sketch representations often require at least an order of magnitude more training examples (e.g., Liwicki and Knipping, 2004; Sharon and van de Panne, 2006).

One significant limitation of the (Lovett et al., 2007a) system was that it required that all of the sketches of a given object be drawn from the same perspective. Many of the qualitative relations used were orientation-dependent. Even a small rotation of a sketched object in 3D changes the relative positions of the edges and junctions and causes some to become occluded while others become visible, causing significant representation changes.

The key to correctly comparing sketches of objects drawn at different orientations is to identify and encode qualitative relations that remain constant across rotations in space. In this paper, we introduce our approach for constructing orientation-invariant representations of 3D objects. Briefly, we begin by segmenting a sketch into edges and using closures among those edges to identify the surfaces of the object. Edges are then classified via line labeling, using an extension of Malik's (1987) junction catalog. The edge labels tell the system when an edge represents a corner between two surfaces and when it is an edge of one surface occluding the other. With this information, the system is able to construct a qualitative

[1] http://www.spatialintelligence.org/projects/cogsketch_index.html

representation of the spatial relations between edges and surfaces that remains relatively stable across rotations.

We start by briefly reviewing SME. The process of building qualitative sketch representations is explained next, followed by the sketch matching algorithm. We then evaluate the system via a recognition task run on a set of 12 sketched objects from an engineering design textbook. Finally, we summarize related and future work.

Comparison via Analogy

Qualitative representations can be compared using the Structure-Mapping Engine (SME) (Falkenhainer et al. 1986). SME is a cognitive model based on Gentner's (1983) structure-mapping theory of analogy. SME takes as input two descriptions, a base and a target. Each description consists of a set of entities, attributes of entities, and relations. First-order relations directly relate two or more entities, while higher-order relations take other, lower-order relations as their arguments[2]. Given the two descriptions, SME finds one to three mappings between the base and target by aligning their common structure. Structural alignment is governed by the *systematicity* constraint, i.e., SME prefers mappings in which higher-order relations align.

Each mapping returned by SME contains: (1) a set of correspondences, or *match hypotheses* (mh's) between elements (entities, attributes, and relations) in the base and elements in the target. (2) the *structural evaluation score* (SES), a measure of similarity. Mappings with greater systematicity, i.e., mappings in which higher-order relations are aligned, receive a higher SES. (3) *candidate inferences*, inferences carried over from the base to the target based upon their common structure.

The Surface Extraction Algorithm

Surfaces are identified in a sketch via a two-stage process (Figure 1). In the first stage, the rough sketch is segmented into edges. In the second stage, edges are grouped together to form surfaces.

Segmentation

1) Identify junctions between polylines, segment polylines at junctions to form pseudo-edges.
2) Segment pseudo-edges at discontinuities in curvature to form edges.

Grouping

3) Identify edge cycle that bounds the object.
4) Identify surfaces within the object by finding minimal closures of edges.
5) Repeat 3 & 4 for internal edges.

Figure 1. The algorithm for finding surfaces

[2] The notion of order in structure-mapping differs from traditional logic: it concerns depth of structure. Entities have order zero; the order of a statement is one plus the maximum order of its arguments.

Segmentation

The representation system begins with a set of polylines, lists of points representing the lines sketched by the user. It does not assume that each polyline corresponds with one edge in the sketch. Rather, it begins by looking for connections between the polylines (Step 1). If two polyline endpoints are sufficiently close to each other, the polylines will be connected by a junction. If one polyline's endpoint is near the middle of another polyline, the second polyline will be split, and all three will be connected by a junction at the intersection point. If two polylines intersect, they will be segmented into four connected polylines. The system searches for connections iteratively at multiple scales, beginning with a small distance threshold and increasing the threshold for endpoints that fail to connect to anything. At the end of this process, any pairs of polylines that are connected by a junction containing only two polylines are joined together, since it is possible the user meant them to both be part of the same edge. The output of this process is a set of proto-edges, as well as junctions between proto-edges.

Proto-edges are then segmented to form the actual edges of the sketch (Step 2). Possible segmentation points are identified by finding maximal derivates of the curvature of each proto-edge. We follow Lowe's (1989) approach of parameterizing a proto-edge's list of points to form x- and y-functions and convolving each function with a Gaussian and a derivative Gaussian to calculate the curvature at each point along the proto-edge. This allows the system to modify the width of the Gaussian to look for changes in curvature at different scales, depending on the length of the proto-edge. Once segmentation points are identified, they are evaluated by looking at the curvedness and relative orientation of the edge segments on either side of the point.

Grouping

Here connected edges are grouped together in order to identify the surfaces of the sketch. All surfaces except the background possess an exterior, a closed cycle of edges that surrounds them. However, not every cycle of edges corresponds to a surface. Our line labeling algorithm assumes that every edge represents a boundary between surfaces. Therefore, only the minimal closures, the tightest possible cycles, correspond to surfaces in the sketch.

Our system simplifies the process of surface detection and line labeling by assuming that a given sketch represents only a single object. It begins by finding the outer boundary of that object (Step 3). This is done by shooting a ray from the center of the sketch outward and identifying the last edge hit by the ray, which must be an external edge. The system then traces clockwise along the junctions between edges, always choosing the edge which is oriented the farthest in the clockwise direction, to determine the cycle of edges that make up the object's

outer boundary. Next, the system traces both clockwise and counter-clockwise among the inner edges that connect to these external edges, in order to find both of the surfaces that meet along each edge (Step 4).

This method will find all surfaces for the set of edges that are connected to the outer boundary of the sketched object. However, there may be other, internal sets of connected edges that do not connect to these edges. These internal edges might represent a hole or protuberance on the object. In order to find surfaces among the internal edges, the entire process is repeated, beginning with shooting out a ray to find an edge representing the exterior of the internal edges (Step 5). Exterior internal edges are also marked for the larger surface in which the internal set of edges is found.

Line Labelling

Surface extraction returns a set of surfaces, along with the cycle of edges that bounds each surface. Line labeling is used to determine which of these edges are actually part of the surface and which edges are part of another surface that is occluding this surface. We use an extension of Malik's (1987) line labeling algorithm that handles curved surfaces. This algorithm labels edges in a drawing as *convex* corners between surfaces, *concave* corners, *occluding* edges where one surface occludes another, and *limb* edges where a surfaces curves away from the viewer. A junction catalog specifies, for each type of junction, all possible combinations of labelings for the edges in it. Constraint satisfaction is used to solve for all edge labels.

Malik's (1987) algorithm and junction catalogue make several assumptions about the objects that are being interpreted. Unfortunately, the class of sketches we are examining, engineering design drawings, violate several of these assumptions. In the subsections that follow, we describe each of the assumptions that is violated and how we have adapted the junction catalog and labeling algorithm to deal with it. Figure 2 contains several example sketches. We will refer to specific junctions and surfaces within this figure by letter. Figure 3 shows the additions which were made to the junction catalogue.

1) Trihedral surfaces

The junction catalogue assumes that no more than three surfaces meet at any vertex. However, some of the design sketches considered contain a type of vertex made up of four surfaces, +-vertices. +-vertices are formed when two cuboids are adjacent but not quite aligned (see junction A in Figure 2). Though they are a meeting of four edges, they always appear in two-dimensional sketches as T-junctions (where two collinear edges are bisected by a third edge). We allow for these types of vertices by adding a new possible labeling for T-junctions, one in which instead of both collinear edges being *occluding* edges, one is an *occluding* edge and the other is a *concave* edge.

2) Piecewise smooth surfaces

Malik's algorithm assumes that surfaces curve smoothly. However, our design sketches often contain surfaces with a discontinuity in their curvature, where they change from being straight to being curved (see surface B). This type of surface has two effects. First, curved-L-junctions, where a straight edge and a curved edge meet, may appear between edges that lie along the exterior of these types of surfaces. We expanded the set of labelings for curved-L-junction to include all the labeling allowed for L-junctions (junctions between two straight edges) as well as one additional labeling in which both edges are *convex* (e.g., junction C). Second, there is a new type of junction, the curved-away-L-junction (junction D), in which the orientations of the straight edge and the curved edge are discontinuous at the point where they meet. This junction appears where a surface (such as B) meets another surface at the point where it changes from straight to curve. Its only possible labeling is *convex* for one edge and *concave* for the other.

3) No curved holes

The existing junction catalogues contain no labellings to

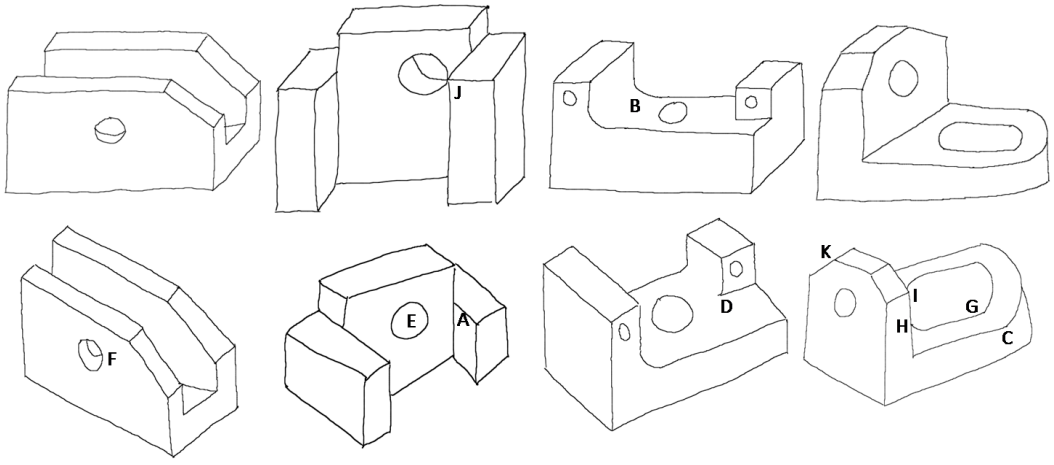


Figure 2. Four of the 12 objects sketched in CogSketch

deal with circular holes. This is a problem for design sketches because objects are often designed to fit together around a cylindrical axle, so the objects will contain holes. Often these holes are drawn as a simple ellipse (junction E). Other times, they appear as curved-T junctions (junction F). Because the edge circling around a hole can vary between convex and occluding, our system simply assigns all edges around curved holes a new label, *hole*.

In theory, an ellipse drawn by the user might indicate a sphere or a ring, as well as a hole. Our system relies on the assumption that the user is sketching only one object. Thus, any interior ellipse must be a hole. Similarly, any set of connected interior edges whose bounding edges are connected by only curved-T-junctions must be a hole. In fact, if a hole is not quite circular, it may also contain curved-L-junctions (see junction G). Thus, curved-L-junctions that are located along the bounding edges of an interior set of edges are reclassified as interior-L-junctions, and their edges can only be labeled as *hole* edges. In this example, a set of connected edges that actually do connect to the exterior edges are considered interior because all connections to the exterior edges are through T-junctions (junctions H and I), indicating that this is probably a set of interior edges that have been occluded by exterior edges.

4) No accidental viewpoints

Finally, traditional line labeling methods assume that drawings contain no accidental viewpoints, i.e., there are no junctions that are distorted by being viewed from just the wrong viewpoint. However, the design sketches contain two types of distortions. First, a viewpoint may place two junctions on top of each other, such that they appear to be a single junction at which four or more edges meet (junction J). Our system utilizes the simple expedient of ignoring any junction with more than three edges during labeling. Second, two connected edges that are not collinear in three-dimensional space may happen to line up in the sketch such that they appear to be collinear, causing a three-edge junction with them and a third edge to appear to be a T-junction (junction K). Our system initially looks for a normal sketch labeling and then, if this fails, looks for a labeling in which at most one T-junction in the sketch is ignored. If this fails, it increases the number of ignored T-junctions. In principle, this approach could result in a significant loss of efficiency, but in practice we have found there is never more than one or two distorted T-junctions.

Dealing with ambiguity

One weakness of the line labeling approach is that it can produce multiple consistent line labellings for a given sketch. Fortunately, the ambiguity can be decreased significantly by assuming that all the exterior edges of the object are occluding the background surface. However, there will still sometimes be a few possible labelings for some edges. In such cases, the system simply assumes that the first labeling found is correct. Unusual junction labels, such as the new T-junction labeling, are considered last to decrease the likelihood that they will be included if a simpler globally consistent labeling is available.

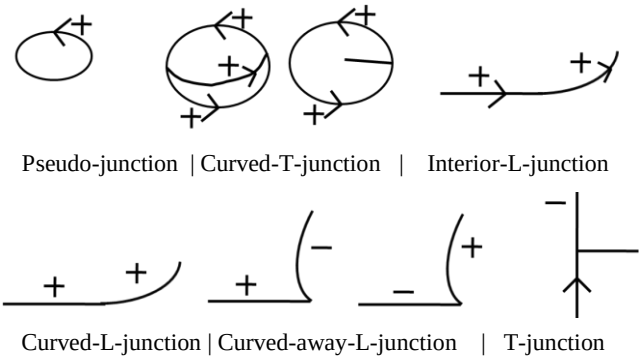


Figure 3. Additions to the junction catalogue (+ convex, - concave, ^ occluding, ^+ hole, unlabelled means unknown)

Qualitative Representation

The representations generated by the system contain an entity for each edge and each surface found in the sketch. In addition, they contain three types of qualitative spatial relations between these entities: corners along a surface, corners along an edge, and parallel surface relations. Corners are relatively local, and thus are represented as only first- or second-order relations. Parallel surface relations are more global, relating large parts of a sketch.

Corners along a surface

Every surface except the background has a cycle of edges that bound it. The edge labels tell the system which of these edges actually lie along the surface, rather than occluding the surface. For each pair of adjacent edges along a surface, the system asserts a relation describing the corner between them. Typically, corners are classified as convex or concave, although several additional classifications are used for corners that fall along unusual junctions (e.g., corners where a flat surface and a curved surface meet). Second-order relations are asserted to describing adjacent pairs of corners along a surface. See Figure 4 for a simple example.

Corners along an edge

Each edge labeled either convex or concave is a corner between two surfaces. Basic, first-order relations are asserted to describe these corners.

To simplify the process of representing holes, a single entity is constructed for a given hole, regardless of the number of surfaces actually found within the hole. Then, each of the edges along the hole is represented as a boundary between the surrounding surface and the hole. Second-order relations are included to represent pairs of adjacent edges that lie around a hole. However, none of the edges lying within the hole are represented. Thus, anything located along the inside of a hole or visible through the hole is ignored in the present representation scheme.

Parallel surface relations

The corner relations described thus far are fairly local. An object may contain a large number of similar-looking corners. A representation consisting only of a large number of similar, low-level relations causes problems for SME, because SME does not perform an exhaustive search for an optimal mapping. Thus larger-scale, higher-order relations are required to anchor the match.

We rely on a heuristic about parallel edges in drawings to infer higher-order structure. As Varley, Martin, and Suzuki (2005) observe, parallel lines in a drawing *usually* correspond to parallel edges in the three-dimensional object being drawn. Therefore, if in the sketch one planar surface has a corner A and another planar surface has a corner B such that the first edge of corner A is parallel to the first edge of corner B and the second edge of corner A is parallel to the second edge of corner B, then the two surfaces are almost certainly parallel in three dimensions.

The system asserts three types of parallel surface relations for surfaces that possess corners with pairs of parallel edges. The first is for cases when one of the pairs of edges is actually collinear. In this case, the evidence for the surfaces being parallel is greatest because collinear lines in a drawing nearly always correspond to collinear edges in three dimensions (Varley et al., 2005). The system asserts a higher-order relation stating that the colinearity of the two edges supports the two surfaces being parallel.

The second type of relation is for cases where the parallel surfaces are each connected to the same third surface by parallel edges. A higher-order relation is asserted stating that the fact that the two corners between the parallel surfaces and the third surface are parallel supports the belief that the two surfaces are parallel.

The third type is for all other cases where two surfaces have corners with pairs of parallel edges. This is the weakest evidence for parallel surfaces, so the system simply asserts a first-order parallel surface relation.

All of the parallel surface relations described thus far are symmetric relations. A symmetric relation is one in which the order of the arguments can be reversed. For example, the relation `(parallelSurfaces A B)` is identical to `(parallelSurfaces B A)`. This representation makes no commitment about the relative position or orientation of the edges and surfaces being related.

People clearly use some orientation-specific information when comparing images, so we include orientation-specific higher-order relations to aid SME in finding the correct mapping. The system asserts an orientation-specific version of the parallel surface relation for collinear edges. Thus, while a mapping can be found between any pair of collinear edge relations, there will be a stronger mapping in cases where the relative position and orientation of the edges and surfaces is maintained.

Comparing Shapes

Given two sketches in CogSketch, our system generates qualitative representations of them as described above and uses SME to find a mapping between them. Given the nature of sketches, we add two additional constraints to the matching process. A mapping is *coherent* if edges that are connected in the base sketch correspond to edges that are connected in the target sketch. In cases where two edge mh's (match hypotheses between an edge in the base and an edge in the target) in a mapping are inconsistent, e.g., the two base edges are connected while the two target edges are unambiguously disconnected, both edge mh's are removed from the mapping, along with any mh's between relations that take those edges as arguments. In cases where an edge mh is inconsistent with a large number of other edge mh's, the entire SME match will be rerun with the constraint that the faulty edge mh must be excluded from all mappings. This may allow SME to find a superior mapping that it failed to find on the first run because of being distracted by incoherent edge matches.

A mapping is *complete* if every edge mh that can be included in the mapping without violating mapping coherence is included. This completeness criterion is very useful when SME is being used to recognize when two sketches are of the same object. Incomplete edge mh's are identified by finding where both the base edge and the target edge connect to additional edges that have been left out of the mapping. Note that if, say, only the base edge connects to additional edges, there is no problem; it may be that the corresponding edges are occluded in the target. Completeness is implemented by forbidding incomplete edge mhs to appear in mappings, along with any mhs between relations that take those edges as arguments.

Evaluation

We evaluated our system using a set of 12 sketched objects taken from an exercise in an introductory engineering

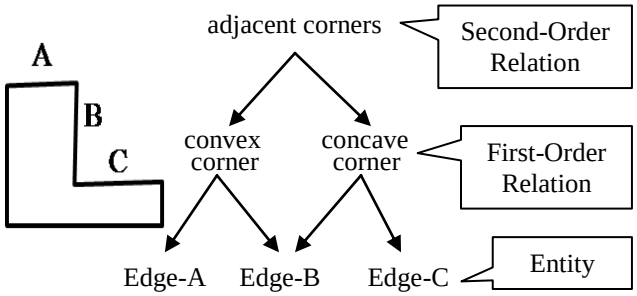


Figure 4. Typical corner relations

design textbook (Lueptow, 2007). In the exercise, sets of four objects were shown in each of three sketching perspectives (isometric, oblique, and orthographic), and students were asked to sketch the objects in the other two perspectives. These sketches were chosen because they were a beginning exercise, and hence not overly complicated, while still being representative of the kinds of 3D sketches engineers would be required to make.

We tested the system's ability to recognize oblique and isometric perspective sketches of the same object. The orientations of these two perspectives are about 45˚ different, resulting in a number of differences in the sketches. See Figure 2 for examples of four objects; the top row are sketched at an oblique perspective, and the bottom row are sketched at an isometric perspective.

A design student was asked to draw all 12 objects in both an oblique and and an isometric perspective. Then, one of the experimenters sketched each of the objects in CogSketch, using the student's sketches as a guide but making corrections where the student had made mistakes, such as forgetting to draw a hole in a surface.

Our system computed qualitative representations for all 24 sketches. Because only one of the objects contained internal edges that were not part of a hole, those edges were left out of the representation. Each sketch was then compared to the other 23 sketches using SME. The measures of success were (a) whether the line labeling algorithm provided correct results on each sketch, and (b) whether a given sketch's closest match was the other sketch of the same object, based on SME's mapping score.

Results

The output of the line labeling algorithm yielded correct results on all edges for 22 of the 24 sketches. The other two sketches showed minor mistakes; typically the correct labeling had also been found, but it was not the first labeling returned by the algorithm.

The recognition evaluation showed an overall success rate of 20/24, or 83%. That is, for 20 of the 24 sketches, the best mapping found by SME was with the other sketch of the same object. Because there were 22 distractor sketches, chance performance would be 1/23, or 4%.

The four mistakes occurred due to the failure of the system to recognize either of the perspectives of two of the objects. The rightmost object in Figure 2 is one of these. These objects contained partially curved edges that proved difficult to segment consistently. Also, the other problem object was rotated enough to make a single surface in one perspective appear to be two surfaces in the other.

Related Work

Most work on sketch recognition focuses on recognizing objects drawn at the same orientation. Nonetheless, recognition systems with quantitative representation systems often require 20-50+ training examples per category (Liwicki & Knipping, 2005; Sharon & van de Panne, 2006), or can only be trained and evaluated on sketches by a single user (Sezgin & Davis, 2007).

Previous work on constructing three-dimensional representations of sketches has tended to focused on recovering frontal geometry (Varley et al., 2005; Kaplan & Cohen, 2006), i.e., the distance to each point along the visible surfaces. Because these distances change as surfaces rotate in space, it is unclear whether this type of representation would be useful in comparing two sketches of an object at different orientations.

Discussion and Future Work

In the evaluation, our system demonstrated that it was capable of constructing qualitative spatial representations sufficiently robust to recognize two sketches of an object drawn at different orientations, despite a large number of distracters. Of the 12 objects being represented, only 2 caused problems for the system. We believe that these initial results are promising, and that they show it is possible, using qualitative representations, to accurately compare different-looking sketches of the same object, at least within the domain of engineering design.

However, the system possesses a major limitation. While it allows for a few junction distortions due to the viewpoint, it assumes the user has sketched the object nearly perfectly, allowing a globally consistent line labeling to be found. This is fine for experts, but for naïve users, a more flexible line labeling strategy will be needed. The probabilistic line labeling algorithm developed by Varley et al. (2004) is one promising option.

Being able to construct robust qualitative 3D representations from 2D sketches and identify them via comparison will facilitate using sketch understanding in a variety of applications. These include education in engineering, geoscience, and other highly spatial areas, plus support tools for creative conceptual design. We hope to explore these in future work.

Acknowledgements

This work was supported by NSF SLC Grant SBE-0541957, the Spatial Intelligence and Learning Center (SILC).

References

Falkenhainer, B., Forbus, K., and Gentner, D. 1986. The Structure-Mapping Engine. In Proceedings of AAAI '86.

Forbus, K., Usher, J., Lovett, A., Lockwood, K., and Wetzel, J. 2008. CogSketch: Open-Domain Sketch Understanding for Cognitive Science Research and For Education. In Proceedings of Eurographics Sketch-Based Interfaces and Modeling.

Forbus, K., Ferguson R., and Usher, J. 2001. Towards a computational model of sketching. In Proceedings of Intelligent User Interfaces.

Gentner, D. 1983. Structure-Mapping: A Theoretical Framework for Analogy. Cognitive Science 7(2): 155-170.

Kaplan, M., and Cohen, E. 2006. Producing models from drawings of curved surfaces. Workshop on Sketch-Based Interfaces and Modeling.

Liwicki, M., and Knipping, L. 2005. Recognizing and Simulating Sketched Logic Circuits. In Proceedings of the 9th International Conference on Knowledge-Based Intelligent Information & Engineering Systems, 588 – 594.

Lovett, A., Dehghani, M., and Forbus, K. 2007a. Incremental Learning of Perceptual Categories for Open-domain Sketch Recognition. In Proceedings IJCAI '07.

Lovett, A., Forbus, K., and Usher, J. 2007b. Analogy with Qualitative Spatial Representations Can Simulate Solving Raven's Progressive Matrices. In Proceedings of the 29th Annual Conference of the Cognitive Society.

Lueptow, R. M. 2007. *Graphic Concepts for Computer Aided Design.* Upper Saddle River, NJ: Prentice Hall.

Malik, J. 1987. Interpreting Line Drawings of Curved Objects. International Journal of Computer Vision 1: 73-103.

Sharon, D., and van de Panne M. 2006. Constellation Models for Sketch Recognition. In 3rd Eurographics Workshop on Sketch-Based Interfaces and Modeling.

Sezgin, T. M., & Davis, R. 2007. Sketch interpretation using multiscale models of temporal patterns. IEEE Computer Graphics and Applications 27(1): 28-37.

Tomai, E., Lovett, A., Forbus, K., and Usher, J. 2005. A Structure Mapping Model for Solving Geometric Analogy Problems. In Proceedings of the 27th Annual Conference of the Cognitive Science Society.

Varley, P. A. C., Martin, R. R., and Suzuki, H. 2005. Frontal Geometry from Sketch of Engineering Objects: Is Line Labelling Necessary? Computer-Aided Design 37: 1285-1307.

Varley, P. A. C., Martin, R. R., and Suzuki, H. 2004. Making the Most of Using Depth Reasoning to Label Line Drawings of Engineering Objects. In Proceedings of the 9th ACM Symposium on Solid Modeling and Applications.

Temporal Logic Patterns for Querying Qualitative Models of Genetic Regulatory Networks*

Pedro T. Monteiro[1,2], Delphine Ropers[1], Radu Mateescu[1], Ana T. Freitas[2], and Hidde de Jong[1]

[1]INRIA Grenoble - Rhône-Alpes, 655 Av. de l'Europe, Montbonnot, 38334 St. Ismier Cedex, France
{Pedro.Monteiro, Delphine.Ropers, Radu.Mateescu, Hidde.de-Jong}@inrialpes.fr
[2]IST/INESC-ID, 9 Rua Alves Redol, 1000-029 Lisboa, Portugal
atf@inesc-id.pt

Abstract

Formal verification based on model checking provides a powerful technology to query qualitative models of dynamical systems. The application of model-checking approaches is hampered, however, by the difficulty for non-expert users to formulate appropriate questions in temporal logic. In order to deal with this problem, we propose the use of patterns, that is, high-level query templates capturing recurring questions which can be automatically translated to temporal logic. We develop a set of patterns for the analysis of qualitative models of genetic regulatory networks, which are sufficiently generic though to be useful in other application domains. The applicability of the patterns has been investigated by the analysis of a model of the network of global regulators controlling the carbon starvation response in *Escherichia coli*.

Introduction

Qualitative simulation provides predictions of the possible qualitative behavior of a dynamical system (Kuipers 1994). It is an attractive approach when little or no quantitative information on parameter values is available, or when one is interested in the range of possible qualitative behaviors compatible with the structure of the system. These conditions are often met in the analysis of biological systems, which explains the popularity of qualitative approaches in mathematical and theoretical biology (*e.g.*, (Batt et al. 2007; Bellazzi et al. 2001; King, Garrett, and Coghill 2005; Thomas, Thieffry, and Kaufman 1995)). An example is the method for the *qualitative simulation of genetic regulatory networks* described in (Batt et al. 2007). This approach is based on a class of piecewise-linear (PL) differential equation models to describe regulatory interactions between genes, and has been implemented in the computer tool Genetic Network Analyzer (GNA).

A problem with the use of qualitative simulation is the potential explosion of the number of qualitative behaviors when dealing with large and complex systems whose dynamics cannot be sufficiently constrained. In order to deal with this problem, the use of *model-checking techniques* has been proposed (Shults and Kuipers 1997). This approach was successfully explored for the validation of qualitative models of genetic regulatory networks, by coupling GNA to state-of-the-art model checkers (Batt et al. 2005). It allows model predictions to be verified by experimental observations expressed as statements in temporal logic.

Formal verification based on model checking provides a powerful technology to query qualitative models, but it raises new issues, notably that of formulating good questions when analyzing a large model. Posing relevant and interesting questions is critical in modeling in general, but even more so in the context of applying formal verification techniques, due to the fact that it is not easy for non-experts to formulate queries in temporal logic. The response to this problem proposed by the formal verification community is the use of *patterns*, that is, high-level query templates that capture recurring questions in a specific application domain and that can be automatically translated to temporal logic (Dwyer, Avrunin, and Corbett 1999). This approach does not seem to have received any attention in qualitative reasoning thus far.

The aim of this paper is to develop a set of patterns for the analysis of models of genetic regulatory networks. Its main contributions are twofold. First, we develop a set of generic query templates, based on a review of frequently-asked questions by modelers, and translate these templates to temporal logic formulas.Although the patterns have been formulated for the analysis of genetic regulatory networks, they are sufficiently generic to carry over to other application domains. Second, we show the interest of the patterns in a case-study, concerned with the analysis of a large and complex model of the *E. coli* carbon starvation response. This model extends a previous model (Ropers et al. 2006) by taking into account additional regulators of bacterial stress responses.

Patterns for querying qualitative models

Description of network dynamics

As a basic hypothesis, we assume that the dynamics of genetic regulatory networks can be modeled by means of *finite state transition systems* (FSTSs) (Clarke, Grumberg, and Peled 1999). The latter formalism provides a general description of a dynamical system that explicitly underlies GNA (Batt et al. 2007), but the predictions of other qualita-

*This paper also appears in the proceedings of the Eighteenth European Conference on Artificial Intelligence, ECAI'08.

tive simulators can also be mapped to FTSTs. The generality of the FSTS formalism is important for assuring the wide applicability of the patterns developed in this section. Moreover, statements in temporal logic are usually interpreted on FSTSs, so that the latter naturally connect qualitative models to model-checking tools.

A finite state transition system is formally defined as a tuple $\Sigma = \langle S, AP, L, T, S_0 \rangle$, where S is a set of states, AP is a set of atomic propositions, $L : S \rightarrow 2^{AP}$ is a labeling function that associates to a state $s \in S$ the set of atomic propositions satisfied by s, $T \subseteq S \times S$ is a relation defining transitions between states, and $S_0 \subseteq S$ is a set of initial states. For our purpose, S describes the possible states of the genetic regulatory network, each of which is characterized by a set of atomic propositions, such as that the concentration of protein P is above a threshold and increasing.

Identification of patterns

The notion of patterns was introduced in the domain of software engineering as a means to capture expert solutions to recurring problems in program design. In the formal verification domain they have been introduced in an influential paper (Dwyer, Avrunin, and Corbett 1999), to help non-expert users formulate their temporal-logic queries. In the latter context, patterns are high-level descriptions of frequently asked questions in an application domain that are formulated in structured natural language rather than temporal logic. The aim of the patterns is not to cover all possible questions an expert can think of, but rather to simplify the formulation of those that are primary.

The difficulty of proposing patterns is to come up with a limited number of query schemas that are sufficiently generic to be applicable in a variety of situations, and at the same time sufficiently concrete to be comprehensible for the non-expert user. Moreover, the overlap between the patterns should be minimal. We analyzed a large number of modeling studies in systems biology (starting from the references in (Szallazi, Periwal, and Stelling 2006)), as well as lists of temporal logic queries (*e.g.*, (Chabrier-Rivier et al. 2004)). This bibliographic research allowed us to identify an open-ended list of questions on the dynamics of genetic, metabolic, and signal transduction networks. For instance, "Is the basal glycerol production level combined with rapid closure of Fps1 sufficient to explain an initial glycerol accumulation after osmotic shock?" (Klipp et al. 2005).

The identified questions were grouped into four categories, depending on whether they concerned the *occurrence/exclusion*, *consequence*, *sequence*, and *invariance* of cellular events. For each of these, we developed an appropriate pattern, capturing the essence of the question and the most relevant variants.

Description of patterns

The patterns consist of structured natural language phrases, represented in schematic form, with placeholders for so-called *state descriptors*. A state descriptor is a statement expressing a state property, and takes the form of (a Boolean combination of) atomic propositions. Let ϕ, ψ be state descriptors, then

$$\begin{aligned} \phi, \psi ::= & \; p_1 \in AP \mid p_2 \in AP \mid \ldots \\ ::= & \; \neg\phi \mid \phi \wedge \psi \mid \phi \Rightarrow \psi \mid \ldots \end{aligned}$$

The state descriptors are interpreted on the FSTS, in the sense that their meaning is formally defined as the set of states $S_1 \subseteq S$ satisfying the state descriptor. In addition to (Boolean combinations of) atomic propositions, the state descriptors may be temporal-logic formulas defined on the atomic propositions AP. However, the precise definition of the state descriptors depends on the particular type of FSTS that is used, as the latter determines AP.

Definition 1 (Occurrence/exclusion pattern)

It	is possible	for a state	ϕ	to occur
	is not possible			

This pattern represents the concepts of *occurrence* and its negation, *exclusion* (to capture safety properties). It will often be used during the development of a model to check for the presence or absence of some property that was experimentally observed. For instance, "It is possible for a state with a high concentration of protein P_1 to occur". Using this pattern, we can also check for *mutual exclusion*, by using the pattern negative form in combination with a conjunctive state descriptor. For instance, "It is not possible for a state to occur in which genes g_1 and g_2 are highly expressed".

Definition 2 (Consequence pattern)

If a state	ϕ	occurs,

then it is	possibly	followed by a state	ψ
	necessarily		

The *consequence* pattern relates two events separated in time. More precisely, it expresses that if the first state occurs, then it is possibly or necessarily followed by the second state. If the latter state necessarily follows, then the consequence pattern expresses a form of causal relation. An instance of this pattern is, for example, "If a state occurs in which the concentration of protein P is below 5 μM, then it is necessarily followed by a state in which the expression of gene *g* is at its basal level".

Definition 3 (Sequence pattern)

A state	ψ	is reachable and

is	possibly	preceded	at some time	by a state	ϕ
	necessarily		all the time		

The *sequence* pattern represents an ordering relation between two events. It ought not to be confused with the *consequence* pattern, since the conditional occurrence of the second state which characterizes the latter is absent in the *sequence* pattern. It must be possible to observe both the first and the second state, in that order, for an instance of the *sequence* pattern to be true.

Four variants of the pattern are distinguished, depending on whether the second state follows possibly or necessarily

Occurrence/Exclusion pattern	**CTL**	**μ-calculus**
It **is possible** for a state ϕ to occur	$EF\ (\phi)$	$\mu X.(\phi \vee \Diamond X)$
It **is not possible** for a state ϕ to occur	$\neg EF\ (\phi)$	$\neg\mu X.(\phi \vee \Diamond X)$
Consequence pattern		
If a state ϕ occurs, then it is **possibly** followed by a state ψ	$AG\ (\phi \Rightarrow EF\ (\psi))$	$\nu X.((\phi \Rightarrow \mu Y.(\psi \vee \Diamond Y)) \wedge \Box X)$
If a state ϕ occurs, then it is **necessarily** followed by a state ψ	$AG\ (\phi \Rightarrow AF\ (\psi))$	$\nu X.((\phi \Rightarrow \mu Y.(\psi \vee \Box Y)) \wedge \Box X)$
Sequence pattern		
A state ψ is reachable and is **possibly** preceded **at some time** by a state ϕ	$EF\ (\phi \wedge EF\ (\psi))$	$\mu X.((\phi \wedge \mu Y.(\psi \vee \Diamond Y)) \vee \Diamond X)$
A state ψ is reachable and is **possibly** preceded **all the time** by a state ϕ	$E\ (\phi\ U\ \psi)$	$\mu X.(\psi \vee (\phi \wedge \Diamond X))$
A state ψ is reachable and is **necessarily** preceded **at some time** by a state ϕ	$EF\ (\psi) \wedge \neg E\ (\neg\phi\ U\ \psi)$	$\mu X.(\psi \vee \Diamond X) \wedge \neg\mu Y.(\psi \vee (\neg\phi \wedge \Diamond Y))$
A state ψ is reachable and is **necessarily** preceded **all the time** by a state ϕ	$EF\ (\psi) \wedge AG\ (\neg\phi \Rightarrow AG\ (\neg\psi))$	$\mu X.(\psi \vee \Diamond X)\ \wedge \nu Y.((\phi \vee \nu Z.(\neg\psi \wedge \Box Z)) \wedge \Box Y)$
Invariance pattern		
A state ϕ **can** persist indefinitely	$EG\ (\phi)$	$\nu X.(\phi \wedge \Diamond X)$
A state ϕ **must** persist indefinitely	$AG\ (\phi)$	$\nu X.(\phi \wedge \Box X)$

Table 1: Rules for the translation of the patterns into CTL and μ-calculus. For each of the four patterns, the translation of all variants is shown. We use the version of μ-calculus presented in (Kupferman, Vardi, and Wolper 2000), which is interpreted on classical Kripke structures. The symbol T stands for True.

after the first state, and whether the system is in the first state all the time or only at some time before the occurrence of the second state. An instance of this pattern is "A steady state is reachable and is necessarily preceded all the time by a state in which nutrient N is absent".

Definition 4 (Invariance pattern)

A state	ϕ	can	persist indefinitely
		must	

The *invariance* pattern is used to check if the system can or must remain indefinitely in a state. In contrast with the *occurrence/exclusion* pattern, the question is not whether a particular state can be reached, but rather whether a particular state is invariable. An instance of this pattern is "A state with a basal expression of gene g must persist indefinitely".

Translation to temporal logic

By defining a temporal-logic translation of the patterns, the user queries can be automatically cast in a form that allows the verification of the specified property by means of model-checking tools. The patterns defined above are independent of a particular temporal logic, which allows the same high-level specification of a user query to be verified by means of different approaches and tools. It is worth noticing though that some of the patterns we propose have a branching-time nature (*e.g.*, the *consequence* and the *sequence* patterns), and therefore these are not translatable into a linear-time formalism, such as LTL (Clarke, Grumberg, and Peled 1999).

Two examples of translations of the previously defined patterns are shown in tabular form: the Computational Tree Logic (CTL) translation and the μ-calculus translation (Table 1). In both CTL and μ-calculus, formulas are built upon atomic propositions. Also, the usual connectors of propositional logic, such as negation ($\neg$), logical or ($\vee$), logical and ($\wedge$) and implication ($\Rightarrow$), can be used in both logics. In addition, CTL provides two types of operators: *path quantifiers*, **E** and **A**, and *temporal operators*, such as **F** and **G**. Path quantifiers are used to specify that a property p is satisfied by some (**E** p) or every (**A** p) path starting from a given state. Temporal operators are used to specify that, given a state and a path starting from that state, a property p holds for some (**F** p) or for every (**G** p) state of the path. Each path quantifier must be paired with a temporal operator. In the case of μ-calculus, two types of operators are provided: the least (μ) and greatest (ν) *fixed points*, and the *modal operators* possibility ($\Diamond$) and necessity ($\Box$). Least and greatest fixed points specify finite and infinite recursive applications of a formula, respectively. For instance, given a state and a path starting from that state, the fact that a property p holds for some state or for all states of the path is expressed using a least (μ) or a greatest (ν) fixed point, respectively. Modal operators are used to specify that, given a state, p possibly ($\Diamond$ p) or necessarily ($\Box$ p) holds on some or all of its outgoing states.

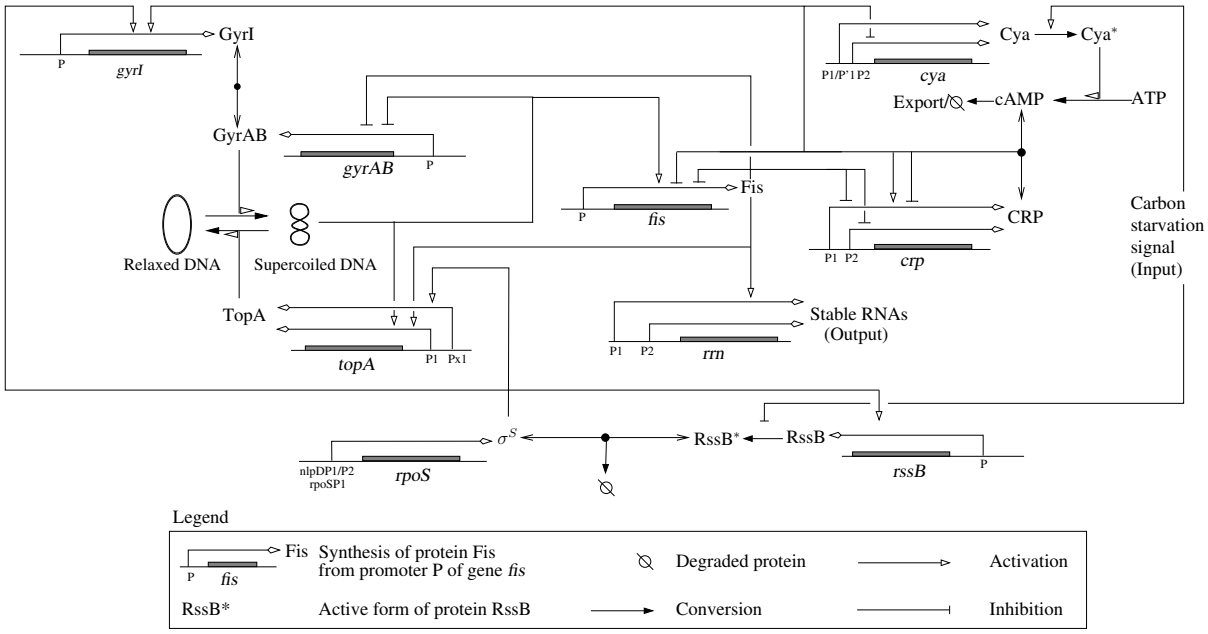
Carbon starvation response in *E. coli*

Model of carbon starvation response

To test the applicability of the temporal logic patterns, we have used our approach for the analysis of a model of the carbon starvation response in the bacterium *E. coli*. In the absence of essential carbon sources in its growth environment, an *E. coli* population abandons exponential growth and enters a non-growth state called stationary phase. This growth-phase transition is accompanied by numerous physiological changes in the bacteria, and controlled on the molecular level by a complex genetic regulatory network.

The molecular basis of the adaptation of the growth of *E. coli* to the nutritional conditions has been the focus of extensive studies for decades (Gutierrez-Ríos et al. 2007; Hengge-Aronis 1996). However, notwithstanding the enormous amount of information accumulated on the genes, proteins, and other molecules, kinetic parameters and the molecular concentrations are absent, with some exceptions, which makes it difficult to apply traditional methods for the dynamical modeling of genetic regulatory networks.

These circumstances have motivated the development of a qualitative model of the carbon starvation response net-



(a)

$$\dot{x}_{gyrAB} = \kappa_{gyrAB}\,(1 - s^{+}(x_{gyrAB}, \theta^{2}_{gyrAB})\; s^{-}(x_{gyrI}, \theta^{1}_{gyrI})\; s^{-}(x_{topA}, \theta^{1}_{topA}))\; s^{-}(x_{fis}, \theta^{4}_{fis}) - \gamma_{gyrAB}\; x_{gyrAB}$$

$$0 < \theta^{1}_{gyrAB} < \theta^{2}_{gyrAB} < \kappa_{gyrAB}/\gamma_{gyrAB} < max_{gyrAB} \quad \text{(b)}$$

Figure 1: (a) Network of key genes, proteins and regulatory interactions involved in the carbon starvation response network in *E. coli*. (b) PL differential equation and parameter inequality constraints for the gyrase GyrAB. The variable x_{gyrAB} denotes the concentration of GyrAB. The protein is produced at a rate κ_{gyrAB} if the DNA supercoiling level is not high, that is, if the concentration of GyrAB itself is below the threshold θ^{2}_{gyrAB}, and the concentrations of the topoisomerase TopA and the gyrase inhibitor GyrI are above the thresholds θ^{1}_{topA} and θ^{1}_{gyrI}, respectively. The regulatory logic of *gyrAB* expression is modeled by means of step functions. For instance, $s^{+}(x_{gyrAB}, \theta^{2}_{gyrAB})$ evaluates to 1, if $x_{gyrAB} > \theta^{2}_{gyrAB}$ (and to 0 otherwise). The protein is degraded at a rate proportional to its own concentration, $\gamma_{gyrAB}\; x_{gyrAB}$. The constraint $\theta^{2}_{gyrAB} < \kappa_{gyrAB}/\gamma_{gyrAB} < max_{gyrAB}$ express that the derepression of the *gyrAB* promoter allows the concentration of GyrAB to reach a high level, above the threshold θ^{2}_{gyrAB}. Instead of numerical values, the qualitative simulator uses such inequality constraints to infer behavior predictions (Batt et al. 2007; 2005).

work using a class of *piecewise-linear (PL) differential equations*. The PL models, originally introduced on (Glass and Kauffman 1973), provide a coarse-grained picture of the dynamics of genetic regulatory networks. They associate a protein concentration variable to each of the genes in the network, and capture the switch-like character of gene regulation by means of step functions that change their value at a threshold concentration of the proteins. The advantage of using PL models is that the qualitative dynamics of the high-dimensional systems are relatively simple to analyze, using inequality constraints on the parameters rather than exact numerical values (Batt et al. 2005; 2007). This makes the PL models a valuable tool for the analysis of the carbon starvation network.

In previous work we developed a PL model that we extend here by the general stress response factor RpoS and related regulators (Ropers et al. 2006; Ropers et al., in preparation). The dynamics of this system are described by nine coupled PL differential equations, and fifty inequality constraints on the parameter values.

Qualitative simulation of starvation response

The mathematical properties of the class of PL models used for modeling the stress response network have been well-studied (Glass and Kauffman 1973). It was previously shown how discrete abstractions can be used to convert the continuous dynamics of the PL system into a FSTS (Batt et al. 2007). The states S of the FSTS correspond to hyperrectangular regions in the concentration space, while the transitions T arise from trajectories entering one region from another. The atomic propositions AP describe, among other things, the concentration bounds of the regions and the trend of the variables inside a region (increasing, decreasing, or steady). The generation of the FSTS from the PL model has been implemented in the computer tool GNA (Batt et al. 2005). GNA is able to export the FSTS to standard model checkers like NuSMV (Cimatti et al. 2002) and CADP (Garavel, Lang, and Mateescu 2007), supporting the use of CTL and μ-calculus, respectively.

The application of this approach to the model of the *E. coli* carbon starvation network generates a huge FSTS. The entire state set consists of approximately $\mathcal{O}(10^{10})$ states, while the subset of states that is most relevant for our pur-

Properties	Response
Occurrence/exclusion pattern: Mutual inhibition of Fis and CRP	True
| It | is not possible | for a state | $x_{crp} \geq \frac{k^1_{crp}+k^2_{crp}+k^3_{crp}}{\gamma_{crp}} \wedge x_{fis} \geq \theta^4_{fis}$ | to occur | and | It | is not possible | for a state | $x_{crp} \leq \frac{k^1_{crp}}{\gamma_{crp}} \wedge x_{fis} \leq \theta^1_{fis}$ | to occur |	
CTL: $\neg EF(x_{crp} \geq \frac{k^1_{crp}+k^2_{crp}+k^3_{crp}}{\gamma_{crp}} \wedge x_{fis} \geq \theta^4_{fis}) \wedge \neg EF(x_{crp} \leq \frac{k^1_{crp}}{\gamma_{crp}} \wedge x_{fis} \leq \theta^1_{fis})$ μ-calculus: $\neg\mu X.((x_{crp} \geq \frac{k^1_{crp}+k^2_{crp}+k^3_{crp}}{\gamma_{crp}} \wedge x_{fis} \geq \theta^4_{fis}) \vee \Diamond X) \wedge \neg\mu X.((x_{crp} \leq \frac{k^1_{crp}}{\gamma_{crp}} \wedge x_{fis} \leq \theta^1_{fis}) \vee \Diamond X)$	
Consequence pattern: Damped oscillations after nutrient upshift	True
| If a state | $x_{signal} < \theta_{signal}$ | occurs, then it is | necessarily | followed by a state | $isOscillatoryState$ |	
CTL: $AG\ ((x_{signal} < \theta_{signal}) \Rightarrow AF\ (isOscillatoryState))$ μ-calculus: $\nu X.(((x_{signal} < \theta_{signal}) \Rightarrow \mu Y.(isOscillatoryState \vee \Box Y)) \wedge \Box X)$	
Sequence pattern: Control of entry into stationary phase by RpoS	True
| A state | $x_{rrn} < \theta_{rrn}$ | is reachable and is | necessarily | preceded | at some time | by a state | $x_{rpoS} \geq \theta^1_{rpoS}$ |	
CTL: $EF\ (x_{rrn} < \theta_{rrn}) \wedge \neg E\ (\neg(x_{rpoS} \geq \theta^1_{rpoS})\ U\ (x_{rrn} < \theta_{rrn}))$ μ-calculus: $\mu X.((x_{rrn} < \theta_{rrn}) \vee \Diamond X)\ \wedge \neg\mu Y.((x_{rrn} < \theta_{rrn}) \vee (\neg(x_{rpoS} \geq \theta^1_{rpoS}) \wedge \Diamond Y))$	
Invariance pattern: Expression of *topA* during growth-phase transitions	False
| A state | $x_{topA} < \theta^1_{topA}$ | can | persist indefinitely |	
CTL: $EG\ (x_{topA} < \theta^1_{topA})$ μ-calculus: $\nu X.((x_{topA} < \theta^1_{topA}) \wedge \Diamond X)$	

Table 2: Translation of properties used in the analysis of the *E. coli* carbon starvation response, following the translation rules in Table 1. The symbol $isOscillatoryState$ is a predicate attributed by the qualitative simulator to a state and indicating that the state is part of a cycle in the state transition graph.

pose, *i.e.* the states that are reachable from an initial state corresponding to a particular growth state of the bacteria, still consists of $\mathcal{O}(10^3)$ states. It is clear that FSTSs of this size cannot be analyzed by visual inspection, and that formal verification techniques are needed.

In the next section we show how the previously defined patterns can speed up the querying of these FSTSs, by simplifying the formulation of relevant properties to be tested.

Analysis of carbon starvation response model using query patterns

Four relevant properties were studied to analyze the *E.coli* carbon starvation response model (Table 2). The properties correspond to the following questions:

- Does the mutual inhibition motif of Fis and CRP (Fis inhibits the expression of gene *crp*, and CRP inhibits the expression of gene *fis*) have an effect on the dynamics of the carbon starvation response network?
- Is a carbon upshift a necessary condition for the occurrence of damped oscillations in the concentration of the regulators of the DNA supercoiling level?
- Is the entry into stationary phase always preceded by the accumulation of the stress response regulator RpoS?
- Is gene *topA* expressed in response to carbon source availability?

The instances of the patterns were translated into CTL following the translation rules of Table 1, and then verified using the model-checker NuSMV. The results are shown in the Table 2. By way of illustration we develop the formulation of the pattern for the third question and interpret the results of the verification process.

RpoS is a general stress response factor that allows cells to adapt to and survive under harmful conditions by entering stationary phase (Hengge-Aronis 1996). Due to its key role, the concentration of RpoS is tightly regulated, at the transcriptional, translational, and post-translational levels. The stability of the protein is mainly controlled in our conditions: while cells grow on a carbon source, RpoS is actively degraded through the protein RssB, which binds to RpoS and targets the factor to an intracellular protease. However, the depletion of the carbon source inactivates RssB, thus allowing RpoS to accumulate at a high concentration.

Given the importance of RpoS for cell survival, one may ask whether the entry into stationary phase is always preceded by the accumulation of RpoS in the cell. We formulated this question using a *sequence* pattern, where the stationary phase is represented by a low level of stable RNAs *rrn* (Table 2). The latter indicator is motivated by the fact that stationary-phase cells do not need high levels of these RNAs, which are necessary for the high translational activity of the exponential phase. The property is true, which indicates that the entry into stationary phase cannot occur before RpoS has accumulated. This points at the central role of RpoS in the growth adaptation of the bacteria.

Discussion

Formal verification techniques are promising tools for upscaling the analysis of qualitative models of genetic regulatory networks and other dynamical systems. The widespread adoption of model-checking approaches is restrained, however, by the difficulty for non-expert users to formulate appropriate questions in temporal logics. Inspired by work in the formal verification community (Dwyer, Avrunin, and

Corbett 1999), the first contribution of the paper consists in the formulation of a set of patterns in the form of query templates in structured natural language. In addition, we have provided translations of the patterns to two different temporal logics, CTL and μ-calculus. The patterns capture a large number of frequently-asked questions by modelers of regulatory networks, as for example listed in (Chabrier-Rivier et al. 2004). The second contribution of the paper concerns the instantiation of the patterns for the analysis of the complex genetic regulatory network involved in the carbon starvation response in *E. coli*. We have extended an existing model of the network with additional global regulators and verified the effect of the extensions on the predicted network dynamics.

The paper addresses issues we were confronted with when applying qualitative simulation techniques to a real-world problem in biology. We have proposed a solution, temporal logic query patterns for the analysis of large FSTSs, that has turned out to be useful in our application. However, we also expect this approach to carry over to other qualitative reasoning applications, where similar problems arise. Model checking is a promising way to analyze the large FSTSs arising in qualitative simulation (Shults and Kuipers 1997), but most modelers are not familiar with temporal logics and have difficulty in expressing their questions by means of these formalisms. Although meant to capture frequently-asked questions in biology, the patterns introduced in this paper are defined for FSTSs in general and seem sufficiently generic to apply to other problems as well. At the very least, they form a good starting-point for the formulation of a new set of query templates, tailored to the specificities of qualitative applications in other domains.

Acknowledgments

This work was partially supported by FCT program (PhD grant SFRH/BD/32965/2006 to PTM) and PDCT program (project PTDC/EIA/71587/2006). DR, RM, and HdJ are supported by the European Commission under project EC-MOAN (FP6-2005-NEST-PATH-COM/043235).

References

Batt, G.; Ropers, D.; de Jong, H.; Geiselmann, J.; Mateescu, R.; Page, M.; and Schneider, D. 2005. Analysis and verification of qualitative models of genetic regulatory networks: A model-checking approach. In Kaelbling, L., ed., *Proceedings of the Intl. Joint Conf. on Artif. Intel.*, 370–375.

Batt, G.; de Jong, H.; Page, M.; and Geiselmann, J. 2007. Symbolic reachability analysis of genetic regulatory networks using discrete abstractions. *Automatica* 44(4):982–989.

Bellazzi, R.; R., G.; Ironi, L.; and Patrini, C. 2001. A hybrid input-output approach to model metabolic systems: an application to intracellular thiamine kinetics. *J. Biomed. Inform.* 34(4):221–48.

Chabrier-Rivier, N.; Chiaverini, M.; Danos, V.; Fages, F.; and Schächter, V. 2004. Modeling and querying biomolecular interaction networks. *Theor. Comput. Sci.* 325(1):25–44.

Cimatti, A.; Clarke, E.; Giunchiglia, E.; Giunchiglia, F.; Pistore, M.; Roveri, M.; Sebastiani, R.; and Tacchella, A. 2002. NuSMV 2: An opensource tool for symbolic model checking. In Brinksma, D., and Larsen, K., eds., *Proceedings of the 14th Intl. Conf. on Comp. Aided Verif.*, volume 2404 of *LNCS*, 359–64. Berlin: Springer-Verlag.

Clarke, E.; Grumberg, O.; and Peled, D. 1999. *Model Checking*. Cambridge, MA: MIT Press.

Dwyer, M.; Avrunin, G.; and Corbett, J. 1999. Patterns in property specifications for finite-state verification. In *Proceedings of the 21st Intl. Conf. on Soft. Eng.*, 411–20.

Garavel, H.; Lang, F.; and Mateescu, R. 2007. CADP 2006: A toolbox for the construction and analysis of distributed processes. In Damm, W., and Hermanns, H., eds., *Proceedings of the 19th Intl. Conf. on Comp. Aided Verif.*, volume 4590 of *LNCS*, 158–63. Berlin: Springer-Verlag.

Glass, L., and Kauffman, S. 1973. The logical analysis of continuous non-linear biochemical control networks. *J. Theor. Biol.* 39(1):103–29.

Gutierrez-Ríos, R.; Freyre-Gonzalez, J.; Resendis, O.; Collado-Vides, J.; Saier, M.; and Gosset, G. 2007. Identification of regulatory network topological units coordinating the genome-wide transcriptional response to glucose in *Escherichia coli*. *BMC Microbiol.* 7(1):53.

Hengge-Aronis, R. 1996. Regulation of gene expression during entry into stationary phase. In F.C. Neidhardt, *et al*., ed., *Escherichia coli and Salmonella: Cellular and Molecular Biology*, 1497–512. Washington DC: ASM Press.

King, R.; Garrett, S.; and Coghill, G. 2005. On the use of qualitative reasoning to simulate and identify metabolic pathways. *Bioinformatics* 21(9):2017–26.

Klipp, E.; Nordlander, B.; Krüger, R.; Gennemark, P.; and Hohmann, S. 2005. Integrative model of the response of yeast to osmotic shock. *Nat. Biotechnol.* 23(8):975–82.

Kuipers, B. 1994. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. Cambridge, MA: MIT Press.

Kupferman, O.; Vardi, M.; and Wolper, P. 2000. An automata-theoretic approach to branching-time model checking. *J. ACM* 47(2):312–60.

Ropers, D.; de Jong, H.; Page, M.; Schneider, D.; and Geiselmann, J. 2006. Qualitative simulation of the carbon starvation response in *Escherichia coli*. *Biosystems* 84(2):124–52.

Shults, B., and Kuipers, B. 1997. Proving properties of continuous systems: Qualitative simulation and temporal logic. *Artif. Intell.* 92(1-2):91–130.

Szallazi, Z.; Periwal, V.; and Stelling, J. 2006. *System Modeling in Cellular Biology: From Concepts to Nuts and Bolts*. Cambridge, MA: MIT Press.

Thomas, R.; Thieffry, D.; and Kaufman, M. 1995. Dynamical behaviour of biological regulatory networks. *Bull. Math. Biol.* 57(2):247–276.

Continuous-Domain Reinforcement Learning Using a Learned Qualitative State Representation*

Jonathan Mugan and Benjamin Kuipers
Computer Science Department
University of Texas at Austin
Austin Texas, 78712 USA
{jmugan,kuipers}@cs.utexas.edu

Abstract

We present a method that allows an agent to learn a qualitative state representation that can be applied to reinforcement learning. By exploring the environment the agent is able to learn an abstraction that consists of landmarks that break the space into qualitative regions, and rules that predict changes in qualitative state. For each predictive rule the agent learns a context consisting of qualitative variables that predicts when the rule will be successful. The regions of this context in with the rule is likely to succeed serve as a natural goals for reinforcement learning. The reinforcement learning problems created by the agent are simple because the learned abstraction provides a mapping from the continuous input and motor variables to discrete states that aligns with the dynamics of the environment.

Introduction

Reinforcement learning in continuous domains is difficult because the agent is unable to gain experience at each individual state. This means that the agent must use an abstraction that allows it to map an infinite number of input and motor states into a manageable number of abstracted states. To be useful to the agent, the abstraction must discriminate states that are different, but if the abstraction makes too many unnecessary discriminations then learning becomes inefficient. This balance is often achieved by having a human create and tune the abstraction.

Our approach to this problem is to use a qualitative state representation. In (Mugan & Kuipers 2007a; 2007b), we showed how an agent could build a qualitative representation of its environment that is not specific to any particular goal. The agent does this by breaking the world up into qualitative regions using *landmarks* and then learning *predictive rules* over changes in qualitative state.

In our approach, the agent experiences the world through a set of continuous input and motor variables. The motor variables and the derivatives of the input variables have an intrinsic landmark at 0, which creates three qualitative states for each of those variables. The continuous input variables themselves have no intrinsic landmarks, and the agent must learn landmarks on these variables as well as additional landmarks on the motor variables. A change in the qualitative state of a variable defines an *event*. The agent searches for rules that predict when one event will follow another in time. For each learned predictive rule the agent searches for regions in the state space where that rule will be reliable and delimits those regions by creating new landmarks. Each new landmark defines new events, which make it possible to learn new predictive rules, and so on.

In this paper we show that this learned qualitative representation enables the agent to do reinforcement learning to perform a simple task. The agent defines its own reinforcement learning problems using the learned predictive rules and landmarks. There has been previous work on enabling an agent to define its own reinforcement learning problems. McGovern and Barto (2001) have proposed a method whereby an agent autonomously finds subgoals based on bottleneck states that are visited often during successful trials. Subgoals have also been found by searching for "access states" (Simsek & Barto 2004; Simsek, Wolfe, & Barto 2005) that allow the agent to go from one part of the state space to another. In this paper we use a different approach to identifying goals for reinforcement learning, we define the goals for reinforcement learning to be the regions defined by landmarks in which a predictive rule is reliable. Additionally, there has been work on learning qualitative values given a model so that the level of abstraction is appropriate for the task (Sachenbacher & Struss 2005). Our work differs from this work because our focus is on learning the model and the abstraction simultaneously.

In the remainder of this paper we first give an overview of the qualitative abstraction and rule learning framework. We then explain how we incorporate reinforcement learning into this framework. Finally, we evaluate the viability of this approach by comparing it to a standard method for reinforcement learning with continuous variables on a simple task.

*This work has taken place in the Intelligent Robotics Lab at the Artificial Intelligence Laboratory, The University of Texas at Austin. Research of the Intelligent Robotics lab is supported in part by grants from the Texas Advanced Research Program (3658-0170-2007), from the National Science Foundation (IIS-0413257, IIS-0713150, and IIS-0750011), and from the National Institutes of Health (EY016089).

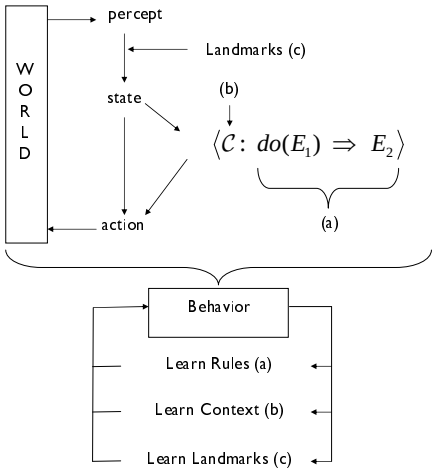


Figure 1: (a) The agent interacts with the world to learn rules stating that when the agent makes one event E_1 occur, that another event E_2 occurs. (b) For each rule the agent learns a context $\mathcal{C}$ that consists of a set of variables upon which the agent can learn a conditional probability table $CPT(r) = Pr(succeeds(r)|\mathcal{C})$ that tells the agent in which situations it can cause E_2 by making E_1 occur. (c) The agent also learns landmarks that determine how it can perceive and reason about the world. Landmarks are proposed based on the behavior of rules and in turn determine which rules can be learned (Mugan & Kuipers 2007a).

Learning Agent Architecture

The overview of the learning agent architecture is shown in Figure 1.

Qualitative State Representation

The raw input and output is represented using three types of variables: continuous motor variables, continuous input variables, and nominal input variables. Internally, the agent represents these variables qualitatively based on QSIM (Kuipers 1994). The agent creates two variables for each continuous input variable $\tilde{v}$, a discrete variable $v(t)$ that represents the magnitude of $\tilde{v}(t)$, and a discrete variable $\dot{v}(t)$ that represents the direction of change of $\tilde{v}(t)$. And for each continuous motor variable $\tilde{v}$, the agent creates a discrete magnitude variable $v(t)$. The result of this is that the agent represents its world using four types of discrete (qualitative) variables: motor variables, magnitude variables, direction of change variables, and nominal variables.

We now describe how the agent converts continuous values to qualitative values. A continuous variable $\tilde{v}(t)$ ranges over some subset of the real number line $(-\infty, +\infty)$. In QSIM, this continuous variable $\tilde{v}(t)$ is abstracted to a discrete variable $v(t)$ that ranges over a *quantity space* $\mathcal{Q}(v)$ of qualitative values. $\mathcal{Q}(v) = L(v) \cup I(v)$, where $L(v) = \{v_1^*, \cdots v_n^*\}$ is a totally ordered set of landmark values, and $I(v) = \{(-\infty, v_1^*), (v_1^*, v_2^*), \cdots (v_n^*, +\infty)\}$ is the set of mutually disjoint open intervals that $L(v)$ defines in the real number line. A quantity space with two landmarks might be described by (v_1^*, v_2^*), which implies five distinct qualitative values, $\mathcal{Q}(v) = \{(-\infty, v_1^*), v_1^*, (v_1^*, v_2^*), v_2^*, (v_2^*, +\infty)\}$.

Each direction of change variable $\dot{v}$ has a single intrinsic landmark at 0, so its quantity space is $\mathcal{Q}(\dot{v}) = \{(-\infty, 0), 0, (0, +\infty)\}$. Magnitude variables initially have no landmarks because zero is just another point on the number line, although landmarks are acquired as the agent learns. Initially, when the agent knows of no meaningful qualitative distinctions among values for $\tilde{v}(t)$, we describe the quantity space as the empty list of landmarks, $()$. Motor variables are given an initial landmark at 0, and like magnitude variables, they can acquire more landmarks as the agent learns. As an implementation note, because we evaluate the algorithm with a fine-grained discrete-timestep simulator, if v_1^* is a landmark and $\tilde{v}(t-1) < v_1^*$ and $\tilde{v}(t) > v_1^*$ then $v(t) = v_1^*$ for the purpose of rule learning.

Events

If a is a qualitative value of a discrete variable A, meaning $a \in \mathcal{Q}(A)$, then the *event* $A_t{\rightarrow}a$ is defined by $A(t-1) \neq a$ and $A(t) = a$. That is, an event takes place when a discrete variable A changes to value a at time t, from some other value. We will often drop the t and describe this simply as $A{\rightarrow}a$. We will also refer to an event as E when the variable and qualitative value involved are not important.

Predictive Rules

We break from previous work (2007a; 2007b) and take inspiration from (Pearl 2000) to define predictive rules based on actions of the agent. A *predictive rule* r has the form $r = \langle \mathcal{C} : do(E_1) \Rightarrow E_2 \rangle$ and states that if the agent executes a plan to bring about event E_1, then event E_2 will soon follow. The probability that E_2 will indeed soon follow E_1 is given in the context $\mathcal{C}$. For an event E we define $do_t(E)$ as a predicate that is true when the agent begins executing a plan at time t to bring about E and is false otherwise.

The predictive rule $r = \langle \mathcal{C} : do(E_1) \Rightarrow E_2 \rangle$ consists of one event $E_1(t)$, and another event $E_2(t')$ that takes place relatively soon after t. That E_2 takes place "relatively soon after" $E_1(t)$ is formalized in terms of an integer time-delay k (in our current implementation $k = 5$, or about 0.25 seconds)

$$soon(t, E_2) \equiv \exists t' \, [t \leq t' \leq t + k \, \wedge \, E_2(t')]$$

If we define $do_{r,t}(E(t'))$ to mean that $do_t(E) = \text{true}$ and that E occurs at time t', then we can define a predicate $succeeds(r, t)$ for the success of rule r as

$$succeeds(r, t) \equiv do_{r,t}(E_1(t')) \, \wedge \, soon(t', E_2)$$

This means that rule r fails if the agent's plan to bring about E_1 fails, or if E_2 does not soon follow. We shorten $succeeds(r, t)$ to the predicate $succeeds(r)$, which is true if r succeeds when activated at an arbitrary time.

Associated with rule r is a context $\mathcal{C}$ that consists of a set of variables. The context induces a conditional probability table (CPT) on the predicate $succeeds(r)$. In Bayesian

network terminology, the variables in $\mathcal{C}$ are the parents of $succeeds(r)$. For a rule $r = \langle \mathcal{C} : do(A \rightarrow a) \Rightarrow B \rightarrow b \rangle$ we require that the elements of the context be magnitude or nominal variables and that for event $B \rightarrow b$ we require that B be a nominal variable or that B be a direction of change variable with $b \neq [0]$.

Learning New Predictive Rules

To learn a new predictive rule the agent searches for two events E_1 and E_2 such that observing event E_1 means that event E_2 is significantly more likely to occur than it would have been otherwise. When two such events are found the agent asserts an initial rule $\langle \emptyset : do(E_1) \Rightarrow E_2 \rangle$ with an empty context.

The set of rules grows out of the motor variables. To create a rule $\langle \emptyset : do(E_1) \Rightarrow E_2 \rangle$ we require that the agent be able to predict event E_1 using the currently existing rules.

Rule Context Greedy Search

The purpose of the context is to tell the agent when the rule will succeed and the agent greedily searches for a good context for each rule. A good context $\mathcal{C}$ for r is one for which there is some value for the variables in $\mathcal{C}$ for which the rule has high reliability. Once this is achieved, the agent desires that the context predict the outcome of the rule in all states.

We formalize the idea of having a value for which the rule is highly reliable using the notation of best reliability $brel(r)$. For a context $\mathcal{C} = \{v_1, \ldots, v_n\}$, we define the product space of qualitative values:

$$\mathcal{Q}(\mathcal{C}) = \mathcal{Q}(v_1) \times \mathcal{Q}(v_2) \times \ldots \times \mathcal{Q}(v_n). \quad (1)$$

With sufficient observations, we then define best reliability as the maximum over this product space

$$brel(r) = \max_{w \in \mathcal{Q}(\mathcal{C})} Pr(succeeds(r)|w) \quad (2)$$

which we can also write as $brel(r) = \max\ CPT(r)$.

Once $brel(r)$ exceeds the threshold $\theta_r = 0.7$ we determine that the rule is reliable. At this point the agent turns its attention to being able to predict the outcome of a rule in any situation. To do this it seeks to minimize the entropy. The *entropy* $H(Y)$ of a random variable is given by

$$H(Y) \quad = \quad -\sum_j P(Y = y_j) \log_2 P(Y = y_j).$$

The conditional entropy $H(Y|X)$ of a random variable Y given X is given by

$$H(Y|X) = \sum_i H(Y|X = x_i) P(X = x_i)$$

and is the weighted average of the entropy of Y given $X = x_i$, weighted by the probability $P(X = x_i)$. We define the entropy $H(r)$ of a rule $r = \langle \mathcal{C} : do(E_1) \Rightarrow E_2 \rangle$ as the conditional entropy of $succeeds(r)$ given $do(E_1) = \mathsf{true}$ and $\mathcal{C}$. In equation form it is

$$H(r) = H(succeeds(r)|do(E_1) = \mathsf{true}, \mathcal{C}). \quad (3)$$

With these definitions we can now describe how the agent determines if one context is better than another. For each rule the agent hillclimbs on the quality of the context. For a rule $r = \langle \mathcal{C} : do(E_1) \Rightarrow E_2 \rangle$ with $brel(r) < \theta_r$ we say that the rule $r' = \langle \mathcal{C}' : do(E_1) \Rightarrow E_2 \rangle$ with improved context $\mathcal{C}'$ is a *sufficient improvement* over r if $brel(r') >> brel(r)$. And for a rule $r = \langle \mathcal{C} : do(E_1) \Rightarrow E_2 \rangle$ with $brel(r) > \theta_r$ we say that the rule $r' = \langle \mathcal{C}' : do(E_1) \Rightarrow E_2 \rangle$ with improved context $\mathcal{C}'$ is a *sufficient improvement* over r if $H(r') << H(r)$, where the operators $>>$ and $<<$ mean sufficiently less than and sufficiently greater than, respectively.

Learning a Context for a Predictive Rule

The context for a predictive rule is learned incrementally. For each rule $r = \langle \emptyset : do(E_1) \Rightarrow E_2 \rangle$ with an empty context, the agent searches for a magnitude or nominal variable v_1 such that if r is modified to be $r' = \langle \{v_1\} : do(E_1) \Rightarrow E_2 \rangle$ then r' is a sufficient improvement.

Using an approach inspired by Drescher (1991), once the agent has learned a rule $r' = \langle \{v_1\} : do(E_1) \Rightarrow E_2 \rangle$ it searches for another discrete magnitude or nominal variable v_2 such that if r' is modified to be $r'' = \langle \{v_1, v_2\} : do(E_1) \Rightarrow E_2 \rangle$ then r'' is a sufficient improvement. This criterion clearly generalizes, but in our current implementation we limit the size of the context to two.

Learning New Landmarks

Inserting a new landmark x^* into (x^*_i, x^*_{i+1}) allows that interval to be replaced in $\mathcal{Q}(x)$ by two intervals and the dividing landmark: (x^*_i, x^*), x^*, (x^*, x^*_{i+1}). Adding this new landmark into the quantity space $\mathcal{Q}(x)$ allows a new distinction to be made that may transform a rule r into a new rule r'. A new landmark can be learned either by improving a predictive rule or by creating an event that reliably precedes another event.

Landmarks that Improve Rules Landmark candidates are generated for a rule $r = \langle \mathcal{C} : do(A \rightarrow a) \Rightarrow B \rightarrow b \rangle$ using the success or failure of r as a reward signal. A landmark candidate for r is adopted if it sufficiently improves r. A landmark can improve r by refining the event $A \rightarrow a$ or by refining a variable in $\mathcal{C}$.

To learn new landmarks it is not necessary to store the entire history. Instead, we only store the real values of all the variables for the last 200 activations of each rule. Landmark candidates are chosen considering the number of data points in the interval and the highest gain (Fayyad & Irani 1993). Depending on the distance from the new landmark x^* to the maximum and minimum observed values of x, this search can result in either a precise numerical value, or a range of possible values for x^* on different occasions: $range(x^*) = [lb, ub]$.

Landmarks Suggested by Events A landmark x^* is created for a variable x if it is estimated that the event $x \rightarrow x^*$ will reliably predict some other event E. To find this landmark, for each event E a histogram is maintained for each continuous variable $\tilde{x}$. Each time E occurs the histogram is updated with the current value of $\tilde{x}$. One or more landmark candidates is created for $\tilde{x}$ when the distribution of $\tilde{x}$ when

E occurs is significantly different from the background distribution of $\tilde{x}$. The location of each landmark x^* is taken to be the middle of a histogram bucket where the difference is the greatest.

Acting in the World

The Controller

The controller enables the agent to learn efficiently by actively choosing rules to test. In Mugan and Kuipers (Mugan & Kuipers 2007a) active learning was motivated by the desire to achieve certain goals, in this paper the motivation for active learning is improving the reliability of rules.

Choosing a Rule to Invoke The controller chooses a rule to invoke based on its weight w. The weight of a rule r consists of two components w_1 and w_2, and $w = max(\epsilon, w_1 w_2)$, where $\epsilon = 0.001$. If we use the notation $Pr(succeeds(r)|\mathcal{C}, s)$ to mean the probability of success of r in the current state s, then the component $w_1 = Pr(succeeds(r)|\mathcal{C}, s)$. The component w_2 reflects the rate at which the reliability of the rule is increasing, inspired by the "curiosity drive" of Oudeyer and Kaplan [2004].

Invoking a Rule Once the rule $r = \langle \mathcal{C} : do(A \rightarrow a) \Rightarrow B \rightarrow b \rangle$ has been chosen, the agent forms a plan to achieve $A \rightarrow a$. To do this, the controller examines the context $\mathcal{C}$. We say that the context is *satisfied* if in the current state the context says the rule will be *sufficiently reliable*, where sufficiently reliable means that $Pr(succeeds(r)|\mathcal{C}) > \theta_{sr}$, where $\theta_{sr} = 0.60$. There are three cases:

1. The context is satisfied.
2. The context is not satisfied and consists of only one variable.
3. The context is not satisfied and consists of more than one variable.

If the context is satisfied, the agent sets the goal to be $do(A \rightarrow a)$. If the context is not satisfied but consists of only a single variable V, then the agent sets the goal to be $do(V \rightarrow v)$ where $v \in \mathcal{Q}(V)$ has the highest $Pr(succeeds(r)|V = v)$. If $do(V \rightarrow v)$ is successful, the agent then sets the goal to be $do(A \rightarrow a)$. If the context is not satisfied and consists of more than one variable, then the agent sets the goal to be any member of the set $Good(CPT(r))$ defined in equation (4) (if the set $Good(CPT(r))$ is empty then the context is ignored). Once this goal is achieved the agent sets the goal to be $do(A \rightarrow a)$.

Backchaining Actions

Goals of the form $do(Y \rightarrow y)$ are achieved through backchaining. The approach to achieve a goal $do(Y \rightarrow y)$ depends on the type of variable Y. **(1)** If Y is a motor variable, then a random real value is picked from the range of the qualitative value y and the action is complete. **(2)** If Y is a direction of change or nominal variable, then the agent looks for a reliable rule of the form $r = \langle \mathcal{C} : do(X \rightarrow x) \Rightarrow Y \rightarrow y \rangle$ that in the current state is predicted to succeed with reliability θ_{sr} and invokes $do(X \rightarrow x)$. If no such rule is found or if r fails, then backchaining fails. **(3)** If Y is a magnitude variable then the agent uses a special rule of the form

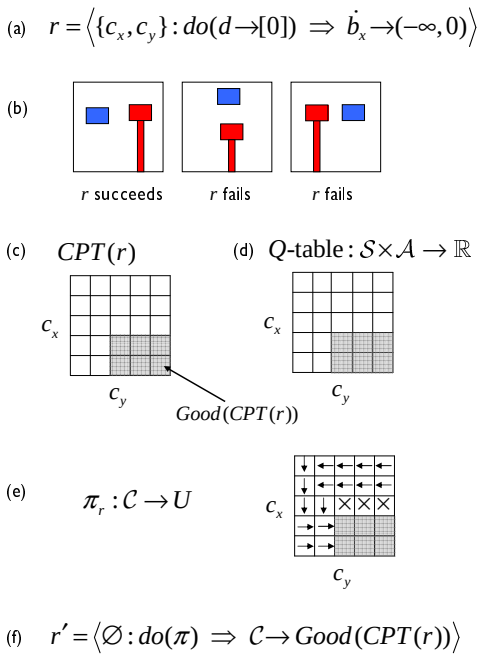


Figure 2: (a) The rule $r = \langle \{c_x, c_y\} : do(d \rightarrow [0]) \Rightarrow \dot{b}_x \rightarrow (-\infty, 0) \rangle$ is an example of a rule learned by the robot. It states that if the distance d between the hand and the block goes to 0, then the event $\dot{b}_x \rightarrow (-\infty, 0)$ of the block moving to the left will occur. The predicted success of this rule depends on the context variables c_x and c_y that give the location of the hand in the frame of reference of the block. (b) The agent gathers experience in the world to learn the context values for which r is successful. The agent learns that the hand must be to the right of and level with the block for r to be successful. (c) Based on $\mathcal{C} = \{c_x, c_y\}$ the agent creates a conditional probability table $CPT(r)$ for r and uses a threshold to determine the set $Good(CPT(r))$ of values of $\mathcal{C}$ for which the rule r is likely to succeed. (d) The agent can then define a simple reinforcement learning problem in which $\mathcal{C}$ defines the state space, and $Good(CPT(r))$ defines the goal states. The agent is rewarded for reaching a state in which the rule r is likely to succeed. To do this, the agent creates a Q-table that maps $\mathcal{S} \times \mathcal{A}$ to a value $\mathbb{R}$, where $\mathcal{A}$ is the set of primitive actions (defined by the qualitative values of the motor variables u_x and u_y). (e) The agent then defines a policy π_r by associating each cell in $\mathcal{C}$ with the primitive action with maximum value. (f) The policy π_r can then be described by a new rule r' that treats π_r as an action leading to the region $Good(CPT(r))$ where r is likely to succeed.

$h_1 = \langle do(\dot{Y} \rightarrow (0, +\infty)) \text{ until } Y \rightarrow y \rangle$ if $\tilde{Y}(t) < y$ and $h_2 = \langle do(\dot{Y} \rightarrow (-\infty, 0)) \text{ until } Y \rightarrow y \rangle$ if $\tilde{Y}(t) > y$. Rule h_1 fails if $do(\dot{Y} \rightarrow (0, +\infty))$ is not achieved or if after $\dot{Y} \rightarrow (0, +\infty)$ an event occurs such that $\dot{Y} \neq (0, +\infty)$ before $Y \rightarrow y$. Rule h_2 works similarly. If during backchaining an event E occurs more than once, or if events $v \rightarrow (-\infty, 0)$ and $v \rightarrow (0, +\infty)$ both occur for some variable v, then backchaining fails.

Once a motor variable is reached, its value is maintained until event $Y \rightarrow y$ occurs or one of the rules used in backchaining fails.

Reinforcement Learning Actions

The agent uses reinforcement learning to achieve goals over multiple variables. For each rule $r = \langle \mathcal{C} : do(E_1) \Rightarrow E_2 \rangle$ with a context of more than one variable, the agent creates a reinforcement learning problem to enable the agent to get into a state such that doing event E_1 will cause event E_2. The overview of this process is shown in Figure 2.

The type of reinforcement learning we use is Sarsa(λ) (Sutton & Barto 1998) where $\lambda = 0.9$, α is one over the number of times the state has been visited, and the discount parameter $\gamma = 0.9$. To learn the policy π_r the agent learns a value-action function $Q : \mathcal{S}, \mathcal{A} \rightarrow \mathbb{R}$.

The state space $\mathcal{S}$ is defined by the qualitative variables in $\mathcal{C}$ and their landmarks. To define the set of primitive actions $\mathcal{A}$ we first define a set $\mathcal{Q}(U) = \mathcal{Q}(u_1) \times \ldots \times \mathcal{Q}(u_n)$ where $u_1, \ldots, u_n$ is the set of motor variables. We can then define a primitive action $a \in \mathcal{A}$ as choosing a $w \in \mathcal{Q}(U)$, taking random values from the ranges of the qualitative values in w, and maintaining those values until the state $\mathcal{S}$ changes or the real values underlying the variables that make up $\mathcal{S}$ stop changing.

For the reward function we use a goal-reward representation (Koenig & Simmons 1996). The reward is based on the set of goal states $Good(CPT(r))$ and is determined by the conditional probability table $CPT(r)$:

$$Good(CPT(r)) = \{w \in \mathcal{Q}(\mathcal{C}) \quad | Pr(succeeds(r)|w) > \theta_{sr}\} \tag{4}$$

The agent then learns the Q-table by using ϵ-greedy action selection where $\epsilon = 0.25$. An episode begins when the rule r is invoked by the controller, and the episode ends when the agent makes it to a goal state or when 20 primitive actions have been taken.

Once the Q-table is learned, a policy π_r can be created whereby the agent chooses the best primitive action for each state. In effect, this in principle leads to a new rule of the form $r' = \langle \emptyset : do(\pi_r) \Rightarrow \mathcal{C} \rightarrow Good(CPT(r)) \rangle$.

Experimental Evaluation

We evaluate our algorithm using the simulated agent shown in Figure 3. The evaluation task we have chosen is for the agent to hit the block in a specified direction. To show that our representation can effectively be used for reinforcement learning, we compare our method to using a hand-created reinforcement learning agent trained specifically for this task. For this evaluation we trained ten agents total, five autonomous agents described in this paper, and five hand-created learning agents.

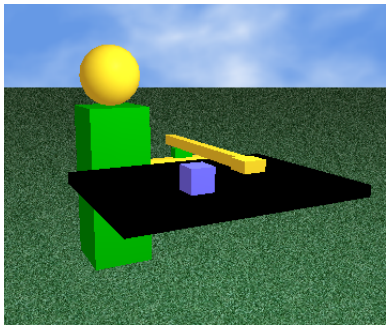


Figure 3: A simulated "robot baby" is implemented in Breve (Klein 2003). It has a torso with a 2-dof orthogonal arm and is sitting in front of a tray with a block. The robot has two motor variables $\tilde{u}_x$ and $\tilde{u}_y$ that move the hand in the x and y directions, respectively. The perceptual system creates variables for each of the two tracked objects in this environment: the hand and the block. The hand is described by two continuous variables $\tilde{h}_x(t)$, $\tilde{h}_y(t)$ that represent the location of the hand in the x and y directions, respectively, and the Boolean variable $h_a(t)$ that represents whether the hand is in view. The variables corresponding to the block are $\tilde{b}_x(t)$, $\tilde{b}_y(t)$, and $b_a(t)$ and they have the same respective meanings as the variables for the hand. The relationship between the hand and the block is represented by the continuous variables $\tilde{c}_x(t)$, $\tilde{c}_y(t)$, and $\tilde{d}(t)$. The variables $\tilde{c}_x(t)$ and $\tilde{c}_y(t)$ represent the coordinates of the center of the hand in the frame of reference of the center of the block, and the variable $\tilde{d}(t)$ represents the distance between the hand and the block. The values of all variables are updated by perceptual trackers at each timestep as the objects move.

We trained each agent in the environment shown in Figure 3 for 340,000 timesteps (almost five hours of physical experience). During this time, the hand-created agents continually repeated episodes of the task, and the autonomous agents performed the learning algorithm described in this paper. During training of the autonomous agents, if the block fell off the tray, moved out of reach of the agent, or was not moved for an extended time, the block was moved to a random location within reach of the agent. For all agents we stored the state of each agent's knowledge every 20,000 timesteps during training (corresponding to about sixteen minutes of physical experience). We then ran the evaluation for each agent using their respective stored knowledge bases.

Each evaluation consisted of 100 trials. At the beginning of each trial the block was placed in a random location within reach of the agent and the evaluator picked one of three goals: hitting the block to the left, hitting the block to the right, or hitting the block forward. The agent then had 300 timesteps to use its knowledge to hit the block in the correct direction. A trial was terminated unsuccessfully if the agent hit the block in the wrong direction. The evalua-

tion metric was the success rate for hitting the block during the 100 trials.

We tested both types of agents under two goal selection regimes, *uniform* and *hard*. During both uniform and hard goal selection, the evaluator selects the goal randomly, with a uniform distribution, and filters out goals that are impossible to achieve. (For example, if the block is on the far left, the agent cannot get its hand on the left side of the block to move it to the right.) During *hard* goal selection, *easy* goals are also filtered out, where a goal is easy if it can be achieved with a single straight-line motion. (During the training period for the hand-created agents, a goal selection regime was randomly chosen at the beginning of each episode, and then the goal was chosen based on that regime.)

The Hand-Built Learner

The hand-built reinforcement learning agents used linear, gradient-descent Sarsa(λ) with binary features (Sutton & Barto 1998) where the binary features come from tile coding. We chose this method because tile coding is a standard method for coping with continuous variables in reinforcement learning (Santamaria, Sutton, & Ram 1997). Tile coding works by using multiple partitions of the state space such that each partition (tiling) is offset just a little from the others. This allows the agent to generalize more effectively than using a single partition with higher resolution.

We now explain the details of the tile coding implementation. The motor variables u_x and u_y were each divided into 10 equal-width bins, and the direction of change variables were each divided into 3 bins: $(-\infty, -0.05), [-0.05, 0.05], (0.05, \infty)$. The goal was represented with a discrete variable that took on three values, one for each of the three goals. The remaining variables were treated as continuous. There were 16 tilings, the tiling was done using a hashing function with a memory size of 65,536. The parameter values used were $\lambda = 0.9$, $\gamma = 0.9$, and $\alpha = 0.1$. To prevent the task diameter from being too high, during both training and testing the agent chose a new action every 10 timesteps (0.5 seconds). Action selection was ϵ-greedy where $\epsilon = 0.05$.

Results

The results are shown in Figure 4. As the agent gains more experience in the world its ability to perform the task improves. We also see that the agent has indeed learned the action as its performance under both the difficult and uniform task selection regime is comparable to that of the hand-created learner.

The hand-created learner enjoys the advantage of only being trained on the evaluation task. But the hand-created learner is at an important disadvantage, it does not know which variables are important for the task. Our agent learns which variables are important autonomously, and this allows it to perform comparably even though it learns more than how to perform the evaluation task. For example, our agent learns when the block will disappear off the tray. It also learns the limits of its movement, and it can also move away from the block instead of towards it. It can use the knowledge learned during one task to learn another. Of particular

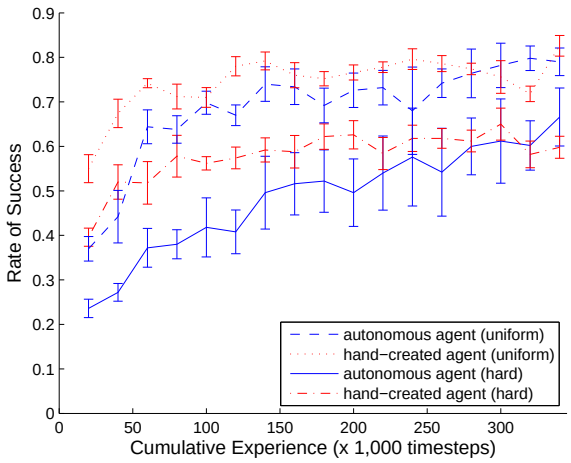


Figure 4: Both the autonomous agent and the hand-created agent improve with experience. The hand-created agent training only on the task initially has better performance, but the autonomous agent later matches it. All error bars are standard error.

importance is that the agent learns a discretization of the action space. Each motor variable is initially given a landmark at 0, but it takes a force of 300 in the simulator to move the arm in any direction. Our agent finds those important landmarks and can then use that knowledge when learning a new task. In contrast, the hand-created learner would need to be trained from scratch for each new task and would not be able to use what it learned in previous tasks for future tasks.

Conclusion and Future Work

The agent begins with a simple set of qualitative distinctions. These distinctions allow it to learn predictive rules. By monitoring the success or failure of these rules, the agent is able to find natural joints (landmarks) in its environment that allow it to discretize its continuous input and motor variables. Using the rules and the discretization of variables, the agent is able to define many small reinforcement learning problems. These reinforcement learning problems lead to policies that allow the agent to move to the regions of the state space that maximize the reliability of its learned rules.

In future work we will focus on generalizing the notion of a rule to include policies learned during reinforcement learning as shown in Figure 2 (f). This will allow the agent learn when such policies will be successful and when they will not. We are also moving towards implementing this on a physical robot using a camera that watches an arm in a workspace.

References

Drescher, G. L. 1991. *Made-Up Minds: A Constructivist Approach to Artificial Intelligence*. Cambridge, MA: MIT Press.

Fayyad, U. M., and Irani, K. B. 1993. Multi-interval discretization of continuousvalued attributes for classification

learning. In *Proceedings International Joint Conference on Articial Intelligence*, volume 2, 1022–1027.

Klein, J. 2003. Breve: a 3d environment for the simulation of decentralized systems and artificial life. In *Proceedings of the International Conference on Artificial Life*, 329–334.

Koenig, S., and Simmons, R. 1996. The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithms. *Machine Learning* 22(1):227–250.

Kuipers, B. 1994. *Qualitative Reasoning*. Cambridge, Massachusetts: The MIT Press.

McGovern, A., and Barto, A. G. 2001. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings International Conference on Machine Learning*, 361–368.

Mugan, J., and Kuipers, B. 2007a. Learning distinctions and rules in a continuous world through active exploration. In *Proceedings of the International Conference on Epigenetic Robotics*.

Mugan, J., and Kuipers, B. 2007b. Learning to predict the effects of actions: Synergy between rules and landmarks. In *Proceedings of the International Conference on Development and Learning*.

Oudeyer, P.-Y., and Kaplan, F. 2004. Intelligent adaptive curiosity. In *Proceedings of the International Conference on Epigenetic Robotics*.

Pearl, J. 2000. *Causality: Modeling, Reasoning, and Inference*. Cambridge: Cambridge University Press.

Sachenbacher, M., and Struss, P. 2005. Task-dependent qualitative domain abstraction. *Artificial Intelligence* 162(1-2):121–143.

Santamaria, J.; Sutton, R.; and Ram, A. 1997. Experiments with Reinforcement Learning in Problems with Continuous State and Action Spaces. *Adaptive Behavior* 6(2):163.

Simsek, O., and Barto, A. 2004. Using relative novelty to identify useful temporal abstractions in reinforcement learning. *Proceedings of the Twenty-First International Conference on Machine Learning* 751–758.

Simsek, O.; Wolfe, A.; and Barto, A. 2005. Identifying useful subgoals in reinforcement learning by local graph partitioning. *Proceedings of the Twenty-Second International Conference on Machine Learning* 816–823.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning*. Cambridge MA: MIT Press.

Interval analysis based learning for fault model identification. Application to control surfaces oscillatory failures.

Renaud Pons[*], Carine Jauberthie[†*] , Louise Travé-Massuyès[*], Philippe Goupil[‡]

{rpons,cjaubert,louise}@laas.fr
[*]LAAS-CNRS
[*]Université de Toulouse
[†]Université Paul Sabatier
7, avenue du Colonel Roche
F-31077 Toulouse
France

philippe.goupil@airbus.com
[‡]Airbus France
Flight Control System department
316, route de Bayonne
31060 Toulouse CEDEX 09
France

Abstract

Interval models may be seen as a trade-off between numerical and qualitative models. They have been often referred as semi-qualitative models. The interval algebra is indeed a specific qualitative algebra with advantageous algebraic properties. This paper presents the application of an interval based parameter estimation method, which is used for learning fault models supporting the detection of Oscillatory Failure Cases (OFC) in Electrical Flight Control System (EFCS) of civil airplanes. The interval estimation method results are guaranteed and computations are performed in finite time. Failures are identified using the fault models which are checked against system input and output measurements.

Introduction

Model based reasoning relies on the soundness of the models supporting the reasoning. This is particularly true for model based fault detection and diagnosis. Nevertheless building models turns out to be an awkward task. At some stage of the process, one may face two kinds of uncertainties. On one side, *unstructured* uncertainties mean that deriving a complete equational model from the physical phenomena is impossible. On the other side, when the structure of the equations is known but some of the parameters are not, uncertainties are said to be *structured*. In addition to these uncertainties, it is not always possible to get informations about disturbances and noises acting on the system. In such cases, assuming bounded uncertainties may be a solution.

Considering structured uncertainties, an interesting way to go is then to use guaranteed estimation methods, which learn the state and/or parameters of the models from data. These methods rely on *interval analysis* that first appeared in (Moore 1966). They are now subject of a growing interest in various communities and are applied for many tasks (Alamo, Bravo, & Camacho 2005; Armengol *et al.* 2001; Guerra, Puig, & Ingimundarson 2006; Jaulin *et al.* 2001; Kieffer & Walter 1998; Kieffer, Jaulin, & Walter 2002; Lesecq, Barraud, & Dinh 2003; Ribot 2006; Ribot, Jauberthie, & Travé-Massuyès 2007).

This paper presents a fault detection method using interval parameter estimation. Parameters of the model are estimated from the input and output measurements of the system. The consistency of this estimation is then checked against parameters computed from a theoretical (possibly faulty) model of the system. Computations use the set inversion algorithm SIVIA (Jaulin & Walter 1993; Jaulin *et al.* 2001). The results are approximated but are bounded in a guaranteed way. The method is applied to detect Oscillatory Failure Cases (OFC) in Electrical Flight Control System (EFCS) of civil airplanes.

The article is organised as follows. Next section positions interval models with respect to qualitative models. Then second section provides an overview of interval analysis, its original purpose and its use for fault detection. The error bounded context is then presented more precisely with parametric estimation using intervals in the fourth section. In fifth section, the case study is presented: we describe what are OFC, and their consequences on the aircraft control surfaces, why such failures must be detected in time and one of the methods currently used on Airbus aircrafts for OFC detection. In sixth section the application and the obtained results are analyzed. Finally some conclusions are outlined in last section.

Qualitative versus interval models

Providing models representing physical systems is a common concern spread over all scientific and engineering communities. Modelling depends on the available knowledge about the physical system. This is why pure numerical models are sometimes disregarded to the benefit of qualitative models which naturally cope with uncertain and inaccurate knowledge. Within the qualitative framework, numerical values are replaced by qualitative values that can be seen as (absolute) *orders of magnitude*[1].

Absolute orders of magnitude are based on partitioning the real line $\mathbb{R}$ into a finite set of basic qualitative val-

[1]Relative orders of magnitude refer to different formalisms based on binary relations used to compare quantities (Dague 1993a; 1993b; Travé-Massuyès *et al.* 2005).

ues. Considering the order relation given by set inclusion, it allows one to build the whole set of qualitative values, organised along to a high semi-lattice (Travé-Massuyès & Piera 1989; Travé-Massuyès, Ironi, & Dague 2003). As an example, (De Kleer & Brown 1984; Forbus 1984; Kuipers 1984) introduced *sign algebra* for which a parameter or a variable x takes values in $\{-, 0, +, ?\}$ depending on whether it is negative, zero, positive, or undetermined. Unfortunately, many operations, *e.g.* $(+) - (+)$, lead to an undetermined result. Absolute order of magnitude algebras were proposed to hinder this problem (Travé-Massuyès, Ironi, & Dague 2003). The real line partitioning defines the *quantity space* of a variable thanks to *landmark values* (Kuipers 1994). It captures the intuition that there are only a few qualitative important values associated to different qualitative behaviors. Whatever partitioning is chosen, an algebra and arithmetical operations can be defined.

The interval algebra can be seen as an extreme case in which the partition elements are provided by every real number and intervals are closed and connected subsets of $\mathbb{R}$. Interval analysis may then be interpreted as a specific case of order of magnitude reasoning.

Interval analysis

Preamble

The key idea of interval analysis is to reason about intervals instead of real numbers and boxes instead of real vectors. The first motivation was to obtain guaranteed results from floating point algorithms and it was then extended to validated numerics (Moore 1959). Let us recall that in computers real numbers can only be represented by a floating point approximation, hence introducing a quantification error. A *guaranteed result* means first that the result set encloses the exact solution. The width of the set, *i.e.* the result precision, may be chosen depending on various criteria among which response time or computation costs. Secondly, it also means that the algorithm is able to conclude on the existence or not of a solution in limited time or number of iterations. The first significant work is due to Moore in its Phd thesis which was the early beginnings of his reference book (Moore 1966).

Main concepts

The matter is to wrap the sets of interest into boxes or union of boxes for which computations may be easier. There are three fundamental operations on intervals which are briefly explained after the definition of an interval.

Interval A real interval $[u] = [\underline{u}, \overline{u}]$ is a closed and connected subset of $\mathbb{R}$ where $\underline{u}$ represents the lower bound of $[u]$ and $\overline{u}$ represents the upper bound. The width of an interval $[u]$ is defined by $w(u) = \overline{u} - \underline{u}$, and its midpoint by $m(u) = (\overline{u} + \underline{u})/2$.

The set of all real intervals of $\mathbb{R}$ is denoted $\mathbb{IR}$.

Two intervals $[u]$ and $[v]$ are equal if and only if $\underline{u} = \underline{v}$ and $\overline{u} = \overline{v}$. Real arithmetic operations are extended to intervals (Moore 1966).

Arithmetic operations on two intervals $[u]$ and $[v]$ can be defined by:

$$\circ \in \{+, -, *, /\},\ [u] \circ [v] = \{x \circ y \mid x \in [u],\ y \in [v]\}.$$

An interval vector (or box) $[X]$ is a vector with interval components and may equivently be seen as a cartesian product of scalar intervals:

$$[X] = [x_1] \times [x_2] \times \ldots \times [x_n].$$

The set of $n-$dimensional real interval vectors is denoted by $\mathbb{IR}^n$.

An interval matrix is a matrix with interval components. The set of $n \times m$ real interval matrices is denoted by $\mathbb{IR}^{n \times m}$. The width $w(.)$ of an interval vector (or of an interval matrix) is the maximum of the widths of its interval components. The midpoint $m(.)$ of an interval vector (resp. an interval matrix) is a vector (resp. a matrix) composed of the midpoint of its interval components.

Classical operations for interval vectors (resp. interval matrices) are direct extensions of the same operations for punctual vectors (resp. punctual matrices) (Moore 1966).

Wrappers Consider a set $\mathbb{U}$ and a set $\mathbb{V}$ of subsets of $\mathbb{U}$. $\mathbb{V}$ is a *set of wrappers* for $\mathbb{U}$ if $\mathbb{U}$ and each singleton of $\mathbb{U}$ belong to $\mathbb{V}$ and $\mathbb{V}$ is closed by intersection.

The figure 1 shows $f([u])$ which is the direct image of a box $[u]$ in $\mathbb{IR}^2$ by a function f, a possible wrapper $[f]([u])$ and the optimal wrapper $[f]^*([u])$. $f([u])$ is called the *range* of f over $[u]$ and is given by:

$$f([u]) = \{f(x) \mid x \in [u]\}.$$

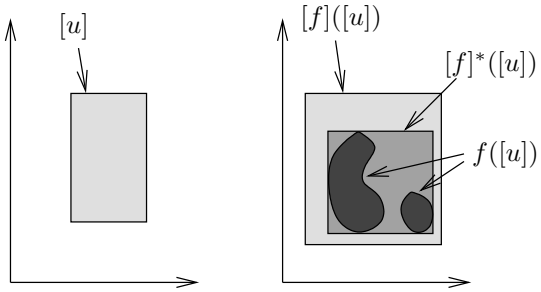


Figure 1: Range of f over $[u]$ and wrappers.

Inclusion function Given $[u]$ a box of $\mathbb{IR}^n$ and a function f from $\mathbb{IR}^n$ to $\mathbb{IR}^m$, the *inclusion function* of f aims at getting an interval containing the image of $[u]$ by f.

An inclusion function of f can be obtained by replacing each occurrence of a real variable by its corresponding interval and by replacing each standard function by its interval evaluation. Such a function is called the natural inclusion function. In practice the inclusion function is not unique, it depends on the syntax of f.

Inclusion test Given a subset $\mathbb{S}$ of $\mathbb{R}^n$, we test if $[x]$ belongs to $\mathbb{S}$, more precisely if $[x] \subset \mathbb{S}$ or $[x] \cap \mathbb{S} = \emptyset$. These tests are used to prove that all points in a given box satisfy a given property or to prove that none of them does.

Contractor The last operation is the *contraction* of $[x]$ with respect to $\mathbb{S}$. This means that we search a smaller box $[z]$ such that $[x] \cap \mathbb{S} = [z] \cap \mathbb{S}$. If $\mathbb{S}$ is the feasibility set of a problem and $[z]$ turns out empty, then the box $[x]$ may not contain the solution (Jaulin *et al.* 2001).

These operations are used to test if a box can or cannot be removed from the solution set. When no conclusion can be drawn, the box may be bisected and each of the sub-boxes can be tested in turn (this corresponds to *branch-and-bound* algorithms).

SIVIA: Set Inversion Via Interval Analysis

Consider the problem of determining a solution set for the unknown quantities u defined by

$$\begin{aligned} S &= \{u \in U \mid \Phi(u) \in [y]\}, \\ &= \Phi^{-1}([y]) \cap U, \end{aligned} \tag{1}$$

where $[y]$ is known a priori, U is an a priori search set for u and Φ a nonlinear function not necessarily invertible in the classical sense. (1) involves computing the reciprocal image of Φ. This can be solved using the algorithm *SIVIA*, which is a recursive algorithm that explores all the search space without loosing any solution. This algorithm makes it possible to derive a guaranteed enclosure of the solution set S as follows:

$$\underline{S} \subseteq S \subseteq \overline{S}. \tag{2}$$

The inner enclosure $\underline{S}$ is composed of the boxes that have been proved feasible. To prove that a box $[u]$ is feasible it is sufficient to prove that $\Phi([u]) \subseteq [y]$. Reversely, if it can be proved that $\Phi([u]) \cap [y] = \emptyset$, then the box $[u]$ is unfeasible. Otherwise, no conclusion can be reached and the box $[u]$ is said undetermined. The latter is then bisected in two sub-boxes that are tested until their size reaches a user-specified precision threshold $\varepsilon > 0$. Such a termination criterion ensures that *SIVIA* terminates after a finite number of iterations.

The algorithm is formally presented below. The functions $L(.)$ and $R(.)$ return respectively the "left" and "right" parts of their interval vector argument once it has been bisected. This bisection may be made using different strategies such as round robin, largest first or random.

Algorithm 1 SIVIA(in: Φ, $[y]$, $[u]$, ε, inout: $\underline{S}$, $\overline{S}$)

if $[\Phi]([u]) \cap [y] = \emptyset$ **then**
  return
end if
if $[\Phi]([u]) \subset [y]$ **then**
  $\underline{S} := \underline{S} \cup [u]$
  $\overline{S} := \overline{S} \cup [u]$
  return
end if
if $width([u]) < \varepsilon$ **then**
  $\overline{S} := \overline{S} \cup [u]$
end if
SIVIA(Φ, $[y]$, $L([u])$, ε, $\underline{S}$, $\overline{S}$)
SIVIA(Φ, $[y]$, $R([u])$, ε, $\underline{S}$, $\overline{S}$)

Fault detection using intervals

Set membership detection uses these concepts to perform state estimation and parameters estimation. In state estimation, a nonlinear dynamical model is approximated by a Taylor expansion (Rihm 1994; Berz & Makino 1998; Nedialkov, Jackson, & Pryce 2001) to compute a box enclosing all possible trajectories of the solution between two successive time steps t_j and t_{j+1}.

The fixed point and Picard-Lindelöf theorems prove the existence and uniqueness of the solution (Rihm 1994). The interval solution becomes obviously wider and wider at each iteration step: this drawback is known as the *wrapping effect*. Numerous methods may circumvent this pessimism: among them one is to use high order Taylor expansion, mean value forms, matrices preconditioning and a predictor-corrector approach (Corliss 1994; Nedialkov 1999; Neumaier 1990; Raïssi, Ramdani, & Candau 2004; Ramdani 1995; Rihm 1994).

Parameter estimation in a bounded error context

Parameters and state estimated from experimental measures are usually obtained within a stochastic framework in which known distribution laws are associated to interferences and noisy measurements. Oppositely, in the bounded error context measures and modeling errors are supposed to be unknown but to stay within known and acceptable bounds.

Errors between measured and predicted outputs may rely on many factors, among them: limited sensors accuracy, interferences, noise, structured uncertainties, ... Some are quantifiable, some are not. We consider here the quantifiable error e, which is added to the model output y. The experimental outputs y_{exp} are given by:

$$y_{\text{exp}}(t_j) = y(t_j) + e(t_j),\ 1 \leq j \leq n. \tag{3}$$

In our context, the error e is supposed to be within an interval whose lower bound is $e_{\min}$ and upper bound is $e_{\max}$. An allowable error set $\mathbb{E}$ may be defined as a set of constraints

$$\mathbb{E} = \{e(t_j) \mid e_{\min} \leq e(t_j) \leq e_{\max}\}. \tag{4}$$

These bounds may be considered constant over time as well as variable. They may be established from data given by constructors for electronic parts for example.

Our system has unknown but bounded initial conditions while input and output values are available at any time. The initial conditions belongs to a set, hence the model output y is also a set denoted $[y]$, as well as the error e which is a set $[e]$ that must be in the domain $\mathbb{E}$.

In the same way than for $[e]$, we define an allowable domain $\mathbb{Y}$ for model output $[y]$ such than

$$\begin{aligned} \mathbb{Y} &= \{[y] \mid [y] \subset [y_{\text{exp}}]\}, \\ &= \{[y] \mid [y] \subset [y - e_{\max}, y - e_{\min}]\}. \end{aligned} \tag{5}$$

Interval analysis is used to reject models that are not consistent with data and error bounds.

Numerous approaches have been tested with linear models: ellipsoid shaped methods (Milanese & Vicino 1991;

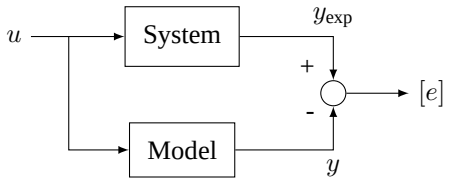


Figure 2: System and model.

Durieu & Walter 2001; Lesecq, Barraud, & Dinh 2003), parallelotopic and zonotopes (Alamo, Bravo, & Camacho 2005).

Consider a nonlinear parametric model described by the following set of equations

$$\begin{cases} \dot{x}(t,p) = f(x(t),u(t),p), \\ y(t,p) = g(x(t),u(t),p), \\ x_0 \in \mathbb{X}_0, \\ p \in \mathbb{P}_0, \end{cases} \tag{6}$$

where

- f and g are continuous nonlinear known functions,
- $x(t) \in \mathbb{R}^n$ is the state vector at time t,
- $u(t) \in \mathbb{R}^m$ is the input vector at time t,
- $y(t) \in \mathbb{R}^p$ is the output vector at time t,
- $\mathbb{X}_0$ is an a priori known set enclosing the initial condition x_0,
- $\mathbb{P}_0$ is the a priori known set enclosing the searched parameter vector p.

A parameter vector p is acceptable if and only if the error between y_{exp} and the model output $[y]$ is bounded in a known way. To estimate system parameters, we have to get the set $\mathbb{P}$ of all parameters p enclosed in the a priori search set $\mathbb{P}_0$ such that error between real data and model outputs denoted

$$[e(p)] = y_{\text{exp}} - [y(p)] \tag{7}$$

belongs to the allowable error set $\mathbb{E}$ whose bounds $e_{\min}$ and $e_{\max}$ are known:

$$\begin{aligned} \mathbb{P} &= \{p \in \mathbb{P}_0 \mid [e(p)] \in \mathbb{E}\}, \\ &= \{p \in \mathbb{P}_0 \mid e_{\min} \leq [e(p)] \leq e_{\max}\}. \end{aligned} \tag{8}$$

The characterization of the set $\mathbb{P}$ may be defined as a set inversion problem (Raïssi, Ramdani, & Candau 2003; Kieffer & Walter 2005):

$$\mathbb{P} = [e^{-1}](\mathbb{E}). \tag{9}$$

A guaranteed approximation of $\mathbb{P}$ may be computed using the SIVIA algorithm presented previously.

Case study

Problem

One of the tasks devoted to flight control computer is to slave the position of the control surfaces. The control surface motion is driven by an actuator in active or damped mode. There are generally two actuators for one control surface. A *master* computer performs control by sending a command on the active actuator. The other one is set in damped mode and follows the surface motion without opposition. When the master computer detects a failure, it switches the active actuator to damped mode and gives control to a *slave* computer that controls the second actuator which is now in active mode.

All parts in the control chain that contain electronic devices may generate interference signals. These signals make the control surface swing. This is called an *Oscillatory Failure Case (OFC)*. In this paper, only OFC located in the servo-loop control of the moving surfaces are considered, that is, between the *Flight Control Computer* and the control surface, including these two elements (*cf.* Figure 3). When an OFC occurs within the actuator bandwidth , it may have the following consequences:

- coupled with the aeroelastic behaviour of the aircraft, it may lead to unacceptably high loads or vibrations, the worst case corresponds to resonance phenomena with aircraft natural modes ;
- it speeds up actuators stress and reduces their lifetime ;
- it lowers passengers comfort.

The plane is designed to take into account these faults in a limited way, depending on oscillation frequency and range. Taking design actions to counteract these faults would indeed require heavily and costly structure reinforcement. It is then very much advisable to detect them using the flight control computers. Monitoring must be performed to ensure that failures stay within predefined limits. Classical monitoring (e.g. position monitoring, runaway monitoring, etc.) does not guaranty such detections, so specific mechanisms must be added.

When an OFC is detected, the flight computer looses regulation over elevators control. As seen previously, another waiting computer ensures surface control with a redundant servo which switches from damped to active mode.

The problem to solve is to detect in the control loop some OFC with a minimal given range within a given number of periods (the maximal overload does not immediately occur on the structure but after some periods of oscillation). The goal is to detect 1° failures within 3 periods, on a frequency range from 0.2 to 5 Hz. This goal has been chosen for this paper. In real cases, it depends on the aircraft type.

Liquid vs. solid failures

Two different kinds of OFC may occur: liquid or solid ones. As shown in the scheme of figure 4, a liquid failure is a interference signal added to the control loop signal. A solid failure is a signal which replaces the control loop one.

In both cases, a failure is a periodic sinus shaped signal whose frequency, range and phase obey to an uniform law. For both cases of failure, residuals corresponding to estimated position subtracted from real position are shown in Figure 5.

These residuals are used to detect the OFC. The current method used in A380 flight control computers relies

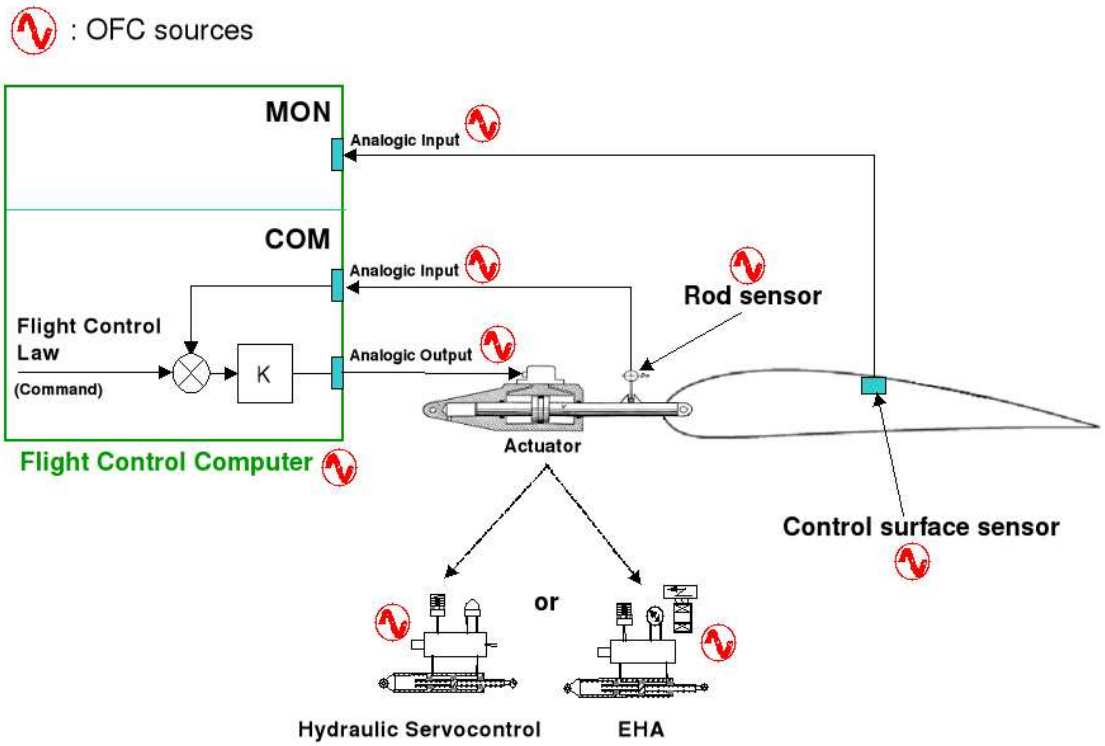


Figure 3: Position control chain.

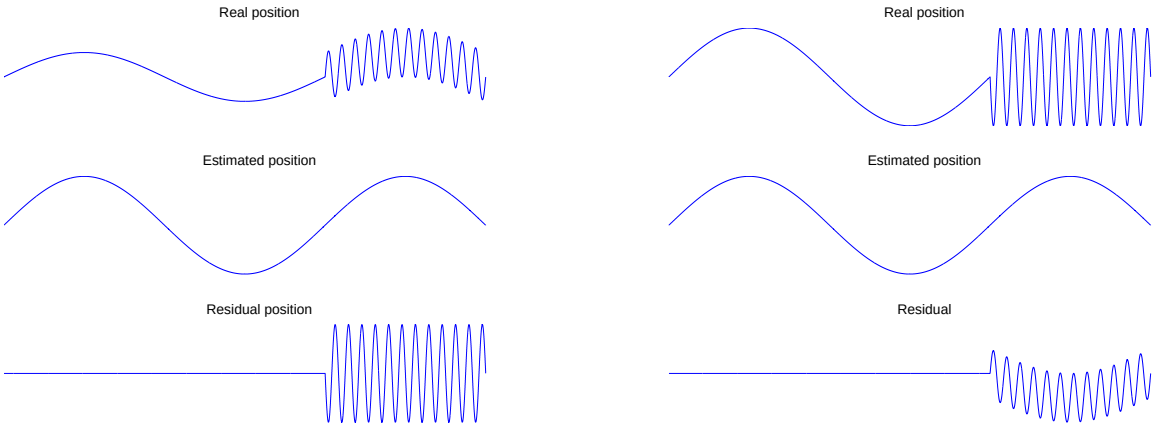


Figure 5: Residuals: liquid failure case on the left side, solid failure case on the right side.

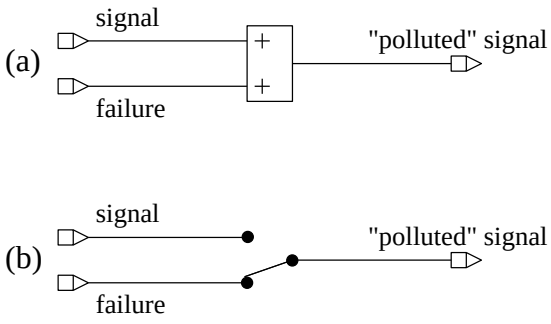


Figure 4: Liquid (a) vs. solid (b) failures.

on residual evaluation by oscillation counting inside spectral subband (Goupil 2007).

Application

In the following section, we address the case of liquid failure with the bounded error parameter estimation method presented previously and use this estimation for detecting OFC. The results are analyzed with respect to the currently used detection method.

Our goal is to perform parameter estimation of the liquid failure model. This fault model defines the shape of the position signal as either a sinus or a triangle. The system to monitor is a simple model of a control surface whose motion is ensured by a hydraulic servo command as presented in figure 6.

In this model, $o(n)$ is the position control signal at time n. The control error ε is given by:

$$\varepsilon(n) = o(n) - \hat{s}(n-1). \tag{10}$$

It is the difference between the position control o at time n and the estimated position $\hat{s}$ at time $n-1$. The estimated current $\hat{i}(n)$ is proportional to the error:

$$\hat{i}(n) = K\varepsilon(n) \tag{11}$$

where K is the constant control gain. A saturation is then applied to the current hence limiting its value within predefined bounds. It is then converted to speed $\hat{v}(n)$ by interpolation with data stored in a look-up table. Finally, the estimated control surface position $\hat{s}(n)$ at time n is computed by integration of the speed.

We ran tests introducing oscillatory failures in the control loop. Two fault models, triangle shaped and sinus shaped, were used. Parameters were estimated over one period of the signal.

Sinus shaped fault

A high noisy sinus-shaped liquid fault signal with a range $A = 1°$ and a frequency of $f = 0.5$Hz is introduced in the control surface model. The initial parameter box is given by $A \times f = [0,3] \times [0,10]$.

Figure 7 shows the results provided by the set inversion algorithm when the fault model is supposed to be sinus-shaped. Range parameter A is showed on the horizontal axis while frequency f is on the vertical one. Blue boxes have been rejected, yellow ones have a length inferior to the stop condition set in the algorithm. The red boxes represent the solution. We notice that they concentrate around the real parameter values.

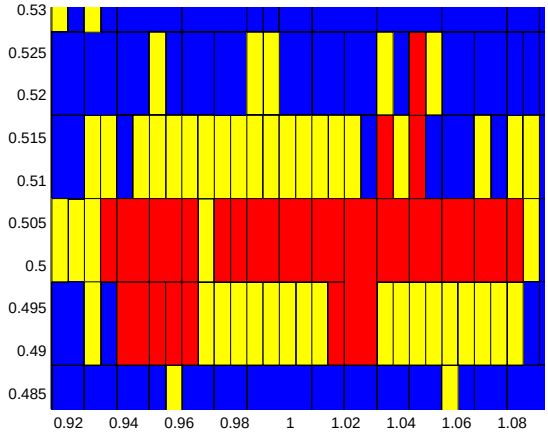


Figure 7: Sinus shaped fault.

When the fault model is triangle-shaped, the algorithm stops after a few iterations and its conclusion is the non-existence of a solution.

Triangle-shaped fault

In this example, the fault is triangle-shaped with a range 2° and a frequency of $f = 0.5$Hz, with a still highly noisy signal. The inital parameter box is now $A \times T = [0,3] \times [0,5]$, with $T = 1/f$.

Figure 8 exhibits the obtained results with a triangle-shaped fault model. The parameter A is on the horizontal axis and the period T on the vertical axis. One can notice that the estimation results are fully in accordance with the injected fault.

With a sinus-shaped fault model, the algorithm concludes again to the non-existence of a solution.

Discussion and conclusion

In this paper we presented a method for failure detection using fault models and an error bounded estimation method. The method is based on interval analysis which provides guaranteed results in an error bounded context. It has been applied to solve plane control surfaces oscillatory failures.

The tests show good results for confirming a fault. Now, the real advantage of the method with respect to others is that it is very efficient to prove the non-existence of the solution, that is to discard specific kinds of failures in the real system. In the two case study scenarios, the invalidation of the triangle-shaped (sinus-shaped) fault model is obtained within a few iterations. We should notice that a stochastic method would not invalidate the non relevant fault model but it would conclude to the existence of a solution with a wide confidence range, which is much more difficult to interpret.

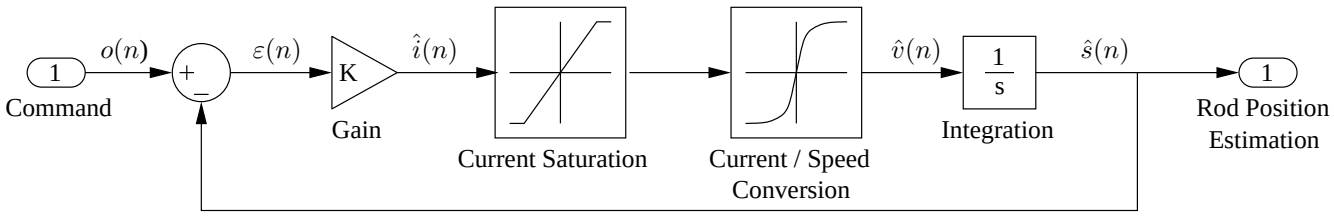


Figure 6: Control surface position estimation model.

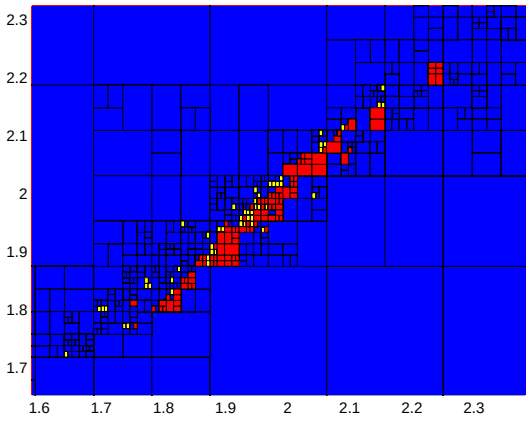


Figure 8: Triangle shaped fault.

Future work will consist in improving the fault model method by studying its properties : response time, false alarm rate, non detection rate and robustness. More simulation tests using alternate fault models against real data will also be performed.

Another direction to go is to use alternate detection methods under the condition to have proper surface control loop models. State estimation and parity state methods, both using interval analysis, should be tested.

References

Alamo, T.; Bravo, J.; and Camacho, E. 2005. Guaranteed state estimation by zonotopes. *Automatica* 41(6):1035–1043.

Armengol, J.; Vehi, J.; Travé-Massuyès, L.; and Sainz, M. A. 2001. Application of modal intervals to the generation of error-bounded envelopes. *Reliable Computing* 7(2):171–195.

Berz, M., and Makino, K. 1998. Verified integration of odes and flows using differential algebraic methods on high-order taylor models. *Reliable Computing* 4:361–369.

Corliss, G. 1994. Guaranteed error bounds for ordinary differential equations. In *Lectures notes at the VI-th SERC Numerical Analysis Summer School*.

Dague, P. 1993a. Numeric reasoning with relative orders of magnitude. In *11th National Conference on Artificial Intelligence, AAAI'93*, 541–547.

Dague, P. 1993b. Symbolic reasoning with relative orders of magnitude. In *International Joint Conference on Artificial Intelligence IJCAI'93*, 1509–1514.

De Kleer, J., and Brown, J. S. 1984. A qualitative physics based on confluences. *Artificial Intelligence* 24:7–83.

Durieu, C., and Walter, E. 2001. *Identification des systèmes*. Paris: Hermès. chapter Estimation ellipsoidales à erreur bornée.

Forbus, K. D. 1984. Qualitative process theory. *Artificial Intelligence* 24:85–168.

Goupil, P. 2007. Oscillatory failure case detection in a380 electrical flight control system by analytical redundancy. In *17th IFAC Symposium on Automatic Control in Aerospace*.

Guerra, P.; Puig, V.; and Ingimundarson, A. 2006. Robust fault detection with state estimators and interval models using zonotopes. In *17th International Workshop on Principles of Diagnosis, DX'06*, 109–116.

Jaulin, L., and Walter, E. 1993. Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica* 29(4):1053–1064.

Jaulin, L.; Kieffer, M.; Didrit, O.; and Walter, E. 2001. *Applied Interval Analysis, with examples in parameter and state estimation, Robust control and robotics*. Londres: Springer.

Kieffer, M., and Walter, E. 1998. *In A.C. Atkinson, L. Pronzato, H.P. Wynn, Advances Model-Oriented Data Analysis and Experimental Design*. Heidelberg: Physica-Verlag. chapter Interval Analysis for guaranteed nonlinear parameter estimation, 115–125.

Kieffer, M., and Walter, E. 2005. Interval analysis for guaranteed non-linear parameter and state estimation. *Mathematical and Computer Modelling of Dynamical Systems* 11(2):171–181.

Kieffer, M.; Jaulin, L.; and Walter, E. 2002. Guaranteed recursive nonlinear state bounding using interval analysis. *International Journal of Adaptative Control and Signal Processing* 6:191–218.

Kuipers, B. 1984. Commonsense reasoning about causality: deriving behavior from structure. *Artificial Intelligence* 24:169–203.

Kuipers, B. 1994. *Qualitative reasoning, modeling and simulation with incomplete knowledge*. Cambridge, Massachusetts: The MIT Press.

Lesecq, S.; Barraud, A.; and Dinh, K. T. 2003. Numerical accurate computations for ellipsoidal state bounding. In *11th IEEE Mediterranean Conference on Control and Automation, MED'03*.

Milanese, M., and Vicino, A. 1991. Estimation theory for non-linear models and set membership uncertainty. *Automatica* 27(2):403–408.

Moore, R. E. 1959. Automatic error analysis in digital computation. Technical Report LMSD-48421, Lockheed Missiles and Space Co, Palo Alto, CA.

Moore, R. 1966. *Interval Analysis*. Englewood Cliffs: Prentice-Hall.

Nedialkov, N. S.; Jackson, K.; and Pryce, J. 2001. An effective high-order interval method for validating existence and uniqueness of the solution of an ivp for an ode. *Reliable Computing* 7(6):449–465.

Nedialkov, N. 1999. *Computing rigorous bounds on the solution of an initial value problem for an ordinary differential equation*. PhD University of Toronto.

Neumaier, A. 1990. *Interval methods for systems of equations*. Cambridge, UK: Cambridge University Press.

Raïssi, T.; Ramdani, N.; and Candau, Y. 2003. Parameter estimation for nonlinear continuous-time systems in a bounded error context. In *42nd IEEE Conference on Decision and Control, CDC2003*.

Raïssi, T.; Ramdani, N.; and Candau, Y. 2004. Set membership staet and parameter estimation for systems described by nonlinear differential equations. *Automatica* 40:1771–1777.

Ramdani, N. 1995. Méthodes ensemblistes pour l'estimation. Habilitation à diriger des recherches.

Ribot, P.; Jauberthie, C.; and Travé-Massuyès, L. 2007. State estimation by interval analysis for a nonlinear differential aerospace model. In *European Control Conference, ECC'07*, 4839–4844.

Ribot, P. 2006. Estimation d'état et de paramètres en utilisant l'analyse par intervalles. Master recherche smis-eeas, Université Paul Sabatier, Toulouse, France.

Rihm, R. 1994. Interval methods for initial value problems in odes. In Herzberger, J., ed., *Topics in Validated Computations. IMACS-GAMM. International Workshop on Validated Computations*. Amsterdam, New York: University of Oldenburg.

Travé-Massuyès, L., and Piera, N. 1989. The order of magnitude models as qualitative algebras. In *11th International Joint Conference on Artificial Intelligence IJCAI'89*.

Travé-Massuyès, L.; Prats, F.; Sanchez, M.; and Agell, N. 2005. Relative and absolute order-of-magnitude models unified. *Annals of Mathematics and Artificial Intelligence* 45:323–341.

Travé-Massuyès, L.; Ironi, L.; and Dague, P. 2003. Mathematical foundations of qualitative reasoning. *AI Magazine* 24(4):91–106. Special Issue on Qualitative Reasoning.

Challenges in Presenting Argumentation Results

Laura Rassbach, Elizabeth Bradley

University of Colorado
Department of Computer Science
Boulder, CO 80309-0430
laura.rassbach@colorado.edu, lizb@cs.colorado.edu

Abstract

We present the initial user interface for the Calvin system. Despite designing for ease-of-use and simplicity, users had significant problems with this initial interface. Possible solutions to these problems are also presented.

1. Introduction

Displaying the results of a qualitative reasoning system to users in a useful and understandable way is almost as important as creating a system that generates the right results in the first place. Even when a system is capable of significantly aiding users with some task, they will tend to choose not to use it when the interface is baffling or impossible to use. However, designing a clear and comprehensible user interface is hardly a trivial task. We encountered significant challenges while designing a user interface for Calvin, an argumentation system for problems in cosmogenic isotope dating (a geological field). Although we believed that Calvin's initial interface would be clear and easy to use, we found that users struggled to understand the interface and did not use it as intended. Significant changes are now required to our initial interface. This paper discusses these challenges and our ongoing plans for improving the user experience with Calvin.

2. Calvin

Experts in cosmogenic isotope dating frequently need to identify what geologic processes are most likely to have affected a set of data they are examining. This process is difficult and time consuming, requiring significant amounts of expertise. Calvin is a qualitative reasoning system aimed at automating portions of this process. The nature of isotope dating (very little data and few unassailable interpretations of any data) makes it difficult to arrive at a definitive answer for any dataset, so Calvin generates a complete argument for each possible process. An argument is similar to a proof in first-order logic, but typically contains both evidence for and against the conclusion in question. Calvin is able to make judgments about which argument is 'best,' but it is not our goal to remove the expert from decision-making. Instead, we intend for the expert to be able to carefully examine Calvin's reasoning and make a final judgement about the process. Calvin's judgments about the best arguments are intended to guide experts' attention to the most likely possibilities.

Internally, the argument for a particular process is a collection of trees. The nodes in each tree are created from rules in Calvin's knowledge base, with every rule used in the argument represented as a node in one or more trees. The root of every tree in the collection for a particular process is a rule directly referring to that process. Because these argument trees are different from proofs in that they are defeasible, Calvin needs some way to judge the relative strengths of different support for a conclusion. Calvin uses a system of 2-element confidence vectors. The first element of the vector is called a 'match' and indicates how closely the current set of data matches Calvin's rule base. Match values are further composed of a truth and a degree. The 'truth' can be either true or false (matches or doesn't match the rule base) and the 'degree' refers to how far the actual data is from any threshold in the rule. For instance, if Calvin's rule base states that elevation greater than 10,000 feet is evidence for snow cover (one possible geologic process), elevation values of 1,000, 8,000, 11,000, and 15,000 feet would all generate different match values. The other dimension of confidence is a quality measure, and can be 'poor,' 'okay,' 'good,' or 'definite.' High elevation is only poor quality evidence for snow cover, whereas ages correlated with boulder size is good evidence (because smaller boulders will be covered with snow for more of the year).

3. User Interface 1.0

Figure 1 shows a screenshot of our initial user interface. We designed the initial user interface to display the complete set of arguments for each process in a simple and intuitive way. We expected this task to be relatively simple because experts quickly grasp Calvin's underlying logic when it is explained to them in an informal setting. When Calvin completes its analysis of a dataset, it brings up a window from which the user can access all the arguments

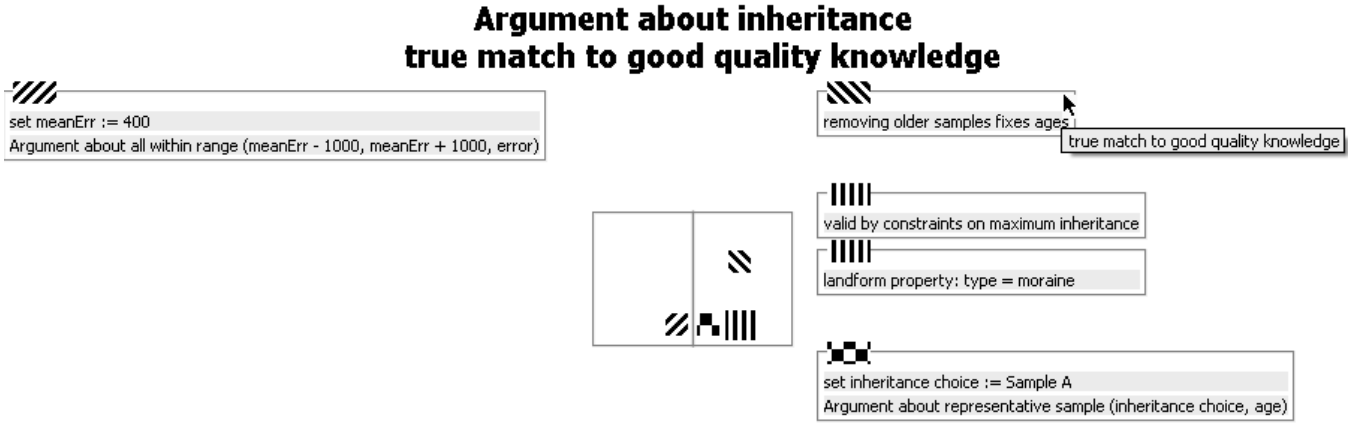


Figure 1: Screenshot of Calvin's initial user interface. Calvin's user interface has several components. A single screen is intended to describe the entirety of Calvin's knowledge about a particular conclusion. At the top of the screen is the conclusion under consideration and a gestalt of Calvin's confidence in that conclusion. Evidence related to the conclusion is divided into evidence against, on the left, and evidence for, on the right. This split is somewhat more apparent in the original because items on the left have red color markings and items on the right are green, however, in this screenshot colors have been replaced with textures for printing. Each distinct piece of evidence is enclosed in a box with text describing Calvin's reasoning. Every box and every text item has a tooltip allowing the user to view Calvin's confidence in these individual items. Users can also click on some text items to get more detail about Calvin's reasoning (either a subargument or the results of a simulation). Finally, the grid in the center provides a visual display of how many items of evidence Calvin has found at each 'level' of confidence: larger boxes represent more items.

it has generated. The first argument displayed is the one with the highest overall confidence.

The display for an individual argument includes the top-level conclusion being argued about (for instance, whether snow cover is a factor for this dataset) and the overall confidence in that conclusion. This confidence is converted directly into text from its internal representation (a typical confidence might read 'very true match to okay quality knowledge'). Evidence for and against the conclusion is sorted into two separate columns, sorted from strongest to weakest. Every piece of evidence contributes an individual confidence to the overall confidence in the conclusion. This contribution is displayed in two ways: as a color-coded block with each piece of evidence and as text available as a tooltip. Users can examine sub-arguments by clicking on their summary; they are displayed in the same format as the top-level arguments. Users can also get a more detailed description of any complex calculations performed by Calvin. For example, Calvin's rule base sometimes needs to determine whether a set of data points falls on a straight line. Users can click on this piece of evidence to get a description of how closely the points fit a line, the statistical significance of the fit, and a graph of the points and the best-fit line.

In the center of the argument display is a final representation of how the individual evidence items contribute to the Calvin's confidence in the overall argument conclusion. This representation is a grid with a location for each possible confidence value. Matches are listed from left to right, most extreme false through most extreme true, and quality is listed from top to bottom highest to lowest quality. Calvin sizes each of the grid cells according to the number of pieces of evidence it has at the corresponding confidence level for the argument. The cells are the same color as the block displayed with the evidence items.

4. Interface Weaknesses and Solutions

Observing experts using this initial interface has revealed a number of problems. The first and most obvious of these problems is that experts refer to Calvin's highest-confidence argument as the 'answer' and do not perform any further analysis when this answer is wrong. In particular, even when the confidence in the initially displayed argument is 'somewhat true match to poor quality knowledge' (the lowest possible 'true' confidence), users do not appear to make the inference that Calvin was unable to generate any good arguments for any process with the current data. Instead, they report a wrong answer and appear to make the inference that Calvin is quite certain of this answer.

An extension to this issue occurs when Calvin is able to generate quite good arguments for several processes. This often happens in datasets with very few samples. In this case, experts have expressed concern with the fact that Calvin came up with any answer at all, since there is quite good evidence for many processes. Instead of looking at the other arguments generated by Calvin, they look only at the best one, and appear to assume that all other arguments generated by Calvin must have been sufficiently inferior to reject them completely. Experts using Calvin appear generally uninterested in looking at the full spectrum of arguments it has generated.

We intend to address both of these problems by presenting a summary of all the arguments generated by Calvin. This will serve as the initial user view, rather than the single best argument generated. This will permit users to get a sense of perspective before viewing individual arguments in more detail.

Even while viewing a single argument in isolation, experts do not use Calvin's interface as intended. In particular, they seem almost uninterested in the evidence that has led Calvin to draw a specific conclusion. Although they appear to glance briefly at it, they seem not to understand it or to think critically about it without prompting. When using Calvin independently, experts did not choose to examine sub-arguments by clicking on them.

It is not yet clear precisely why users behave in this way. We believe that the size and color of the labels may make them difficult to read, and their positioning on the screen makes them less prominent. In addition, it has been observed by non-expert users that the text actually displayed in these labels is somewhat obscure and difficult to follow. We plan to make these descriptions of evidence physically easier to read and less obscure, but expect to need more iterations before experts are examining them closely and critically.

Finally, experts observed that the confidence system was somewhat confusing. This is especially true because understanding the report of a confidence level requires understanding Calvin's internal representation of confidence. However, even after discussing the meaning of various confidence values, experts seemed to have difficulty understanding what a specific confidence meant. Users did appear able to understand the centered, colored grid displaying the confidence contributed by all the evidence after an explanation of its purpose.

Clearly we need to alter Calvin's interface to make confidence more approachable to the user. We are considering a number of changes. The first and most simple is obviously to 'tweak' the language used to describe a confidence value to the user so it is more natural. If this proves to be impossible or insufficient, we may attempt to map Calvin's 2-dimensional confidence values onto a single dimension (e.g. 'no evidence,' 'little evidence,' 'some evidence,', etc.). In addition, we plan to experiment with the idea of never combining the confidence in evidence against some conclusion with evidence for it. Instead, confidence in the evidence for and against a conclusion could be presented separately. Doing this might have the additional benefit of encouraging users to examine the actual evidence that has produced these confidences.

5. Conclusion

Creating a usable and intuitive user interface for displaying the results of a qualitative reasoning system is an interesting challenge. Even when the underlying system is comparatively intuitive, conveying its results in a way that users understand is more complicated than simply placing them in a graphical interface. Despite our efforts, Calvin's user interface will require significant changes to meet our usability goals.

A Definition of Entropy based on Qualitative Descriptions

Llorenç Roselló and **Francesc Prats** and **Mónica Sánchez**
Polytechnical University of Catalonia, Barcelona, Spain
e-mail: {llorenc.rosello, francesc.prats, monica.sanchez}@upc.edu

Núria Agell *
Esade, Ramon Llull University, Barcelona, Spain
e-mail: nuria.agell@esade.edu

Abstract

A new concept of generalized absolute orders of magnitude qualitative spaces is introduced in this paper. The new structure makes it possible to define sets of qualitative labels of any cardinality, and is consistent with the classical structure of qualitative spaces of absolute orders of magnitude and with the classical interval algebra. In addition, the algebraic structure of these spaces ensures initial conditions for adapting measure theory to a qualitative environment. This theory provides the appropriate framework in which to introduce the concept of entropy and, consequently, the opportunity to measure the gain or loss of information when working within qualitative spaces. The results obtained are significant in terms of situations which arise naturally in many real applications when dealing with different levels of precision.

INTRODUCTION

Qualitative Reasoning (QR) is a subarea of Artificial Intelligence that seeks to understand and explain human beings' ability for qualitative reasoning (Forbus 1996), (Kuipers 2004). The main objective is to develop systems that permit operating in conditions of insufficient numerical data or in the absence of such data. As indicated in (Travè-Massuyès and Dague 2003), this could be due to both a lack of information as well as to an information overload.

A main goal of Qualitative Reasoning is to tackle problems in such a way that the principle of relevance is preserved; that is to say each variable has to be valued with the level of precision required (Forbus 1984). It is not unusual for a situation to arise in which it is necessary to work simultaneously with different levels of precision, depending on the available information, in order to ensure interpretability of the obtained results. To this end, the mathematical structures of Orders of Magnitude Qualitative Spaces (OM) were introduced.

*This work has been partly funded by MEC (Spanish Ministry of Education and Science) AURA project (TIN2005-08873-C02). Authors would like to thank their colleagues of GREC research group of knowledge engineering for helpful discussions and suggestions.

The word *information* appears constantly in QR. However, its meaning is as yet undefined within a qualitative context. The implicit and explicit use of the term and concept addresses the need to define and, perhaps paradoxically, to quantify them.

In this work it is presented a way of measuring the amount of information of a system when using orders of magnitude descriptions to represent it. Taking into account that the entropy can be used to measure the information, this work is intended to be a first step towards this measure by means of orders of magnitude qualitative spaces.

The concept of entropy has its origins in the nineteenth century, particularly in thermodynamics and statistics. This theory has been developed from two aspects: the macroscopic, as introduced by Carnot, Clausius, Gibbs, Planck and Caratheodory and the microscopic, developed by Maxwell and Boltzmann (Rokhlin 1967). The statistical concept of Shannon's entropy, related to the microscopic aspect, is a measure of the amount of information (Shannon 1948),(Cover and Thomas 1991).

In order to define the concept of information within the QR framework, this paper adapts the basic principles of Measure Theory (Halmos 1974), (Folland 1999) to give OM a structure in which to define the concept of entropy, and, consequently, the concept of information.

Section 2 defines the concept of generalized absolute orders of magnitude qualitative spaces. In Section 3, the algebraic structure of these spaces is analyzed in order to ensure initial conditions in which to adapt the Measure Theory. A measure and the concept of entropy in the generalized absolute orders of magnitude spaces are given in section 4 and 5 respectively. The paper ends with several conclusions and outlines some proposals for future research.

GENERALIZED ABSOLUTE ORDERS OF MAGNITUDE QUALITATIVE SPACES $\mathbb{S}_g^*$

The classical version of the qualitative orders of magnitude that appears in (Travè-Massuyès and Dague 2003) is an abstraction of intuitive concepts of "very small ","small","big", or "very hot", "hot", etc., i.e. an abstraction of concepts with which human beings reason. This abstraction is done through the introduction of *qualitative labels* in a way that defines a finite and discrete set of labels representing the above concepts. This paper proposes a further step towards the generalization of qualitative orders of magnitude. This generalization makes it possible to define orders of magnitude as either a discrete or continuous set of labels, providing the theoretical basis on which to develop a Measure Theory in this context.

Definition 1 *Let X be a non-empty set, I a subset of $\mathbb{R}$, and $B : I \to \mathcal{P}(X)$ an injective function. Then each $B(t) = B_t \subset X$ is a* generalized basic label on X *and the set $\mathcal{S}$ of* generalized basic labels on X *is*

$$\mathcal{S} = \{B_t \mid t \in I\}.$$

Note that if $t \neq t'$, then $B_t \neq B_{t'}$.

Definition 2 *If $i, j \in I$, with $i < j$, the* generalized non-basic label $[B_i, B_j)$ *is defined by*

$$[B_i, B_j) = \{B_t \mid t \in I, i \leq t < j\}.$$

In the case $i = j \in I$, the convention $[B_i, B_i) = \{B_i\}$ will be used. If necessary, $[B_i, B_i) = \{B_i\}$ can be identified with the basic label B_i.

Definition 3 *If $i \in I$, the* generalized non-basic label $[B_i, B_\infty)$ *is defined by*

$$[B_i, B_\infty) = \{B_t \mid t \in I, i \leq t\}.$$

Note that B_∞ is a symbol, not a basic label.

Definition 4 *The set of* Generalized Orders of Magnitude $\mathbb{S}_g^*$ *is:*

$$\mathbb{S}_g^* = \{\emptyset\} \cup \{[B_i, B_j) \mid i, j \in I,\ i \leq j\} \cup \{[B_i, B_\infty) \mid i \in I\}.$$

In this definition of $\mathbb{S}_g^*$ the basic label B_i has been identified with the singleton $\{B_i\}$.

It is important to remark that the function $B : I \to \mathcal{P}(X)$ determines the elements of $\mathcal{S}$ and $\mathbb{S}_g^*$, and the cardinal of the set $I \subset \mathbb{R}$ determines the cardinal of $\mathcal{S}$ and therefore the cardinal of $\mathbb{S}_g^*$.

The *classical orders of magnitude qualitative spaces* (Travè-Massuyès and Dague 2003) verifies the conditions of the generalized model that has just been introduced. This model are build from a set of ordered basic qualitative labels determined by a partition of the real line.

Let X be the real interval $[a_1, a_n)$, and a partition of this set given by $\{a_2, \ldots, a_{n-1}\}$, with $a_1 < a_2 < \ldots < a_{n-1} < a_n$. The set of basic labels is

$$\mathcal{S} = \{B_1, \ldots, B_{n-1}\},$$

where, for $1 \leq i \leq n-1$, B_i is the real interval $[a_i, a_{i+1})$. The set of indexes is $I = \{1, 2, \ldots, n-1\}$.

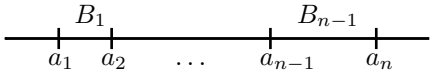


Figure 1. Classical aualitative labels $\mathbb{S}_n$

For $1 \leq i < j \leq n-1$ the non-basic label $[B_i, B_j)$ is:

$$[B_i, B_j) = \{B_i, B_{i+1}, \ldots, B_{j-1}\},$$

and it is interpreted as the real interval $[a_i, a_j)$.

For $1 \leq i \leq n-1$ the non-basic label $[B_i, B_\infty)$ is:

$$[B_i, B_\infty) = \{B_i, B_{i+1}, \ldots, B_{n-1}\},$$

and it is interpreted as the real interval $[a_i, a_n)$.

The complete universe of description for the Orders of Magnitude Space is the set

$$\mathbb{S}_n = \{[B_i, B_j) \mid B_i, B_j \in \mathcal{S}, i \leq j\} \cup \{[B_i, B_\infty) \mid B_i \in \mathcal{S}\},$$

which is called the absolute orders of magnitude qualitative space with granularity n, also denoted $OM(n)$. In this case, $\mathbb{S}_g^* = \{\emptyset\} \cup \mathbb{S}_n$.

There is a partial order relation $\leq_P$ in $\mathbb{S}_n$ "to be more precise than", given by:

$$L_1 \leq_P L_2 \Longleftrightarrow L_1 \subset L_2.$$

The least precise label is denoted by ? and it is the label $[B_1, B_\infty)$, which corresponds to the interval $[a_1, a_n)$.

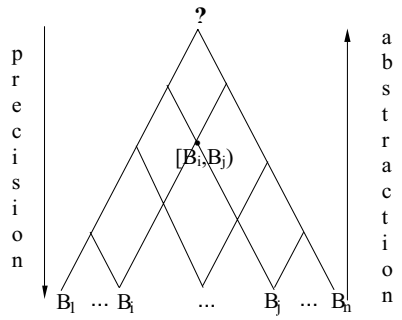


Figure 2. The space $\mathbb{S}_n$

This structure permits working with all different levels of precision from the label ? to the basic labels.

In some theoretical works, orders of magnitude qualitative spaces are constructed by partitioning the whole real line $(-\infty, +\infty)$ instead of a finite real interval $[a_1, a_n)$. However, in most real world applications involved variables do have a lower bound a_1 and an upper bound a_n, and then values less than a_1 or greater than a_n are considered as outliers and they are not treated like any other.

The classical sign algebra $\mathcal{S} = \{-, 0, +\}$ was the first absolute orders of magnitude space considered by the QR community. It corresponds to the case $\mathcal{S} = \{B_{-1} = (-\infty, 0), B_0 = \{0\}, B_1 = (0, +\infty)\}$. The sign algebra is obtained via a partition of the real line given by an unique landmark 0. The classical orders of magnitude qualitative spaces are built from partitions via a set of landmarks $\{a_2, \ldots, a_{n-1}\}$, and the classical interval algebra is built from the finest partition of the real line whose landmarks are all real numbers.

It is important to remark the significance of the presented mathematical formalism in the sense that it permits to lump together a family of $\mathbb{S}_g^*$ forming a continuum from the sign algebra $\mathcal{S} = \{-, 0, +\}$ to the interval algebra corresponding to $\mathcal{S} = \mathbb{R}$.

THE MEASURE SPACE $(\mathcal{P}(X), \Sigma(\mathbb{S}_g^*), \mu^*)$

To introduce the classical concept of entropy by means of qualitative orders of magnitude spaces, Measure Theory is required. This theory seeks to generalize the concept of "length", "area"and "volume", understanding that these quantities need not necessarily correspond to their physical counterparts, but may in fact represent others. The main use of the measure is to define the concept of integration for orders of magnitude spaces. First, it is necessary to define the algebraic structure on which to define a measure.

Definition 5 *A class of sets $\Im$ is called a* semi-ring *if the following properties are satisfied:*

1. $\emptyset \in \Im$.
2. *If $A, B \in \Im$, then $A \cap B \in \Im$.*
3. *If $A, B \in \Im$, $A \subset B$, then $\exists n \in \mathbb{N}, n \geq 1$ and $\exists\, D_1, D_2, \ldots, D_n$ such that $A = D_0 \subset D_1 \subset \ldots \subset D_n = B$, with $D_k - D_{k-1} \in \Im, \forall k \in \{1, \ldots, n\}$.*

Proposition 1 *$\mathbb{S}_g^*$ is a semi-ring.*

Proof:

1. $\emptyset \in \mathbb{S}_g^*$ by definition.
2. If $[B_i, B_j), [B_k, B_l) \in \mathbb{S}_g^*$, it is trivial to check that $[B_i, B_j) \cap [B_k, B_l) \in \mathbb{S}_g^*$, taking into account the relative position between the real intervals $[i, j)$ and $[k, l)$. Analogously, in the case of intersections $[B_i, B_j) \cap [B_k, B_\infty)$ or $[B_i, B_\infty) \cap [B_k, B_\infty)$.
3. If $[B_i, B_j), [B_k, B_l) \in \mathbb{S}_g^*$ such that $[B_i, B_j) \subset [B_k, B_l)$, then two cases are considered:
   (a) If $B_k = B_i$ or $B_l = B_j$, it suffices to take $D_0 = [B_i, B_j)$ and $D_1 = [B_k, B_l)$.
   (b) Otherwise, take $D_0 = [B_i, B_j), D_1 = [B_i, B_l)$ and $D_2 = [B_k, B_l)$.
   The cases $[B_i, B_j) \subset [B_k, B_\infty)$ and $[B_i, B_\infty) \subset [B_k, B_\infty)$ are proved in a similar way.

□

Definition 6 *A class $\mathcal{A}$ of subsets of a non-empty set X is called an* algebra *when it contains the finite unions and the complements of its elements. If finite unions are replaced by countable unions, it is called a σ-algebra.*

The smallest σ-algebra that contains $\mathbb{S}_g^* \subset \mathcal{P}(X)$ is called the σ-algebra generated by $\mathbb{S}_g^*$, denoted by $\Sigma(\mathbb{S}_g^*)$.

Definition 7 *Let X be a non-empty set and $\mathcal{C} \subset \mathcal{P}(X)$, with $\emptyset \in \mathcal{C}$. A* measure on $\mathcal{C}$ *is an application $\mu : \mathcal{C} \to [0, +\infty]$ satisfying the following properties:*

1. $\mu(\emptyset) = 0$.
2. *For any sequence $(E_n)_{n=1}^{\infty}$ of disjoint sets of $\mathcal{C}$ such that $\cup_{n=1}^{+\infty} E_n \in \mathcal{C}$, then*
$$\mu(\bigcup_{n=1}^{+\infty} E_n) = \sum_{n=1}^{+\infty} \mu(E_n).$$

Any measure μ on the whole $\mathcal{P}(X)$, when it is restricted to $\mathbb{S}_g^*$, gives a measure on $\mathbb{S}_g^*$.

Definition 8 *Let μ be a measure on $\mathbb{S}_g^*$. The* outer measure *on an arbitrary subset A of X is defined by:*
$$\mu^*(A) = \inf\{\sum_{k\in\mathbb{N}} \mu([B_{s_k}, B_{t_k})),\ A \subset \bigcup_{k\in\mathbb{N}} [B_{s_k}, B_{t_k})\}.$$

Carathéodory theorem (Halmos 1974) assures μ^* of definition 7 is a measure on $\Sigma(\mathbb{S}_g^*)$, and $(\mathcal{P}(X), \Sigma(\mathbb{S}_g^*), \mu^*)$ is called a measure space. It is proved that, since $\mathbb{S}_g^*$ is a semi-ring, $\mu^*_{|\mathbb{S}_g^*} = \mu$.

In this measure space an integration with respect μ^* can be defined. Because of the fact that $\mu^*_{|\mathbb{S}_g^*} = \mu$, in any integration on $\mathbb{S}_g^*$ the measure μ^* can be replaced by μ.

ENTROPY BY MEANS OF $\mathbb{S}_g^*$

Once the integration in $\mathbb{S}_g^*$ has been defined, entropy can then be considered. To introduce the concept of entropy by means of qualitative orders of magnitude, it is necessary to consider the qualitativization function between the set to be qualitatively described and the space of qualitative labels, $\mathbb{S}_g^*$.

To simplify the notation, let us express with a calligraphic letter the elements in $\mathbb{S}_g^*$; thus, for example, elements $[B_i, B_j)$ or $[B_i, B_\infty)$ shall be denoted as $\mathcal{E}$.
Let Λ be the set that represents a magnitude or a feature that is qualitatively described by means of the labels of $\mathbb{S}_g^*$. Since Λ can represent both a continuous magnitude such as position and temperature, etc., and a discrete feature such as salary and colour, etc., Λ could be considered as the range of a function

$$a : I \subset \mathbb{R} \to Y,$$

where Y is a convenient set. For instance, if a is a room temperature during a period of time $I = [t_0, t_1]$, Λ is the range of temperatures during this period of time. Another example can be considered when $I = \{1, \ldots, n\}$ and $\Lambda = \{a(1), \ldots, a(n)\}$ are n number of people whose eye colour we aim to describe. In general, $\Lambda = \{a(t) = a_t \,|\, t \in I\}$.

The process of qualitativization is given by a function

$$Q : \Lambda \to \mathbb{S}_g^*,$$

where $a_t \mapsto Q(a_t) = \mathcal{E}_t =$ minimum label (with respect to the inclusion $\subset$) which describes a_t, i.e. the most precise qualitative label describing a_t. All the elements of the set $Q^{-1}(\mathcal{E}_t)$ are "representatives" of the label $\mathcal{E}_t$ or "are qualitatively described" by $\mathcal{E}_t$. They can be considered qualitatively equal.

The function Q induces a partition in Λ by means of the equivalence relation:

$$a \sim_Q b \Longleftrightarrow Q(a) = Q(b).$$

This partition will be denoted by $\Lambda/\sim_Q$, and its equivalence classes are the sets $Q^{-1}(Q(a_j)) = Q^{-1}(\mathcal{E}_j)$, $\forall j \in J \subset I$. Each of these classes contains all the elements of Λ which are described by the same qualitative label.

Definition 9 *Let μ be a measure on $\mathbb{S}_g^*$ such that*

$$\int_{\bigcup_{i\in I}\{B_i\}} d\mu = 1.$$

The entropy H with respect the partition $\Lambda/\sim_Q$ is the integral:

$$H(\Lambda/\sim_Q) = -\int_{Q(\Lambda)} \log \mu \, d\mu, \qquad (1)$$

where $Q(\Lambda)$ is the set of labels mapped by Q (logarithms are to the base 2).
The expression (1) can be written as:

$$H(\Lambda/\sim_Q) = -\sum_{j\in J} \log(\mu(\mathcal{E}_j))\mu(\mathcal{E}_j). \qquad (2)$$

As in most definitions of entropy, it gives a measure of the amount of information. In Definition 9 entropy can be interpreted as the measure of the amount of information that provides the knowledge of Λ by means of Q.

Nevertheless, the inner features of the orders of magnitude structure considered introduce some differences between the entropy defined in (1) and the entropy defined by Rokhlin (Rokhlin 1967) and Shannon (Shannon 1948), as can be seen in the following example:

Example 1 Suppose that Q maps each element of Λ to the same label $\mathcal{E} \in \mathbb{S}_g^*$; then the induced partition $\Lambda/\sim_Q$ contains only one class equal to Λ and the entropy defined in equation (1) is $H(\Lambda/\sim_Q) = -\mu(\mathcal{E})\log\mu(\mathcal{E})$. In the classical interpretation of the entropy, the knowledge about Λ induced by this particular Q will lead to an entropy equal to zero, because in the given situation it is understood that this trivial partition of Λ provides no information at all. On the contrary, in the approach that has been presented in this paper, although Q map the whole set to the same label it could give a certain information about Λ: the intrinsic information provided by the measure of the label itself.

Two different measures that show this fact are considered in the following examples. On the one hand, the first differs from Shannon's classical interpretation of entropy as noted in Example 1: although Q map each element of Λ to the same label $\mathcal{E} \in \mathbb{S}_g^*$ entropy is not equal to zero . On the other, the entropy corresponding to Example 3 behaves like the classical interpretation of Shannon and Rokhlin, in the sense just discussed. Example 2 takes into account the lengths of the intervals corresponding to the labels, and Example 3 is related to the cardinality of the set of representatives of each label.

Example 2 Let us define a particular measure μ on $\{\emptyset\} \cup \mathbb{S}_n$ as follows:
For the basic labels $B_i = [a_i, a_{i+1})$, whith $i = 1, \ldots, n-1$, let

$$\mu(B_i) = \frac{a_{i+1} - a_i}{a_n - a_1}.$$

This measure is proportional to the knowledge of imprecision about the magnitude and it is normalized with respect to the "basic" known range given by the length $a_n - a_1$. For non-basic labels the measure is, for $i, j = 1, \ldots, n-1$, $i < j$:

$$\mu([B_i, B_j)) = \sum_{k=i}^{j-1} \mu(B_k) = \frac{a_j - a_i}{a_n - a_1},$$

and for $i = 1, \ldots, n-1$:

$$\mu([B_i, B_\infty)) = \sum_{k=i}^{n-1} \mu(B_k) = \frac{a_n - a_i}{a_n - a_1}.$$

Elements of Λ represented by quite precise labels will provide a bigger contribution to entropy H than those who are represented by less precise labels. Considering the particular case in which Q maps all the elements of Λ to the same label: $Q(\Lambda) = \{\mathcal{E}\}$, then $\Lambda/\sim_Q = \Lambda$ and $H(\Lambda/\sim_Q) = -\mu(\mathcal{E})\log(\mu(\mathcal{E})) \neq 0$.

Example 3 Another interpretation of the entropy defined in equation (1) is obtained by defining another measure μ over $\{\emptyset \cup \mathbb{S}_n$ as follows: For each $\mathcal{E}_t \in \{\emptyset\} \cup \mathbb{S}_n$,

$$\mu(\emptyset) = 0,\ \mu(\mathcal{E}_t) = \mathrm{card}(Q^{-1}(\mathcal{E}_t))/\mathrm{card}(\Lambda).$$

This case recovers the classical interpretation of Shannon and Rokhlin in the sense that if Q maps all the elements of Λ to the same label, then the partition does not give information of Λ because the entropy is $H(\Lambda/\sim_Q) = -1 \cdot \log 1 = 0$. Moreover, the entropy reaches its maximum when different elements of Λ are mapped to different labels $\mathcal{E}_t \in \mathbb{S}_n$, i.e., when Q is an injective map from Λ onto $\mathbb{S}_n$. This maximum is $H(\Lambda/\sim_Q) = \log(\mathrm{card}\,\Lambda)$.

CONCLUSION AND FUTURE WORK

This paper introduces the concept of entropy by means of absolute orders of magnitude qualitative spaces. This entropy measures the amount of information of a system when using orders of magnitude descriptions to represent it.

In order to define the concept of entropy within Qualitative Reasoning framework, this paper adapts the basic principles of Measure Theory to give the space of absolute orders of magnitude the necessary structure. With the presented structure, we obtain a family of qualitative spaces forming a continuum from the sign algebra to the classical interval algebra.

From a theoretical point of view, future research could focus on two lines. On the one hand, it could focus on the comparison of the given entropy with the macroscopic concept of Caratheodory entropy. On the other hand, the adaptation of Measure Theory provides the theoretical framework in which developing a rigorous analytical study of functions between orders of magnitude spaces. The continuity and differentiability of these functions will allow the dynamical study of qualitatively described processes.

Within the framework of applications, this work and its related methodology will be orientated towards the modelization and the resolution of financial and marketing problems. Regarding financial problems, the concept of entropy will facilitate the study of the evolution and variation of the financial ratings. On the other hand, entropy as a measurement of coherence and reliability is useful in group decision-making problems arising from retail marketing applications.

Moreover, the introduced entropy will allow defining a conditional entropy in this framework, which in turn will allow considering the Rokhlin distance to be used in decision-making problems of ranking and selection of alternatives.

References

Cover, T. M., and Thomas, J. A. 1991. *Elements of Information Theory*. Wiley Series in Telecomunications.

Folland, G. 1999. *Real Analysis: Modern Techniques and Their Applications*. Pure and Applied Mathematics: A Wiley-Interscience Series of Texts, Monographs, and Tracks. John Wiley & Sons, Inc.

Forbus, K. 1984. Qualitative process theory. *Artificial Intelligence* 24:85–158.

Forbus, K. 1996. *Qualitative Reasoning*. CRC Hand-book of Computer Science and Engineering. CRC Press.

Halmos, P. R. 1974. *Measure Theory*. Springer-Verlag.

Kuipers, B. 2004. Making sense of common sense knowledge. *Ubiquity* 4(45).

Rokhlin, V. 1967. Lectures on the entropy of eeasure ereserving eransformations. *Russian Math. Surveys* 22:1 – 52.

Shannon, C. E. 1948. A mathematical theory of communication. *The Bell System Technical Journal* 27:379 – 423.

Travè-Massuyès, L., and Dague, P., eds. 2003. *Modèles et raisonaments qualitatifs*. Hermes Science (París).

Qualitative Modeling for Diagnosis of Machines Transporting Rigid Objects

Peter Struss, Axel Kather, Dominik Schneider, Tobias Voigt

Technische Universität München
Arcisstr. 21, D-80333 Munich, Germany
struss@in.tum.de

Abstract

We present models of various elements of a plant that involves the transportation of lumped material. An application context is provided by a project on diagnosing disturbances in food packaging plants and, more specifically, bottling plants. While there exist models of flow of homogeneous matters, such as liquid material in a hydraulic system, based on simultaneous equations of Kirchhoff/Ohm type, in our project we need to cope with non-negligible transportation time of objects and capture phenomena like the tailback of units (if transportation is blocked) or the propagation of gaps in the flow of units. Because the application context requires compositionality of the model, i.e. local, context-free models of the individual transportation elements, we are also facing the problem that whether or not a single element produces an output flow (or accepts an input flow) cannot be determined solely by the model of this element, but only through modeling the interaction with the subsequent element, which may block the output (or the previous one not providing the input). This issue is addressed by modeling the potential of an existing flow distinctly from the actual occurrence of a flow, an idea which also can enhance models of continuous flow.

1. Introduction

Modeling the flow of some matter in a system is quite widespread in model-based systems, e.g. in model-based diagnosis of hydraulic or pneumatic systems. At least under certain simplifying assumptions, mathematical first principles models exist, and it appears to be straightforward to abstract them into adequate input to a model-based problem solver.

Typically, such models assume that the flowing matter is continuous and homogeneous and does not have to be modeled as an object or in its detailed structure. And they usually incorporate the analogies to Kirchhoff's and Ohm's Laws, which leads to simultaneous equations that imply instantaneous propagation of pressure and disregard time needed by the matter to be transported through the system.

There are classes of application domains that involve a flow of objects through a plant and, hence, suggest the use of some flow model, but require dropping some of the simplifying assumptions mentioned. One instance of this class is given by food packaging plants, which are subject to a diagnosis project we are carrying out, and, more specifically, by bottling plants, which we will use as an example in this paper. Such plants involve streams of objects of different types, bottles, crates, and pallets being the most prominent ones. On the one hand, modeling the transportation of individual objects is prohibitive or useless. On other hand, the abovementioned flow models of a homogeneous matter fail to capture essential features, such as gaps in the flow or the creation of a tailback by some blockage and its propagation through the plant in finite time. Furthermore, an inflow and outflow of a single transportation element of a line cannot definitely be predicted by a local model of this element, because they depend also on the supply of the previous element and the intake capacity of the following one, resp. As a consequence, we had to develop a model that

- includes transportation times,
- covers interrupted flows,
- handles the exchange of flows between neighboring elements appropriately.

The paper focuses on presenting a base model addressing the requirements (section 3), its validation through simulation (section 4) and a qualitative diagnosis model obtained from it (section 5). The diagnosis engine will be presented in a separate paper.

The following section presents an application context of this work, namely bottling plants

2. An Application Domain: Bottling Plants

Food packaging at industrial scale is carried out in high output packaging lines consisting of specific machines and conveyors. There are different machines for specific packaging tasks, such as primary packaging of food or beverages (e. g. with foil packs, pouches, or containers), secondary packaging (boxes, multipacks, crates, etc.), and tertiary packaging (e. g. pallets or displays). Additionally, machines for de-palletizing and unpacking of returnable bottles, cleaning, inspection and sorting out improper objects may be involved. Plant constellations are configured using one machine of a specific type or several ones in parallel. Machines of different types are connected by conveyors. Because of the high speeds and output rates (up to 100.000 packages per hour), machines and

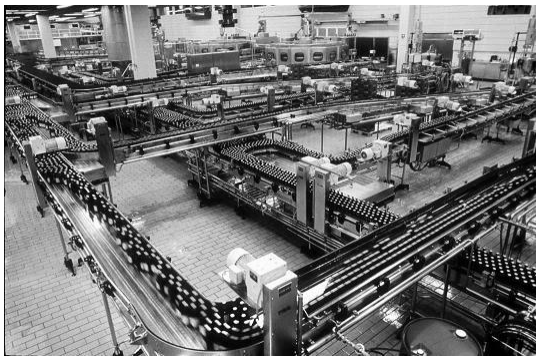


Figure 1. *Conveyors of a bottling plant for returnables*

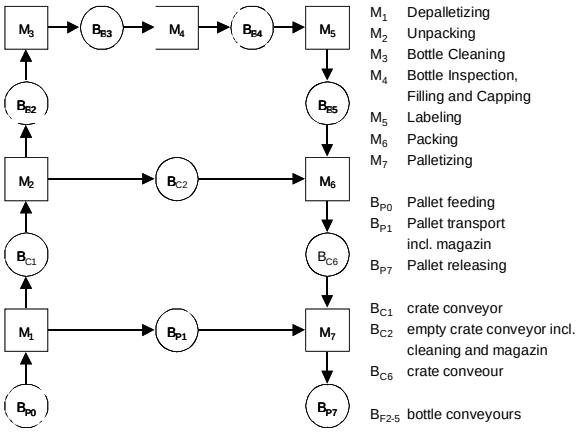


Figure 2. *Generic structure of a bottling plant for returnable bottles*

conveyors are failure-sensitive with an availability degree of 92-98 percent.

As a specific example for packaging plants, our project considers bottling plants for beverages (e.g. the one shown in Fig 1).

In order to fill beverages into returnable bottles, the material flows of pallets, crates, and bottles (plus labels, glue, etc.) need to be coordinated. This leads to complex line configurations comprised of machines that remove crates from pallets and bottles from crates, process, inspect, or sort objects, and package different types of objects (Fig. 2 shows an abstract, but typical example).

To prevent oxygen intake ore microbiological contaminations of the beverage, the filling process should not be interrupted. Therefore transportation by consecutive machines needs to be decoupled. Otherwise, each individual failure would inevitably cause downtime of the entire plant: In particular, this would stop the filling process and decrease the efficiency of the entire production. To prevent this, the conveyors of bottling plants are designed as transporting buffers like the abstract bottle conveyor shown in Fig. 3.

Transporting buffers perform two tasks. One is to carry the objects from one machine to the next one. The other is to store objects in order to be able to compensate for a downtime of the upstream machine and to prevent the immediate propagation of a tailback in case of a downtime of the downstream machine. In addition, the machines located upstream and downstream w.r.t. the filling machine work with higher output rates than the filler. This enables full upstream buffers and receptive downstream buffers to compensate for short downtimes of single machines.

These design principles help achieving a continuous operation of the filling machine. However, in practice, they

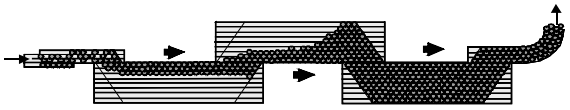


Figure 3. *A three step transporting buffer for bottles*

cannot guarantee avoidance of unwanted idle time of the filler, and (unplanned) downtime of the plant can lie in the range of 10-30 percent.

Machine failures of significant duration, gaps caused by a large number of objects being sorted out, stoppages caused by toppled or jammed objects, or just mistakes of the operators result in downtime of the filling machine and decrease the availability of the entire plant. Because of the interlaced flows of the various object types, time offsets, and the large scale of the plants, the reasons for such plant downtimes can be difficult to identify by the plant operators, particularly since their number has been progressively reduced over the past years. In consequence, bottle filling and packaging industries is highly interested in an automated diagnosis tool for their plants.

There are a number of requirements and challenges to automated diagnosis raised by this application task. A fundamental economical condition is the fact that many of the potential end users, e.g. breweries, are small or medium enterprises, which could not afford spending many resources on the establishment or adaptation of a tailored diagnosis system for their plant. Another practical requirement is to cheaply accommodate frequent changes in the structure of the line, due to rearrangement or addition of machines. Both issues suggest a **model-based solution** to diagnosis (see [Struss 08]), which allows performing adaptation by simply (re-)specifying the plant structure.

Additional arguments for such a solution stem from the facts that usually a plant is a combination of machines from various manufacturers with different instrumentation and available data and that there may be temporarily missing data due to technical problems. This requires a flexible solution that derives the best diagnosis from **whatever data is available** (in contrast, for instance, to decision trees based on a fixed set of observables).

Heterogeneity and changes of the set of machines also establishes a requirement on the model: firstly, it has to be machine-centered and **compositional**; secondly, it has to

be stated at a level of abstraction that covers **types of machines**, independently of specificities and the manufacturer.
Besides these fundamental characteristics, the model has to be capable of properly predicting the propagation of **gaps** in the stream of objects (potentially causing a lack in supply to subsequent machines) and **tailbacks** caused by blockages, as well the **propagation of special features** and deficiencies of the transported objects, which may be caused by improper performance of one machine (e.g. improper cleaning) and may affect the (mis-)behavior of another element downstream (e.g. an inspection machine).
The available data is inherently incomplete and imprecise. Even balance equations do not necessarily hold, because bottles may have been removed by an operator (for inspection or because they blocked the flow) or simply have fallen off the belt.

3. Models of Transportation Elements

3.1 Previous Work

The only similar work we are aware of (except for discrete-event-simulation models used for validation of the control, which do not lend themselves easily to model-based diagnosis) is in the domain of transport of paper in a copier. [Gupta-Struss 95] presents a process-oriented model, and [Fromherz et al. 03] develop a component-oriented model for control generation. Both models are compositional, but focus on the motion of individual sheets, rather than the more abstract perspective of flow of objects.

3.2 Modeling Assumptions

We first list the most important assumptions underlying the transportation models presented here, which are fulfilled in our project domain (under normal conditions), but should also apply to a much broader class of problems.

- The transported objects are rigid bodies with fixed spatial extensions and are not significantly deformed through transportation.
- They are transported with a fixed orientation (like crates), or the orientation does not affect transportation times significantly (e.g. due to a symmetric cross-section, as for bottles).
- There is no interaction among the objects or between objects and the components that has a significant impact on the transportation process (such as bouncing).
- Objects can move only in the direction of the motion of the transportation means (or not at all), although not necessarily with the same speed.

3.3 A Model of a Transportation Element with Buffer

In order to present the essentials of the modeling approach, we consider some sort of archetype of model, which can be specialized or extended to accommodate other kinds of machines. This is a machine that

- has one input and one output with v_{in}, v_{out} being the respective speeds of the means for transportation (e.g. belts),
- possibly transforms or modifies one kind of object (as, for instance, cleaning of bottles), but does not amalgamate several objects to form a new one,
- has a buffer with a (constant) capacity C.

The process of buffering the objects can be fairly random, as illustrated by the bottle conveyor in Figure 3, where bottles may gather in bulks. However, it is assumed, that (under normal behavior) no object is prevented from approaching the output unless it is blocked by other objects ahead, waiting for output. For instance, within the bottle conveyor, its shape and several parallel belts with different speeds ensure that bottles are not left in some corner, but pushed towards the "ideal" fastest belt, if there is space.
The intuition behind the model can be best described in terms of three fundamental concepts and five "behavior rules", each of which is first introduced informally and then turned into equations. As stated before, one of the problems to be solved stems from the fact that a local machine model in isolation cannot determine whether an actual flow occurs at its input and output. But it can and has to express the limits on the machine's **potential** to take in or output objects. This is reflected by

Concept 1 The **potential input and output flow**, $in.q_{pot}$ and $out.q_{pot}$, represent the maximal flow the machine can accept or generate, dependent on its internal state.

The **actual** flows are represented by two different variables, $in.q_{act}$ and $out.q_{act}$. The first restriction is determined by

Rule 1 The **potential input flow** is given by the input speed of the transportation element, unless the buffer is full. In this case, it cannot be higher than the actual output flow.

In the mathematical model (see Fig. 4), this rule is formalized by equation 1, where d denotes the diameter of the object cross-section and B is the filling degree of the buffer (in terms of number of objects). It involves the assumption that an actual outflow generates the potential for intake instantaneously, which is not true in practice and, hence, another reason for expressing tolerance intervals with values and time. Note that we take all speeds and flows as positive, as their sign is determined by their association with the intrinsic direction of the transportation element. Computing B is straightforward:

Rule 2 The **change in the total** number of buffered objects **is determined by the actual input and output flows.**

The respective equation 2 indicates that B will be computed by integrating the difference of the actual flows. Setting up the model fragments for the potential output flow is based on the second key idea:

Transportation Element with Buffer

State variables

$B(t)$ # objects in buffer
$B_{out}(t)$ # objects buffered for immediate output
$v_{in}(t)$ velocity of input transportation means
$v_{out}(t)$ velocity of output transportation means
$t_d(t)$ minimal transportation time

Parameters

d_0 diameter of transported object (in transportation plain)
C Capacity (as number of objects)

Interface variables

$in.q_{pot}(t)$ potential inflow [objects/s]
$out.q_{pot}(t)$ potential outflow [objects/s]
$in.q_{act}(t)$ actual inflow [objects/s]
$out.q_{act}(t)$ actual outflow [objects/s]

Equations

(1) $in.q_{pot}(t) = v_{in}(t) / d_0$ if $B(t)<C$
$in.q_{pot}(t) = \min (v_{in}(t) /d_0, out.q_{act}(t))$ if $B(t)=C$
(2) $dB/dt = in.q_{act}(t) - out.q_{act}(t)$
(3) $out.q_{pot}(t)= v_{out}(t) /d_0$ if $B_{out}(t)\geq 1$
$out.q_{pot}(t)= \min (in.q_{act}(t - t_d) , v_{out}(t) /d_0)$ else
(4) $dB_{out}(t) /dt = in.q_{act}(t - t_d) - out.q_{act}(t)$

Connector between Transportation Elements

Interface variables

$TE_{n+1}.in.q_{pot}(t)$ potential inflow of upstream element TE_{n+1}
$TE_n.out.q_{pot}(t)$ potential outflow of downstream element TE_n
$TE_{n+1}.in.q_{act}(t)$ actual inflow of upstream element TE_{n+1}
$TE_n.out.q_{act}(t)$ actual outflow of downstream element TE_n

Equations

(5) $TE_n.out.q_{act}(t) = \min (TE_{n+1}.in.q_{pot}(t) , TE_n.out.q_{pot}(t))$
$TE_n.out.q_{act}(t) = TE_{n+1}.in.q_{act}(t)$

Figure 4. *Equations of buffer and connector*

Concept 2 B_{out} denotes the number of **buffered output objects** at time t, i.e. the number of objects that can possibly be subject to output at this time.

Before we clarify this crucial concept, we use its intuitive understanding and the third concept for formulating the rule for the potential output flow.

Concept 3 The **minimal transportation time**, t_d, is the time an object needs to get directly from the input to the output, i.e. if it is not delayed by other objects that are piling up.

In case of the bottle conveyor, this means that the bottle stays on the fastest (innermost) belt.

Rule 3 The **potential output flow** is determined solely by the output speed, if there is more than one buffered output object. Otherwise, it cannot be higher than the actual input flow at the time reduced by the minimal transportation time.

One should be aware that in the second case, each single object may (potentially) leave the output with the speed v_{out}. However, if the input flow at the time when it entered was lower, there will be a gap occurring after the output of the object, which makes the (average) flow lower than v_{out}. As a special case, the potential output flow becomes zero, if the actual input flow was zero at the respective time. Again, the respective equation 3 in Figure 4 formalizes this. Computing B_{out} also involves the minimal transportation time t_d. If an object entered the transportation element later than time $t - t_d$, it cannot possibly reach the output at time t and, hence, cannot become part of the buffered output objects. If it entered earlier, it may or may not have already left the output before t, depended on how the actual output flow reduced B_{out}. This consideration is captured by

Rule 4 The change in the **number of buffered output objects** at time t is determined by the actual input flow at time $t - t_d$ diminished by the actual outflow at time t.

Hence, also B_{out} is obtained by integration according to equation 4, which completes the model of the transportation element with buffer. Note, that B_{out} is **not** necessarily the number of objects that form a contiguous pile in front of the output. It could be less, because the last objects that joined the pile entered later than $t - t_d$.

3.4 Interaction of Transportation Elements

What remains to be done is determining the actual flows from the potential flows of connected machines. This interaction is captured by a model of a generic connector used for connecting all types of transportation elements. The respective rule and equation 5 (Fig. 4) are straightforward:

Rule 5 The **actual output flow of a machine** is limited by both its own potential output flow and the potential input flow of the following machine (and equal to the actual input flow of this machine).

3.5 Other Features and Transportation Elements

The buffer model leaves options for different use and specialization. Due to lack of space, we can only sketch some important cases, many of which are fairly straightforward. For instance, v_{in} and v_{out} could be different as for the entire bottle conveyor shown in Figure 3. In this case, the minimal transportation time t_d needs to be calculated or estimated based on varying speeds along the "ideal path". Alternatively, the same conveyor can be considered as an aggregation of several buffers in series each with one unique speed on its fastest belt, which eases the computation of t_d. Note that the speeds are subject to control and may vary dynamically. Therefore, in case of a unique speed, t_d is determined by the equation

$$l = \int_{t-t_d}^{t} v(\tau) d\tau ,$$

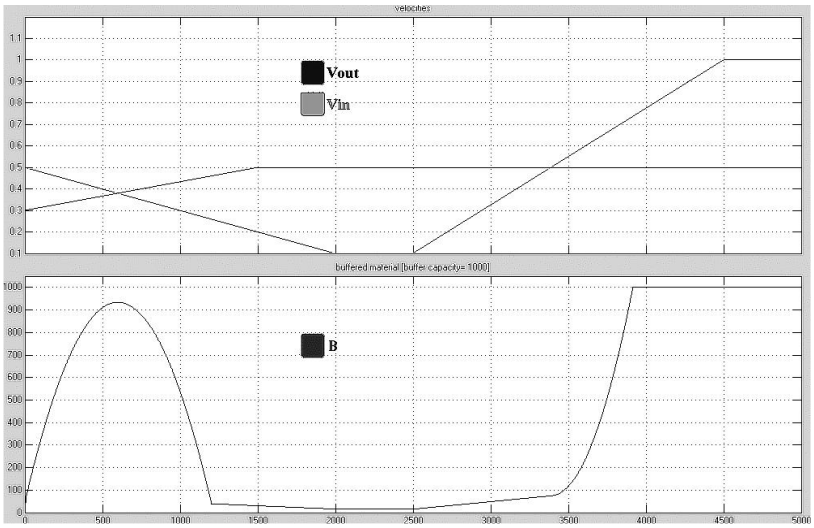


Figure 5. *Plots showing the changes of the buffer (lower) in response to variation of input and output speeds (upper)*

where l is the length of the "ideal path" and v(t) its time-varying speed.

Gates may sit at the input or output of transportation elements and are controlled in a binary manner in order to block the flow entirely if necessary. This is captured by multiplying the respective speed with a factor of (1 – $state_{gate}$), if $state_{gate}$ is 1 for a closed gate and 0 otherwise.

While the bottle conveyor has no fixed relation between the speed of the belts and the motion of the bottles, which may slide, other machines, such as the filler, transport objects by locking them to certain sockets. This is obtained as a specialization of the buffer model with a unique speed and the capacity given by the number of sockets that can be occupied by objects while processing them.

Some elements, such as the bottle cleaning unit, may have n inputs of the same type of objects). To accommodate this feature in the model, we simply have to replace the actual input flow by the sum of several individual input flows. Elements having several outputs (for objects of the same type) usually require some modeling of the mechanism that distributes the objects among the various outputs, e.g. evenly (if possible) or according to some criteria. An example for the latter case is given by inspection machines ejecting objects that fail to pass some test.

Another class of machines produces an output by **combining** objects of different kinds, as for instance the packaging of 20 bottles in a crate. The ratio of the number of different objects participating in this combination is usually not arbitrary, but exactly specified. This ratio links the various potential and actual inflows and the outflow, which is then limited by the "slowest" input flow (relative to the ratio of the respective object type).

The counterpart to this very generic combination element is the **separation** element, with unpackers being a subclass, in which the slowest actual outflow of a decomposition result limits the potential inflow of the composite object.

This set of fairly generic model types turns out to cover the variety of machines in a bottling plant and, more generally, also in the food packaging plants that we encountered.

4. Validation of the Base Model

In order to validate the component models described above we implemented them as numerical simulation models in MATLAB/SIMULINK® [MathWorks 08] and compared the simulated behavior (using the solver \ode4" (Runge-Kutta) with a fixed-step size of one second) with the one of real plants.

Every component was modeled using the equations introduced above and tested in isolation to check whether it was adequate of and stated in a context-independent manner, which is a prerequisite for compositionality. In a second step, a model of a complete plant was configured using the validated components.

In testing the individual components, values of single parameters and variables were varied, and the response of the simulated behavior was monitored. For example, the predicted changes in the buffered material B of a component for different values of the input speed v_{in} and the output speed v_{out} are shown in Figure 5. It depicts that the buffer fills as long as the input speed is higher than the output speed (assuming a sufficient supply), whereas with the input speed reduced to its minimum 0.1 and the output speed being still high, the amount of buffered objects decreases.

Because of the minimal transportation time, t_d, of the component, the buffer is not completely emptied, as long as there is input available. Furthermore, only the objects represented by the variable B_{out} determine the existence of an output flow. Another real characteristic behavior can be reproduced when increasing the input speed while maintaining the output speed constant. Although v_{in} is still

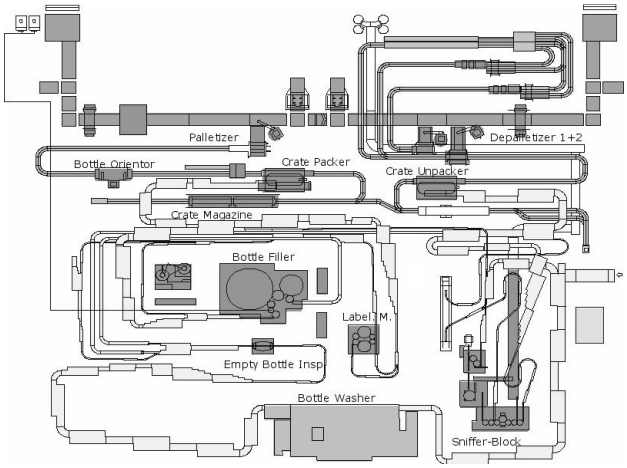


Figure 6. *The structure of one of the test plants*

higher than v_{out}, the buffer filling degree remains constant after a certain time, because it is limited by the maximum capacity of the component.

Similar results were achieved by testing the other component type models, providing evidence that the models capture the features relevant to the diagnostic task and do not violate context-independence.

The second challenge was validation by comparing the simulated behavior of a plant model with the behavior of a real plant. Several test cases were constructed, based on real-world downtimes scenarios of the bottling plant whose topology is shown in Fig. 6.

The simulated plant consists of a primary flow of bottles and a secondary object flow of crates. In one test case, the downtime propagation of a failure of the crate washer was simulated and analyzed. This failure interrupts both object flows. After some delay, missing input occurs at the crate packer. Also the unpacker stops at some point, due to its output being blocked. The details of the propagation of failure depend on the capacities and filling degrees of the various buffers connecting the machines. For instance, if the crate magazine is empty and all other buffers are filled with a sufficient degree, the lack of crates will rapidly reach the crate packer. This causes a blockage of the labeling machine and the bottle filler (because the packer is not able to process the bottles) before the lack of bottles in the primary flow (caused by the inoperable unpacker) reaches the filling machine. In contrast, if the crate magazine is completely full, the crate packer keeps working for some time, and the filling machine will be stopped due to a lack of bottles.

Even for this complex scenario, the simulation model reproduces the behavior of the real world plant. Similarly, the characteristics of fault propagation occurring in real plants were predicted for other relevant scenarios.

5. Abstraction to Qualitative Diagnosis Models

Using the model presented above directly for diagnosis is not appropriate. Firstly, as for all numerical models, its accuracy is only a pretended one in many respects, e.g. in assuming conservation laws to hold and in ignoring the imprecision in the available data, e.g. when flows are determined via counters or the speed of belts. Secondly, the diagnostic task requires the analysis of **qualitative**, rather than arbitrarily small numerical deviations from the nominal behavior and, hence, needs to be addressed by an appropriate level of abstraction in the model.

The level of model abstraction depends on the intended goal of the diagnosis: we first focused on "hard" failures (stop of the filling machine, that is) caused by hard faults (blockage of another machine), which can be based on distinguishing zero from non-zero flow only. For capturing "soft" faults (deviating behaviors) that lead, perhaps in combination, to a hard failure or a non-optimal behavior, a different model is required.

5.1 Sign-based Absolute Model

The total interruption of the flow requires distinctions between zero and non-zero flows only. Sign abstraction of the numerical model yields the qualitative constraints on the variables shown in Fig. 7 (we omit equations (2) and (4), which are difficult or impossible to exploit because neither B(t) nor $B_{out}(t)$ can be observed properly) together with the respective finite relations. (Remember that flows and speeds cannot be negative).

The abstraction of combination elements (such as the crate packer) outlined in section 3.5 will include the application of the three model fragments of Fig. 7 to all individual inflows as well as a constraint simply stating the qualitative equality of all inflows (the ratio of the flows drops out, because it is a positive number):

$[in_1.q_{pot}(t)] = [in_2.q_{pot}(t)] = \ldots = [in_k.q_{pot}(t)]$.

This captures, for instance, the fact that one lacking input will stop all other inputs, as well. The dual applies to separation elements.

This model has been validated using the diagnosis tool RAZ'R [Raz'r 08] on several scenarios, including the one described at the end of section 4, which involves a fault in the washer. (Because the current version of RAZ'R does not support the required temporal indexing of the predictions, the temporal information was stripped off and cyclic prediction was prevented in order to avoid spurious inconsistencies due to different values occurring at different times). The model is consistent with a lack of crates for the packer, which propagates backwards to a potential stop of the unpacker, which in turn may be caused by the inoperability of the washer.

We briefly demonstrate that the inferential power of the model suffices for handling the considered class of faults and failures despite its simplicity: assume that a transportation element TE_n with a single speed, $v_{in}(t)$ =

Transportation Element with Buffer

(1) $[in.q_{pot}(t)] = [v_{in}(t)]$ if C-B(t) > 0

$[in.q_{pot}(t)] = \min([v_{in}(t)], [out.q_{act}(t)])$ if C-B(t) = 0

$[in.q_{pot}(t)]$	$[v_{in}(t)]$	$[out.q_{act}(t)]$	[C-B(t)]
0	0	*	+
+	+	*	+
+	+	+	0
0	0	+	0
0	+	0	0

(3) $[out.q_{pot}(t)] = [v_{out}(t)]$ if $B_{out}(t)-1 \geq 0$

$[out.q_{pot}(t)] = \min([in.q_{act}(t - t_d)], [v_{out}(t)])$ if $B_{out}(t)-1<0$

$[out.q_{pot}(t)]$	$[v_{out}(t)]$	$[in.q_{act}(t - t_d)]$	$[B_{out}(t)-1]$
0	0	*	0
0	0	*	+
+	+	*	0
+	+	*	+
0	0	+	-
0	+	0	-
+	+	+	-

Connector between Transportation Elements

(5) $[TE_n.out.q_{act}(t)] = \min([TE_{n+1}.in.q_{pot}(t)], [TE_n.out.q_{pot}(t)])$

$[TE_n.out.q_{act}(t)] = [TE_{n+1}.in.q_{act}(t)]$

$[TE_n.out.q_{act}(t)]$	$[TE_{n+1}.in.q_{pot}(t)]$	$[TE_n.out.q_{pot}(t)]$
0	0	+
0	+	0
+	+	+

Figure 7. *Sign-based qualitative models of buffer and connector. [x] denotes the sign of x. "*" in a row represents "no restriction" and, hence, the entire row multiple tuples*

$v_{out}(t)$, produces an output, i.e. $[TE_n.out.q_{act}(t)] = +$, but has no inflow, $[TE_n.in.q_{act}(t)] = 0$. Then the constraints yield:

$[TE_n.out.q_{act}(t)] = + \ (5) \Rightarrow \ [TE_n.out.q_{pot}(t)] = +$

$(3) \Rightarrow \ [TE_n.v_{out}(t)] = [TE_n.v_{in}(t)] = +$

$[TE_n.out.q_{act}(t)] = + \ \wedge \ [TE_n.v_{in}(t)] = +$

$(1) \Rightarrow \ [TE_n.in.q_{pot}(t)] = +$

$[TE_n.in.q_{pot}(t)] = + \ \wedge \ [TE_n.in.q_{act}(t)] = 0$

$(5) \Rightarrow \ [TE_{n-1}.out.q_{pot}(t)] = 0$

If TE_{n-1} is operational, which implies $[TE_{n-1}.v_{out}(t)] = +$, then

$[TE_{n-1}.out.q_{pot}(t)] = 0 \ \wedge [TE_{n-1}.v_{out}(t)] = +$

$(3) \Rightarrow \ [TE_{n-1}.in.q_{act}(t - t_d)] = 0$.

This means, even without information about the buffers, the lack is propagated backwards across the models of

Transportation Element with Buffer

(1) $\Delta in.q_{pot}(t) = \Delta v_{in}(t) \ \vee \ \Delta in.q_{pot}(t) = \Delta out.q_{act}(t)$

$\Delta in.q_{pot}(t)$	$\Delta v_{in}(t)$	$\Delta out.q_{act}(t)$
0	0	*
-	-	*
+	+	*
0	*	0
-	*	-
+	*	+

(3) $\Delta out.q_{pot}(t) = \Delta v_{out}(t) \ \vee \ \Delta out.q_{pot}(t) = \Delta in.q_{act}(t - t_d)$

$\Delta out.q_{pot}(t)$	$\Delta v_{out}(t)$	$\Delta in.q_{act}(t - t_d)$
0	0	*
-	-	*
+	+	*
0	*	0
-	*	-
+	*	+

Connector between Transportation Elements

(5) $\Delta TE_n.out.q_{act}(t) = \Delta TE_{n+1}.in.q_{pot}(t)$

$\vee \ \Delta TE_n.out.q_{act}(t) = \Delta TE_n.out.q_{pot}(t)$

$\Delta TE_n.out.q_{act}(t)$	$\Delta TE_{n+1}.in.q_{pot}(t)$	$\Delta TE_n.out.q_{pot}(t)$
0	0	*
-	-	*
+	+	*
0	*	0
-	*	-
+	*	+

Figure 8. *Qualitative deviation models of buffer and connector.* $\Delta x = [x_{act} - x_{ref}]$ *is the qualitative deviation of a variable from a reference value (e.g. nominal or "healthy" value). "*" in a row represents "no restriction" and, hence, the entire row multiple tuples.*

correct elements (but will be consistent with a "block" mode, for instance) as expected.

5.3 Qualitative Deviation Model

The base model can also be used as the starting point for an abstraction that allows analyzing more subtle problems: the filling machines may not always be forced to stop operation, but, perhaps, run at reduced speed due to insufficient supply. For this purpose, the base model can be transformed into one that captures the propagation of deviations from some reference along the lines of [Struss 04]). A deviation of a variable x is defined as

$$\Delta x = [x_{act} - x_{ref}],$$

i.e. the difference between the actual and some reference value, which may remain unspecified. Usually, the latter represents some optimal or nominal value. This definition plus the sign-based abstraction for deviation variables and

dropping B(t) and $B_{out}(t)$ transforms the base model into the deviation model of Fig. 8. Both the domain abstraction to signs and the projection that eliminates the buffer variables establish a true abstraction of the original model.
Besides the analysis of reasons for suboptimal performance, such a model may be useful or even necessary for the diagnosis of filler stoppages, as well. The reason is that the filler may be stopped not because its inflow is zero for a long time interval, but because the available inflow is less than the flow requested by its speed, i.e. $v_{in}(t)/d_0$, and, hence, there is a gap in the supply and the filler is not supplied with a bottle for each socket, as required.
This model has not yet been validated in the diagnostic setting. However, we provide again some evidence for its inferential power. The "soft version" of the previous example states that the output and the speed of TE_n do not deviate, but its inflow is too low. We obtain

$$\Delta\, TE_n.out.q_{act}(t) = 0 \;\wedge\; \Delta\, TE_n.v_{in}(t) = 0$$

$$(1) \Rightarrow \quad \Delta\, TE_n.in.q_{pot}(t) = 0$$

$$\Delta\, TE_n.in.q_{pot}(t) = 0 \;\wedge\; \Delta\, TE_{n-1}.out.q_{act}(t) = -$$

$$(5) \Rightarrow \quad \Delta\, TE_{n-1}.out.q_{pot}(t) = -$$

$$\Delta\, TE_{n-1}.out.q_{pot}(t) = - \;\wedge\; \Delta\, TE_{n-1}.v_{out}(t) = 0$$

$$(3) \Rightarrow \quad \Delta\, TE_{n-1}.in.q_{act}(t - t_d) = - ,$$

i.e. again, the deviation is propagated upstream.

6. Summary and Outlook

The validation has provided evidence that the models really capture the essential features of plant behavior we are interested in from a diagnostic perspective. However, we do not only have to cope with inaccurate values of quantities, such as flows, speeds etc. due to the actual process and the available measurements. Also the temporal inferences are not crisp. For instance, from zero output flow of a normally behaving machine during some time interval i_1, an earlier time interval i_0 can be inferred, in which zero input flow must have occurred. This means, in contrast to other temporal propagation schemes, the prediction cannot state that the flow was zero during the **entire** interval i_0, but only that there exists a **subinterval** $i'_0 \subseteq i_0$ with zero flow, which has to be taken into account in the consistency check. Furthermore, propagation will lead to progressively larger time intervals, which prompts for an approach that uses observations interleaved with prediction to narrow down the intervals.
There are also different types of diagnostic tasks, such as our current focus, off-line post-mortem diagnosis (through analysis of stored data), on-line post-mortem diagnosis, and predictive diagnosis.
Finally, the project aims at a contribution to improving the general conditions through standardization of the data acquisition. Partners of the consortium are the originators of an existing standard that has now been widely accepted for bottling plants. This has now been extended on the one hand regarding data relevant to diagnosis and on the other hand generalizing it for food packaging plants. This will significantly improve the conditions for effective and easily adaptable diagnostic solutions.

References

[Fromherz et al. 03] Fromherz, M., Bobrow, D., and de Kleer, J.: Model-based Computing for Design and Control of Reconfigurable Systems. In: Bredeweg, B. and Struss P. (eds), Qualitative Reasoning. Special issue of AI Magazine, Winter 2003, 24(4), AAAI Press, Menlo Park, USA

[Gupta-Struss 95] Gupta, V., Struss, P.: Modeling a Copier Paper Path: A Case Study in Modeling Transportation Processes. In: QR-95, Working Papers of the Ninth International Workshop on Qualitative Reasoning, Amsterdam, 1995.

[MathWorks 08] http://www.mathworks.com/products

[Raz'r 08] http://www.occm.de/

[Struss 04] P. Struss: Models of Behavior Deviations in Model-based Systems. In: Proceeding of the 16th European Conference on Artificial Intelligence (ECAI04), 2004

[Struss 08] P. Struss: Model-based Problem Solving. In: van Harmelen, F., Lifschitz, V., and Porter, B. (eds.). Handbook of Knowledge Representation, Elsevier, 2008

Integrating Open-Domain Sketch Understanding with Qualitative Two-Dimensional Rigid-Body Mechanics

Jon W. Wetzel and Kenneth D. Forbus

Qualitative Reasoning Group, Northwestern University
2145 Sheridan Road, Evanston, IL 60208-0834 USA
{jw, forbus}@northwestern.edu

Abstract

Sketching is a powerful modality for thinking through, and communicating about, mechanical designs. Qualitative mechanics reasoning has been applied to sketched input before but not without succumbing to limitations of domain-based recognition or requiring complex annotation and additional explicit knowledge from the user. This paper presents solutions to three problems in integrating qualitative mechanics reasoning with a sketch understanding platform: identifying objects and forces, discovering regions of interaction between them, and understanding the effects of these interactions. We show how the spatial knowledge available in a conceptually labeled sketch is sufficient to solve the first two problems and enables the use of qualitative mechanics to solve the third. Examples from an implemented system are used for illustration. This system is the first step in a plan to create a sketch understanding system capable of providing corrective feedback for sketched engineering designs.

Introduction

Sketching is a valuable way to work through ideas and communicate them to others. This is especially true in conceptual design, when the feasibility of basic ideas is under consideration. Qualitative reasoning seems a natural fit for such tasks, since precise details are typically unavailable during this stage of design. However, existing sketch understanding systems do not currently provide much support for these tasks. Some existing systems rely on animation for feedback. In (Alvarado & Davis 2007), ink recognition is used to set up input for a mechanics simulator, much the way that Quickset (Cohen *et al* 1997) was used to set up scenarios for a military simulator. Unfortunately, it is not clear how useful such animated output is for design tasks: If a student's design doesn't work, all of the knowledge about why something failed is hidden within the procedures of the off-the-shelf physics engine. In contrast, qualitative causal models can be used to provide explanations, both to designers and especially to engineering students learning to design. The SketchIt approach of Stahovich *et al* (1998) provides qualitative reasoning about sketches of mechanical systems that seems to be more useful in such tasks. However, SketchIt required the human user to identify surfaces of interest and describe the range of their interaction manually. This is tedious for experienced designers and not feasible for novice designers.

This paper describes how we are using a combination of qualitative mechanics and open-domain sketch understanding to enable software to reason about forces, mechanical constraint, and motion from hand-drawn sketches. We begin with our motivating context, coaching entering engineering students in how to use sketches to communicate. We then summarize CogSketch, the sketch understanding system we are using as a platform, and the qualitative mechanics theories we are building upon. We describe how we are embedding qualitative mechanics reasoning into sketch-based representations. We identify and present solutions for three problems in carrying out this integration: identifying objects and forces, discovering regions of interaction between them, and understanding the effects of these interactions. We describe some examples of our system's reasoning, and close with related and future work.

Helping Students Learn to Communicate

Communication is an important skill for engineering students. At Northwestern, 1st and 2nd year engineering students take Engineering Design and Communications, which teaches both skills in an integrated manner. Students working in teams of three or four tackle problems for real clients. Examples include patients at the Rehabilitation Institute in Chicago, who need new tools to help them achieve everyday tasks, like chopping vegetables or trimming their nails, despite physical handicaps. Students build prototypes of their designs to explore particular issues, with regular feedback from potential users. Conversations with instructors revealed that one significant problem they had was helping students learn to use their sketches to communicate ideas, both within the team and to clients. We are creating a sketch-based system to address this problem.

The idea of the *Design Buddy*, as we are calling it, is to be a "crash test dummy" for students to practice explaining their designs. They will sketch their ideas, explain the parts, what they are made of, their intended behaviors, and the intended functional roles of the parts. The system will reason through the possible behaviors itself, based on its understanding of qualitative mechanics, materials, and

everyday actions. It will compare its predictions of behaviors with the student's intended behaviors, and ask the student questions about discrepancies. These may include the student not mentioning some critical aspect of the behavior in their explanation, or predicting a behavior that the system does not think is possible. This is a very challenging task, for three reasons: (1) The qualitative mechanics reasoning must be very general and robust. The design projects change constantly, and a wide range of problems arise. (2) The interface must be both sufficiently natural to not be a distraction, and must incidentally help the student learn to explain things in terms that practicing engineers would use. (3) The coaching software must have enough strategies to provide students with effective help in learning how to think about their particular design and the design process itself. This paper focuses on a particular aspect of the problem (1), namely, doing robust qualitative mechanics reasoning in a sketch-based environment.

Background

We briefly review the relevant aspects of CogSketch, our sketch understanding system, and prior work on qualitative mechanics.

CogSketch

The platform we are using is CogSketch, a publicly available sketch understanding system built on the nuSketch architecture (Forbus *et al.* 2004). CogSketch is an open-domain system. Most sketch understanding systems equate understanding ink with classifying it as a symbol drawn from a fixed vocabulary of items. By contrast, nuSketch systems treat recognition as a catalyst, not a necessity. Pieces of ink (called *glyphs*) can be labeled by the user with concepts drawn from a large underlying knowledge base[1]. CogSketch automatically computes a number of visual relationships between glyphs, such as topological relationships (using RCC8 (Cohn, 1996)) and positional relationships (e.g., above, left, etc.). CogSketch can also analyze the ink within a glyph, identifying straight-line segments within the ink, corners, and so forth.

Annotation glyphs are used to provide additional information about other glyphs. Examples of annotation glyphs include applied forces, axes of rotation, and centers of mass. Arrows are automatically recognized when drawing annotation glyphs, so that, for example, the orientation and position of application of force can be automatically computed.

[1] We use the contents of OpenCyc, containing over 58,000 concepts, plus our own group's extensions for qualitative reasoning and analogy.

Qualitative Mechanics

Our model of qualitative mechanics is based on work done by Nielsen (1989) and Kim (1993). Nielsen's work represents direction and orientation in terms of qualitative vectors (*q-vectors*). For a 2-d space, there are 8 translational q-vectors (right, up, left, down and one for each quadrant in between) and two rotational q-vectors (clockwise and counter-clockwise). All forces/torques and movements are expressed in these directions.

Objects are represented as sets of surfaces. Each surface has a parent object, a direction outward from the object (surface normal) and a direction towards the center of rotation of the object. Representations for translational and rotational freedom and constraint are used to state if/when an object can move or rotate. Forces and motion are transmitted from object to object through their surface contacts, with the net force and net motion determining the final motion of the object in a given state.

Nielsen's representations supported envisionment of a variety of mechanical systems, including clocks (Forbus *et al* 1991). Kim's work adds several more representations to Nielsen's, including notions of bounded stuff and flow fields. This enabled Kim's system to reason about non-rigid bodies. The end result was a system which could understand lift pumps, laminar flow over surfaces, and automobile engines. However, neither of these systems dealt with hand-drawn inputs. Kim's system assumed predicate calculus input representations. Nielsen's system could accept as input scanned images of parts, and automatically simplify the configuration space it computed to handle noise. For example, it was able to change resolution to see a gear train as having only one degree of freedom, despite noise in the scanned input. However, scanned photographs are far more accurate than hand-drawn sketches, making hand-drawn sketches an even harder problem. Adapting these qualitative mechanics ideas to work with hand-drawn sketches is the challenge this paper addresses.

In this paper, we restrict ourselves to a subset of qualitative mechanics, namely rigid-body mechanics involving 2D polygonal bodies. This is an important first step, because it allows for a wide variety of test cases while requiring only a subset of the qualitative mechanics.

Interpreting Rigid-Body Sketches

In this section we begin with an overview of the three main problems we encountered when integrating qualitative mechanics with a sketch understanding system. We explain our solutions to these problems through our system. This includes an overview of the sketching process, an explanation of how the sketch is translated into a qualitative representation, and how inference is used to answer questions about the sketched system.

Issues with Integrating Sketched Input

The three main problems we encounter when interpreting rigid-body sketches are: identifying objects and forces, discovering regions of interaction between them, and understanding the effects of these interactions. The first problem has often been approached before using recognition. However, in a completely open domain (even limited to two-dimensional polygons) this is not an option. Rather, we must make do with other clues that a human would understand from looking at a sketch. This includes the knowledge that arrows are not objects themselves but rather information that describes or affects the objects in the sketch.

Identifying how objects in a sketch interact has been approached by annotations—for instance, specifying objects be drawn with terminals connecting them. In a 2-d rigid-body system the "terminals" are surface contacts. At first this appears trivial, but as demonstrated in **Error! Reference source not found.** work must be done to disambiguate the exact nature of the interaction in a surface contact. As Nielsen (1988) showed, the decomposition of a surface into qualitatively distinct regions depends upon mechanical constraints beyond just the surface shape.

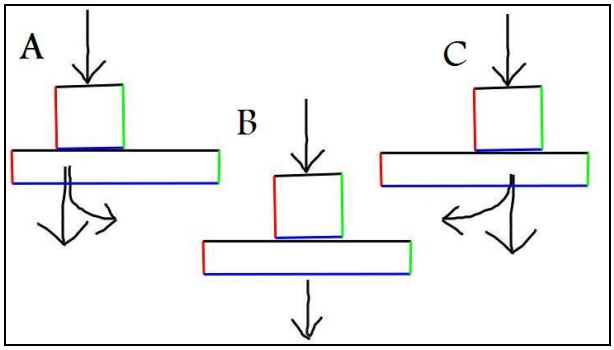


Figure 1: Slightly different positions of contact allow very different motions.

The problem of understanding the effects of these interactions is more easily solved if the representation generated by the first two steps provides the appropriate qualitative decomposition of surfaces. Next we explain how we help users generate sketches, how we construct the decompositions of surfaces, and how qualitative mechanics is applied to these representations to answer questions about the sketched mechanism.

The Sketch

The user draws the forces and objects of their mechanism as CogSketch glyphs. They label the glyphs with concepts from its knowledge base. Our system looks for four labels of glyphs in particular:

- `RigidPhysicalObject`
  Because we are working in a rigid-body domain, this category identifies all objects of interest.
- `FixedPhysicalObject`
  This category represents objects that are completely constrained from moving or rotating (e.g. the surface of the earth, walls).
- `ForceArrow`
  Force arrows are used to indicate external forces acting on the system. This includes global forces such as gravity.
- `RotOrigin`
  An annotation glyph, marking the point around which its parent object is free to rotate. An object with a `RotOrigin` is prevented from translating in all directions.

From a user interface perspective this might seem like a lot of required labeling. However, our system makes some simplifying assumptions. First, we take advantage of CogSketch's arrow recognition and assume any two or three stroke arrow glyph is a `ForceArrow`. Any annotation glyph that is not an arrow is assumed to be a `RotOrigin` for its parent glyph. Finally, all remaining glyphs are assumed by default to be `RigidPhysicalObject`. Thus the only specific labeling required of the user is the annotation glyph distinction and labeling fixed objects as `FixedPhysicalObject`.

After they are done drawing their sketch, the user can perform any of the following queries:

- Will object x move?
- Will object x rotate?
- What forces are on object x?
- Is the sketched system stable? (Will any object move?)

Performing a query begins the translation process.

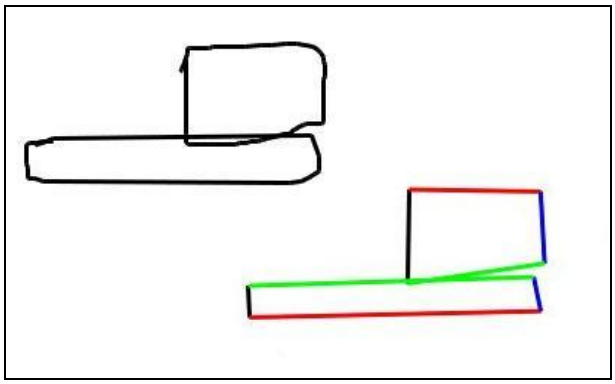


Figure 2: Blocks in a raw sketch (top left) are decomposed into idealized edges (bottom right), simplifying surface contact detection.

Translation to QM Representation

When prompted with a query, the system begins interpreting the sketch. It is here we solve the first two of the three problems addressed in this paper: identification of objects, forces, and their properties, and discovering regions of interaction.

Identifying Objects Forces, and their Properties. The first step is to know what objects and forces are depicted in the sketch, and determine their specific properties. For forces, these properties include direction and the objects they directly affect. For each object this includes identifying the surfaces of that object, and determining whether the object is fixed, fixed-axis, or free to move. If the object is fixed-axis then its center of rotation must be located.

First we check the labeling for each glyph in the sketch to see if they are a rigid object and/or a fixed rigid physical object. This gives us the fixed property. Then, each glyph representing a rigid object is decomposed into edges (Figure 2), and each edge becomes reified as one or more surfaces in our predicate representation. Every surface also has a normal vector pointing outward from the object and a q-vector towards the object's axis of rotation. The system is limited to processing polygons, so every edge will be a line segment. Thus we can infer that each edge will have only one normal but may have *multiple vectors towards the axis of rotation.* Since the edge is a line segment there will be up to five surfaces per edge. The cases of one, two, and three edges are demonstrated in **Error! Reference source not found.**.

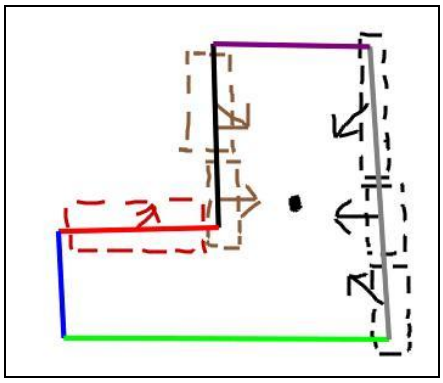


Figure 3: Straight edges of objects can be divided into up to five qualitative surfaces, each with a different q-vector in the direction of the axis of rotation.

If the axis of rotation is not given in the sketch, we assume a uniform density and choose the center of area of the shape as the axis of rotation. While multiple axes of rotation may exist over the course of a mechanism's operation, our system currently only analyzes one instant of time. Thus, we choose the axis active at the current moment indicated by the sketch and ignore the others.

Having finished with objects, we move on to forces. The force represented by each force arrow in the sketch is reified as either a force or a torque. If the force is global then a force is reified with a qvector matching the direction of the arrow. If the force arrow is on a specific surface then whether it creates a force or torque depends on the `OriginDir` of that surface. If the arrow is the inverse of its surface normal then a force is applied, otherwise a torque is applied in the appropriate direction. In **Error! Reference source not found.** applying a leftward force arrow to the top-right surface would create a counter-clockwise torque. Applying the same force arrow to the surface just below that would create a leftward force.

If the force arrow is an annotation glyph, a force or torque is added for each object that glyph annotates. If it is not an annotation glyph, the force is added for all objects in the sketch. This allows for global forces such as gravity to be input easily.

Discovering Regions of Interaction. This problem entails finding all surface contact relationships. This is done by first identifying all rigid objects which may be in contact. This step is straightforward since CogSketch automatically computes topological relations for each pair of glyphs in the sketch. For each pair of intersecting or connected objects, their edges are checked pairwise for contact. If two edge are nearly parallel, in close proximity to each other, and overlap by a significant amount (that is, not merely in a line one after another), they are considered in contact. If the edges contain multiple surfaces, the overlap is calculated and surface contact is only reified for the two surfaces which contain share the midpoint of the overlap.

Once all of the surfaces, surface contacts, and forces have been identified, the system is ready to make the inferences required to answer the user's query.

Answering User Queries

After the qualitative representation is complete, the system begins finding the answer to the user's query. We now are at the problem of understanding the effects of the interacting regions. The user query is passed to our backchainer, whose rules are an implementation of QM theory. These rules are written as Horn clauses in which the first statement is the consequent and the conjunction of the remaining statements is the antecedent. Some of the rules are listed here:

Constraining translation for fixed objects:

```
(<== (transConstraint ?obj ?dir)
     (isa? obj FixedPhysicalObject))
```

Determining motion constraint in a particular half-plane:

```
(<==(sufficientlyConstrained ?obj ?dir)
     (transConstraint ?obj ?dir1)
     (transConstraint ?obj ?dir2)
     (transConstraint ?obj ?dir3)
     (openHalfPlane ?dir ?dir1)
     (openHalfPlane ?dir ?dir2)
     (openHalfPlane ?dir ?dir3)
     (different ?dir1 ?dir2 ?dir3))
```

An open half-plane is defined in Nielsen's work as the set of qvectors within 90 degrees of a given direction, excluding those at exactly 90 degrees. So for direction Left, the open half-plane would contain directions Quad1 and Quad4 but not Up or Down. The above rule defines "sufficiently constrained" in a direction if motion is constrained in all directions in that direction's half-plane.

Tranferring constraint through surface contacts:

```
(<== (transConstraint ?obj1 ?dir)
     (hasSurface ?obj1 ?s1)
     (hasSurface ?obj2 ?s2)
     (surfaceContact ?s1 ?s2)
     (surfaceNormal ?s1 ?sn)
     (sufficientlyConstrained ?obj2 ?sn)
     (openHalfPlane ?sn ?dir))
```

Object 1 cannot move in direction `dir` if object 2 is constrained in all directions in that `dir`'s half-plane. Otherwise object 2 can move in one of those directions, allowing object 1 to move in `dir`.

Freedom is the absence of constraints:

```
(<== (transFreedom ?obj ?dir)
     (isa ?obj RigidOb)
     (isa ?dir 2DQVector)
     (evaluate ?x
      (CardinalityFn
       (TheClosedRetrievalSetOf ?dir
        (transConstraint ?obj ?dir))))
     (equals ?x 0))
```

Force + Freedom causes motion:

```
(<== (transMotion ?obj ?dir)
     (force ?obj ?dir)
     (transFreedom ?obj ?dir))
```

The force predicate here is the net force on the object. In the current version this must be specified by the user when the direction of the net force is ambiguous.

Transfer of translation across surfaces:

```
(<== (transMotion ?obj2 ?d2)
     (hasSurface ?obj1 ?s1)))
     (hasSurface ?obj2 ?s2)))
     (surfaceContact ?s1 ?s2)))
     (surfaceNormal ?s2 ?sn)))
     (inverseVector ?sn ?invsn))
     (openHalfPlane ?invsn ?d1))
     (transMotion ?obj1 ?d1))
     (openHalfPlane ?invsn ?d2))
     (transFreedom ?obj2 ?d2)))
```

This rule stipulates conditions in which object 2 will move because of contact with another moving object, object 1. The `openHalfPlane` relation means that the two directions are within 90 degrees of each other. This allows an object to transfer motion through multiple directions if necessary.

Force applied to a surface via surface contact:

```
(<== (forceApplied ?s ?sn ?obj1)
     (force ?obj1 ?dir)
     (hasSurface ?obj1 ?s1)
     (surfaceContact ?s1 ?s)
     (surfaceNormal ?s1 ?sn)
     (openHalfPlane ?sn ?dir))
```

Force applied to a surface causing force on object:

```
(<== (force ?obj ?dir)
     (hasSurface ?obj ?s)
     (forceApplied ?s ?dir ?c))
```

Forces are also translated through other objects. In general, every force applied through a surface contact gets applied to the next object as a translational force if both of the following conditions hold:

1) The object is free to translate.
2) The inward normal of the contact surface points towards the object's axis of rotation.

In the version presented here, it is up to the user to resolve ambiguities in the forces. The work in progress includes rules that try to find the resultant vector of a set of forces, and resolve ambiguities by asking the user which forces are larger or by using the magnitude field of the force annotation in CogSketch.

Torque propagation is not yet implemented in the current version of the system but it will follow the same principles. These and other qualitative mechanics principles are all defined as rules. By backchaining through these rules, the system deduces what forces are acting on objects and whether they will move.

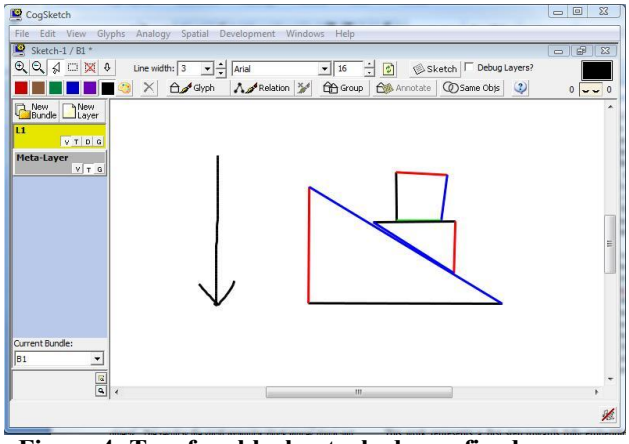


Figure 4: Two free blocks stacked on a fixed ramp. The arrow on the right represents a global downward force affecting all three objects. The result is the small triangular block moves down and right (quad 4) and the square block moves down.

The example shown in Figure 4 is based on an equivalent one in Nielsen's work. The sketch contains a ramp with two blocks stacked one upon another and one arrow pointing downward, drawn off to the left. The arrow is not an annotation glyph, and the ramp is labeled as a `FixedPhysicalObject`. When the user asks for the motion of all the objects in this sketch, the system begins to build its QM representation. First, each of the three non-arrow glyphs is decomposed into their respective edges, which become their surfaces. Since the force arrow is not

annotating a specific glyph, its downward force is assumed to be affecting all glyphs in the sketch.

Next, the system searches for surface contacts by checking each pair of objects for contact or overlap. Using the topological relations calculated by CogSketch, the system finds overlap between the ramp and the triangular block and between the triangular block and the square. For both pairs it performs a line-line proximity comparison between their surfaces to find the surface contacts.

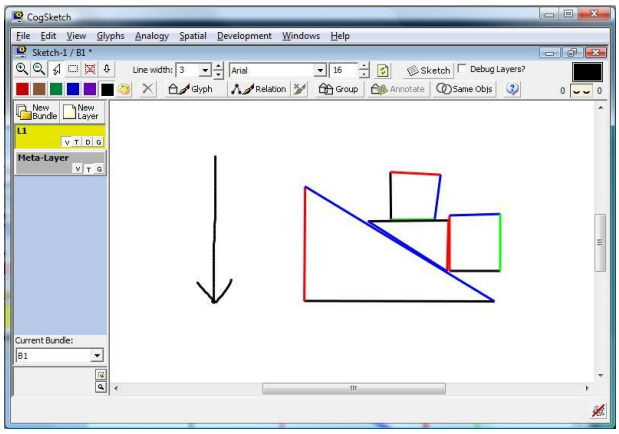


Figure 5: The two free blocks (Figure 4**) are now constrained by an additional block, labeled as fixed.**

With the surface contacts reified and the forces applied, the system begins backchaining to find any motion that each object possesses at the moment pictured. The ramp is a `FixedPhysicalObject` so it is stationary. The square block has a downward force on it, and because the triangular block is not completely constrained from moving in the downward half-plane, the square block will begin moving downward. The triangular block has a downward force and is free to move in the down-right direction; consequently, it does. Adding a stop block to the above example (see Figure 5) prevents the triangle from moving, and thus prevents the square block from moving.

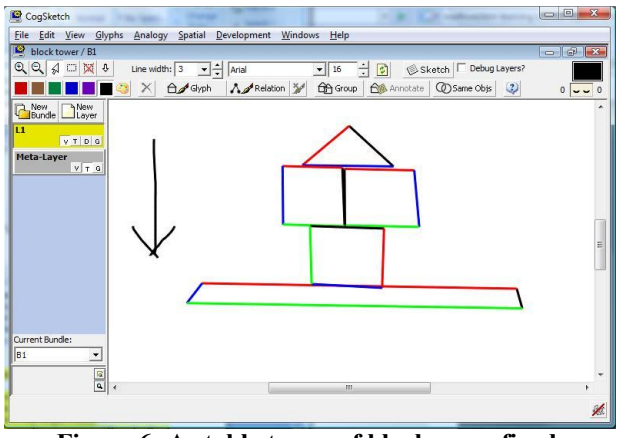


Figure 6: A stable tower of blocks on a fixed platform.

Error! Reference source not found. shows more transfer of motion constraints. The constraint is propagated upward from the fixed base to the topmost block. If the user were to draw this and query for motion, the system would return no motion. In this way, one can test a design for stability.

Related Work

Our approach of using shape edge decomposition, RCC8 relations, line-line proximity and overlap calculation is a novel solution to the problem of discovering and analyzing surface interactions between rigid bodies. Kurtoglu and Stahovich (2002) used line-line and other proximity pairs in a system to identify the type of connection between two sketched objects, with the goal of classifying those objects in categories such as rigid body or electric motor. Our system goes two steps further for rigid bodies, breaking them down into their individual surfaces and then determining exactly how the position and size of an overlap of two surfaces affects their motions.

Prior sketching systems for mechanical reasoning have relied on human input for the analysis of surface contacts. The QM theory on which our system is based had full propositional representations as its input. (Nielsen 1989; Kim, 1993) Later systems such as SketchIT (Stahovich et.al. 1998) required the designer to mark the important surfaces and build state machines describing their interactions. In the example in Figure 4 SketchIT would require the user to highlight the contact surfaces.

Progress has also been made in the area of automatically recognizing the objects in sketches (Alvarado & Davis, 2004; Kurtoglu & Stahovich, 2002). By eliminating the need for extra human input we have moved closer to a sketch-understanding system that can reason deeply about hand-drawn sketches.

Discussion and Future Work

This work represents a first step towards fully embedding qualitative mechanics in systems that reason with hand-drawn sketches. This required tackling the problems of identifying objects, forces, and their properties; discovering the interactions between said objects and forces; and finally, computing the exact effects of these interactions. The advances which enabled us to do this include using a combination of shape decomposition, RCC8 relations, and line-line proximity and overlap calculation, allowing us to identify the different surfaces of two-dimensional objects, their areas of contact, and compute the consequences of those interactions.

Our goal is to have a complete, robust qualitative mechanics reasoner that can operate over a wide range of hand-drawn sketches. We see two key next steps. First, handling curved surfaces is important for many kinds of designs. This presents new challenges to segmentation. Second, the system currently only reasons about

instantaneous force/motion transfers. Reasoning about motion over time, including automatically deducing plausible changes in contacts (cf. Nielsen 1988), is also important. Longer term, we plan to extend the system to handle 3D shapes, flexible bodies, laminar flow situations, and fluids as well as rigid bodies.

Acknowledgements

This work was supported by NSF SLC Grant SBE-0541957, the Spatial Intelligence and Learning Center (SILC).

References

Alvarado, C., Davis R. 2004. Multi-domain sketch understanding. Massachusetts Institute of Technology, Cambridge, MA.

Alvarado, C., Davis R. 2007. Resolving ambiguities to create a natural computer-based sketching environment. In *International Conference on Computer Graphics and Interactive Techniques ACM SIGGRAPH 2007 courses*. San Diego, California.

Cohen, P., Johnston, M., McGee, D., Oviatt, S., Pittman, J., Smith, I., Chen, L. and Clow, J. 1997. QuickSet: Multimodal interaction for distributed application. In *Proceedings of the Fifth Annual International Multimodal Conference*, 31-40. Seattle, WA.

Cohn, A. 1996. Calculi for Qualitative Spatial Reasoning. In Calmet J., Campbell J. A., Pfalzgraph J., Verlag S. (Eds.), *Artificial Intelligence and Symbolic Mathematical Computation*, LNCS 1138, 124-143.

Forbus, K., Nielsen, P., and Faltings, B. 1991. Qualitative spatial reasoning: The CLOCK project. In *Artificial Intelligence*, 51(1-3).

Forbus, K., Lockwood, K., Klenk, M., Tomai, E., and Usher, J. (2004). Open-domain sketch understanding: The nuSketch approach. Proceedings of the AAAI Fall Symposium on Making Pen-based Interaction Intelligent and Natural, October, Washington, D.C.

Kim, H. (1993). Qualitative reasoning about fluids and mechanics. Ph.D. dissertation and ILS Technical Report, Northwestern University. Evanston, IL.

Nielsen, P.E. (1988). A qualitative approach to rigid body mechanics. (Tech. Rep. No. UIUCDCS-R-88-1469; UILU-ENG-88-1775). Urbana, Illinois: University of Illinois at Urbana-Champaign, Department of Computer Science.

Stahovich T.F., Davis R., Shrobe H. 1998. Generating multiple new designs from a sketch. In *Artificial Intelligence* 104 (1998) 211–264.

Kurtoglu T., Stahovich T.F., 2002. Interpreting Schematic Sketches Using Physical Reasoning, In *AAAI Spring Symposium on Sketch Understanding*, AAAI Technical Report SS-02-08, 78-85.

Learning Qualitative Models from Image Sequences

Jure Žabkar and **Ivan Bratko**
University of Ljubljana,
Faculty of Comp. and Inf. Science,
Tržaska 25,
SI-1000 Ljubljana, Slovenia

Gregor Jerše
University of Ljubljana
Faculty of Mathematics and Physics
Jadranska 19,
SI-1000 Ljubljana, Slovenia

Johann Prankl and **Matthias Schlemmer**
Vienna University of Technology,
Automation and Control Institute,
Gusshausstrasse 27-29 / E376,
A-1040 Vienna, Austria

Abstract

In this paper, we describe the autonomous learning of qualitative models with a robot's on-board vision. Those models are used to describe spatio-temporal qualitative relations between observed objects. Therefore, the algorithm QING is described which extracts the necessary qualitative relations between the objects from the sequence of images. The robot uses these features together with other sensory data to learn about the environment.

Keywords: qualitative (spatial) reasoning, cognitive vision, cognitive robotics

Introduction

In this paper we tackle certain problem from the field of cognitive robotics, namely how the robot can use its on-board vision for autonomous learning. This problem is highly connected to the field of cognitive vision as the robot should somehow reason about the information it gets from image sequences. Following the definition of (Vernon 2008), "A cognitive vision system can achieve the four levels of generic computer vision functionality of detection, localization, recognition, and understanding." Whereas classical computer vision is mainly concerned with the first three points, the last issue affords interdisciplinary work in order to integrate higher-level reasoning functions. This work aims at incorporating a specific machine learning technique in order to qualitatively reason about the arrangement of objects. The abstraction from quantitative pixel data to a more qualitative layer seems to be of great importance to cognitive vision. In this abstraction step, the vision part is concerned with segmenting the image to proto-objects (groupings of pixels that are likely to belong to the same object). In this paper, we are mainly concerned with the learning part, therefore the proto-object grouping is assumed to be given. However, higher-level qualitative reasoning is highly relevant for providing feedback to the vision part, as its ability to predict the existence and the arrangement of proto-objects in the subsequent image(s) can support low-level image processing.

From a roboticist's perspective, a qualitative model at any layer can help interpreting a given situation. This paper tries to bridge this link for one of the lowest, namely the perceptual layer, providing a model for visual scene interpretation. Motivation for this work comes from the European project XPERO, in which a robot should gain insights about the real world by experimenting and meaningfully relate it's intero- and exteroceptive information so to arrive at a level of qualitatively understanding its environment.

The main idea of this paper is to apply the algorithm QING (see corresponding Section) to extract qualitative spatio-temporal features from an image sequence and use them together with other sensory data for autonomous learning about the concept of occlusion. In this paper we present an artificial scenario in which the robot circles around two balls of different colour and builds a qualitative model in the form of a qualitative non-deterministic finite automaton (qNFA). The robot learns autonomously without any external intervention. The final model enables us as well as the robot to reason about the occlusion, e.g. it tells us that it is not possible to go from the state of non-occlusion directly to the state of total occlusion but rather through the state of partial occlusion. Our basic goal is to build a system which the robot could use for reasoning about its visual input and based on this reasoning improve its visual perception, e.g., detecting regions of interest. Our system can discover qualitative relations between the objects in the images and how these relations change over time. Currently, it is capable of discovering topological relations.

As this paper focuses on the use of QING for learning the relations, we will not describe the vision part. We must mention though that the extraction of the "interesting" colour blobs from the images can be motivated by a simple curiosity mechanism. For a robot tuned to learn about sensory input it has not seen before, colour is a strong cue. More elaborate techniques, such as the bottom-up Grouping of line features, as described in the next Section, can be applied as well.

Algorithm QING

QING (Žabkar *et al.* 2007) is an algorithm for qualitative analysis of continuous class variable f w.r.t. given attributes $(x_1, \ldots, x_n)$, where n is the dimension of the attribute space. For simplicity we will in this short descrip-

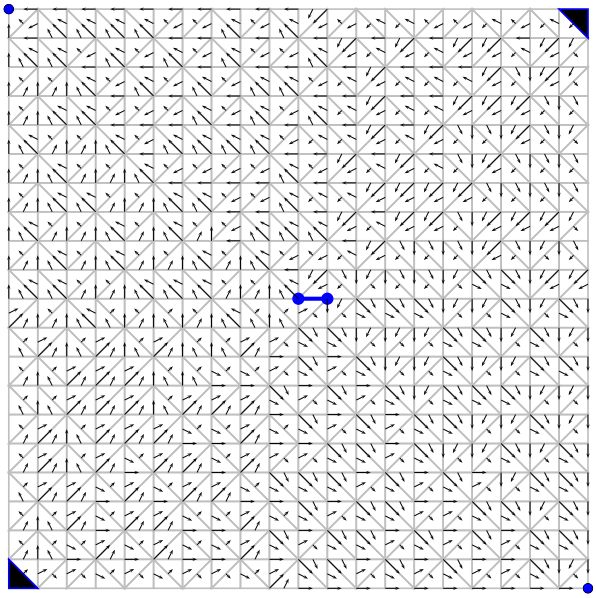


Figure 1: Qualitative field for $f(x, y) = xy$. The arrows point in the direction of function decrease.

tion restrict ourselves to two attributes. Theoretically, QING works for any dimension n, but is practical for $n \leq 5$ due to the complexity of triangulation.

To be more illustrative, we accompany the description of the algorithm with a simple example $f(x, y) = xy$ defined on an orthogonal mesh (see Fig. 1) on the domain $[-10, 10] \times [10, 10]$. Learning examples are represented as points in the attribute space, each point having assigned a value of its class variable. The domain is triangulated in order to be analysed with discrete Morse theory. Critical points, i.e. maxima, minima and saddles, are reconstructed using the algorithm of (King, Knudson, & Mramor Kosta 2005). The output of QING is a qualitative field (Fig. 1), a set of critical points and a labeled qualitative graph (Fig. 2), which is a visualisation of the qualitative model. Detailed definitions of these terms are given in (Žabkar *et al.* 2007). The main difference between QING and other algorithms for induction of qualitative models is in attribute space partitioning. Unlike algorithms that split on attribute values (e.g. trees, rules), QING triangulates the space (domain) and constructs a qualitative field which for every learning example tells the directions of increasing/decreasing class.

The example image, in the experiment that we describe in the next Section, is processed in a similar way. However, to capture the time, we need to connect the neighboring images in the sequence. To do this, we use parametric Morse theory with time as a parameter and we follow the critical simplices through the slices as described in (King, Knudson, & Mramor Kosta 2007).

Experiments

We performed the experiments on artificial data in a domain in which the robot circles around a red and a blue ball as

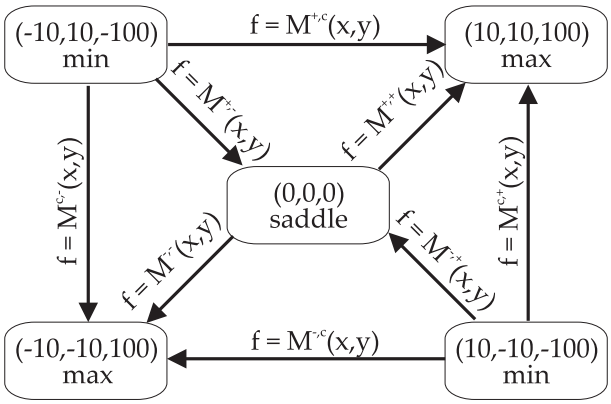


Figure 2: Qualitative graph for $f(x, y) = xy$.

shown in Fig. 3. The robot uses an overview camera to measure the distances to the balls ($bdred$ and $bdblue$) and it uses an on-board camera to observe the balls and collect the data for learning a qualitative model.

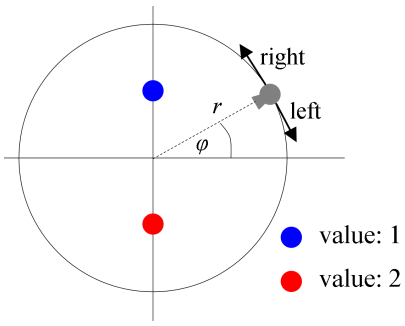


Figure 3: The robot circles around the balls and observes them with its on-board camera.

The robot is also aware of its polar coordinates, so it knows about its angle and the radius. Its actions are *left* (clockwise) and *right* (counter clockwise along the circle as shown in Fig. 3). The robot observes the qualitative change of its distances to the red and the blue ball w.r.t. the action. For example, if the robot resides at $(\varphi = 0\,^\circ, r)$ and chooses to go right, i.e. φ increases, the distance to the red ball would increase while the distance to the blue ball would decrease, $bdred = Q(+\varphi)$ and $bdblue = Q(-\varphi)$. The robot can observe similar relations in the image sequence. The relations that it can detect on a simple image of two balls are the following (see also Fig. 4):

- the red ball and the blue ball do not touch
- the balls touch
- only the red ball is visible
- only the blue ball is visible

Inside QING algorithm, the objects are distinguished by their colour and each colour is represented by a unique numerical value. In our example, we define that the value 1

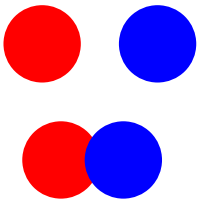


Figure 4: The images from the on-board camera where the balls don't touch (top) and when they overlap (bottom).

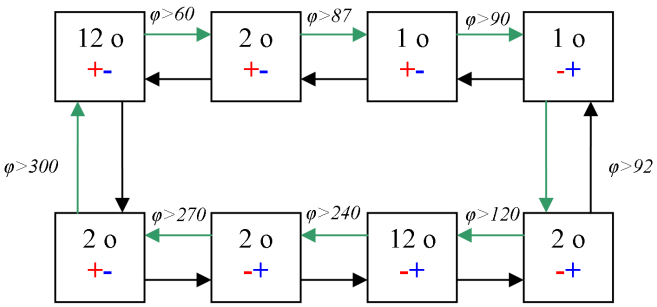


Figure 5: The learned qualitative NFA describing how the robot can change its states with the chosen actions (right...green arrows; left...black arrows). The red signs stand for $bdred = Q(s\varphi)$ and the blue signs are $bdblue = Q(s\varphi)$.

stands for the blue color and the value 2 stands for the red one. The background has value 0. QING constructs a discrete vector field in a 3-dimensional space (2D image and time) and assigns the appropriate values to each pixel, according to the color of the pixel. Although QING can handle noise well, there was no noise in our artificial data. To obtain the qualitative relations QING computes extreme points in this space and uses the discrete vector field to track the movement of these extrema over time.

When the balls don't touch, QING finds two maxima with values 1 and 2 (we mark such a state with '12'). If only one of the balls is visible, it finds either 1 (for blue) or 2 (for red), while if they touch it finds one maximum with value 2. In our image sequence, the changes are rare since most of the time the image at angle φ_i is the same as φ_{i+1}. This, we denote as 'o', meaning there is no change, i.e. steady. Considering the type of the qualitative relation in each image and $bdred = Q(+\varphi)$ and $bdblue = Q(-\varphi)$ from above we build a class value for each learning example, e.g. 12o +-, meaning that the balls stay separated, the distance to the red ball increases (the first sign) and the distance to the blue ball decreases (the second sign).

Results

The learned model is in the form of qualitative nondeterministic finite automaton (NFA) as shown in Fig. 5. Its states are qualitative descriptions of the observations derived from the image features and other available attributes, i.e. distances to both balls and the angle φ. The transitions explain the possible changes of states given an action. Nondeterminism is hidden in the fact that the same action applied in the same state may result in the same state or the neighbouring one, i.e. self transitions are always possible. This is due to the qualitative descriptions of the states. However, such NFA gives us enough information to reason qualitatively about the system. We can observe that total occlusions ($\varphi = 90$ or $\varphi = 270$, changes of $+, -$ signs) may only happen from partial occlusions (states with *2o*).

Related work

Many authors have addressed the problem of qualitative spatial or spatio-temporal reasoning. (Cui, Cohn, & Randell 1992) describes an envisionment-based qualitative simulation program that can reason about space and time, considering the topological relations between objects. Learning temporal patterns from unannotated video data is presented in (Fleischman, Decamp, & Roy 2006). Well known theoretical approaches to qualitative spatio-temporal reasoning are described in (Cohn & Hazarika 2001) and (Randell, Witkowski, & Shanahan 2001). The latter is especially interesting for us as it considers spatial occlusion. (Cao, Mamoulis, & Cheung 2005) discovers sequential patterns in a spatio-temporal series of movements of mobile objects. An interesting approach to mining temporal patterns in multivariate time series, using Unification-based Temporal Grammars is described in (Mörchen & Ultsch 2004). It only considers the temporal dimension but there seems to be no reason against applying a similar technique to spatial dimensions. On the other hand, (Bailey-Kellogg & Zhao 2004), (Lundell 1994) and (Faltings 1995) study only qualitative spatial reasoning.

Concerning the vision part, literature on computer vision is extremely diversified, wherefore we are focusing here on low-level algorithms powerful enough to support the task at hand as well as State-of-the-Art attempts to fuse qualitative reasoning with computer vision.

For grouping pixels to likely objects (so-called proto-objects), a recent work is (Zillich 2007). In this work, edges are grouped based on Gestalt principles, e.g., continuity and proximity. Using a parameter-free anytime algorithm, this tool is capable of delivering the most likely locations of proto-objects in terms of closures and ellipses very fast. Alternatively, colour-based segmentation can be used, for example the graph-based method of (Felzenszwalb & Huttenlocher 2004).

Work on fusing qualitative reasoning with vision techniques has been done by (Bennett *et al.* 2008). In this paper, the authors recognise and track multiple objects throughout a scene (e.g., basketball players) supported by a reasoning about the spatio-temporal continuity. (Huang & Essa 2005) are tracking multiple objects through complex occlusion situations, where a colour blob tracker is backed by a reasoning step of where currently unseen objects are. Their task is very similar to ours except they are using genetic algorithms to match the objects from one image to the next one while

we use parametric discrete Morse theory to do this.

Discussion and future work

The above work shows a promising direction towards an autonomous robot system with on-board vision that could learn from the vision input as well as improve on visual perception using qualitative models. We believe that our approach can help the robot extract dynamic features from its vision system and use them in qualitative models. Using these features the robot can detect the region of interest in its environment. The latter is especially interesting combined with the task of embodied learning by experimentation where regions of interest may drive the robot to interact with the world.

From the technical perspective, our future work will include further improvement of the QING algorithm. We would also like to investigate how the vision part can make use of qualiative models, e.g. to improve the image segmentation.

Acknowledgment

The work described in this article has been funded by the European Commission's Sixth Framework Programme under contract no. 029427 as part of the Specific Targeted Research Project XPERO ("Robotic Learning by Experimentation").

References

Bailey-Kellogg, C., and Zhao, F. 2004. Qualitative spatial reasoning extracting and reasoning with spatial aggregates. *AI Magazine* 24(4):47–60.

Bennett, B.; Magee, D. R.; Cohn, A. G.; and Hogg, D. C. 2008. Enhanced tracking and recognition of moving objects by reasoning about spatio-temporal continuity. *Image Vision Computing* 26(1):67–81.

Cao, H.; Mamoulis, N.; and Cheung, D. W. 2005. Mining frequent spatio-temporal sequential patterns. In *ICDM*, 82–89. IEEE Computer Society.

Cohn, A., and Hazarika, S. 2001. Continuous transitions in mereotopology.

Cui, Z.; Cohn, A. G.; and Randell, D. A. 1992. Qualitative simulation based on a logical formalism of space and time. In *National Conference on Artificial Intelligence*, 679–684.

Faltings, B. 1995. Qualitative spatial reasoning using algebraic topology. In *COSIT*, 17–30.

Felzenszwalb, P., and Huttenlocher, D. 2004. Efficient graph-based image segmentation. *International Journal of Computer Vision* 59(2):167–181.

Fleischman, M.; Decamp, P.; and Roy, D. 2006. Mining temporal patterns of movement for video content classification. In *MIR '06: Proceedings of the 8th ACM international workshop on Multimedia information retrieval*, 183–192. New York, NY, USA: ACM.

Huang, Y., and Essa, I. 2005. Tracking multiple objects through occlusions. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, 1051–1058.

King, H. C.; Knudson, K.; and Mramor Kosta, N. 2005. Generating discrete morse functions from point data. *Exp. math.* 14(4):435–444.

King, H. C.; Knudson, K.; and Mramor Kosta, N. 2007. Birth and death in discrete morse theory. *Preprint*.

Lundell, M. 1994. Qualitative reasoning with spatially distributed parameters. In *International Workshop on Qualitative Reasoning about Physical Systems*, 13–20.

Mörchen, F., and Ultsch, A. 2004. Mining hierarchical temporal patterns in multivariate time series. In Biundo, S.; Frühwirth, T. W.; and Palm, G., eds., *KI*, volume 3238 of *Lecture Notes in Computer Science*, 127–140. Springer.

Randell, D. A.; Witkowski, M.; and Shanahan, M. 2001. From images to bodies: Modelling and exploiting spatial occlusion and motion parallax. In *IJCAI*, 57–66.

Vernon, D. 2008. Cognitive vision – the development of a discipline. online at: `http://www.eucognition.org/ecvision/about_ecvision/Cognitive_Vision.pdf`.

Žabkar, J.; Jerše, G.; Mramor, N.; and Bratko, I. 2007. Induction of qualitative models using discrete morse theory. In *Proceedings of the 21st Workshop on Qualitative Reasoning*.

Zillich, M. 2007. *Making Sense of Images: Parameter-Free Perceptual Grouping*. Ph.D. Dissertation, Vienna University of Technology.

Learning Qualitative Models by an Autonomous Robot

Jure Žabkar and **Ivan Bratko**
AI Lab, Faculty of Computer
and Information Science,
University of Ljubljana,
SI-1000 Ljubljana, Slovenia

Ashok C Mohan
University of Applied Sciences
Bonn-Rhein-Sieg
Grantham-Allee 20,
53757 Sankt Augustin, Germany

Abstract

In this paper we present a qualitative exploration strategy for an autonomous robot that learns by experimentation. Particularly, we describe a domain in which a mobile robot observes a ball and learns qualitative prediction models from its actions and observation data. At all times it uses these models to predict the results of the actions that it has decided to execute and to design new experiments that would lead it to learn a better model of the world, and for planning of the execution of these experiments. We experimentaly evaluate the exploration strategy.

Introduction

The idea of autonomous robots that are capable of learning by themselves, without any human intervention is one of the most fundamental goals of AI. Among several paradigms of learning, learning by experimentation demands no teacher, but rather learns autonomously, interacting with the real world. In this paper we present a showcase in which an autonomous robot is learning qualitative models by conducting experiments in its environment.

There are several ways of how the robot chooses its actions, designs and plans experiments. In order to learn efficiently, the strategy which it uses to explore its environment is very important. We propose a qualitative exploration strategy for autonomous robot learning. We evaluate our strategy by comparing it to random strategy. The results show that using our strategy, the robot is learning faster and it learns better models. We consider learning of *qualitative* models an important aspect. Qualitative models are easier to learn and sufficient to design and plan the experiment. They reduce the complexity of numerical models considerably and also enable humans to easily understand what the robot has learned.

The robot has no prior knowledge about its environment. In particular, it has no knowledge regarding the relations between its actions and observations. Its task is collecting the data and gradually learning a model which it immediately uses for moving and designing new experiments. Its goal is to learn a model that would relate its actions to its observations. At each step, the robot decides on one of several possible actions. It then uses its current model to predict the result of its action, executes the action and collects observations. It then compares its prediction with the actual observations. The result of this comparison leads to further experiments that helps to revise the model.

In the setting just described, we apply a new method, called parametric Padé, for learning qualitative models in dynamic domains. We present the method itself elsewhere but we provide a short description of the method in section "Parametric Padé" to keep this paper self contained.

Learning qualitative models by experimentation

Our task is to equip the robot with an exploration strategy that would enable the robot to learn without any external intervention. Further, we want the robot to learn qualitative models so that its insights would be easily comprehensible to humans. The robot is restricted to learn by experimentation and to use self generated models for moving and designing new experiments. The robot's motivation for learning is a part of the built-in algorithm. The idea is simple - since the robot depends on its own model, the robot wants to optimize the model's prediction accuracy. To improve the model in time requires collecting new observation data, particularly data most useful for the improvement of the model. Hopefully, if all goes well, after some time the robot will come up with a model whose predictions are always correct, i.e. the robot has learned everything about its actions and their effects in the given environment.

Experimental domain

Our problem domain consists of a mobile robot, a ball and an overview camera, as shown in Fig. 1. The robot uses the overview camera to observe its distance to the ball (ball distance, denoted by bd) and the angle between its orientation and the ball (ball angle, denoted by ba).

The robot is of differential drive type and moves by setting the speeds of the left and the right wheel (L and R respectively). In our case, L and R are always positive, and the robot was restricted to choose between speeds 4 and 5 only. So the robot can move straight ahead ($L = R = 5$), right ($L = 5, R = 4$) and left ($L = 4, R = 5$), as shown in Fig. 2. The robot is not aware of any coordinate system. It

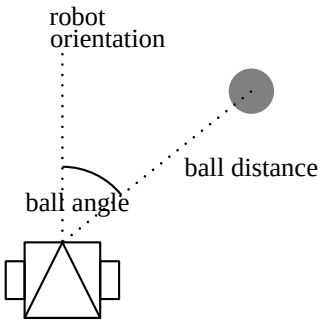


Figure 1: The robot and the ball.

is only aware of the actions it performs (L and R) and the observations from the sensors (bd and ba).

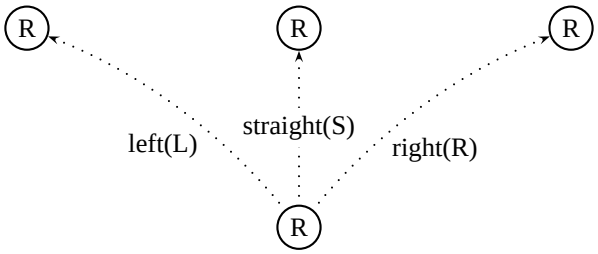


Figure 2: The actions of the robot.

The overall goal that we want the robot to achieve is that it learns a qualitative model describing the relations between its actions and observations. By densely sampling the whole space of above mentioned variables and learning a qualitative tree we obtained an "almost ideal" model of our domain. Note, that we did not use this model in any other way than to see for ourselves what the robot should eventually learn. This "almost ideal" qualitative tree for our domain is shown in Fig. 5. The model is explained in the next section.

We have performed all the experiments in the simulator Simon which is a part of the machine learning framework Orange (Zupan, Leban, & Demšar 2004).

Exploration algorithm

At the begining, i.e. at time t_0, the robot has no knowledge about the effects of its actions. For example, it does not know how its actions from the current state influence its observations in the next step. Namely, there are no relations known to the robot between actions (L and R) and observations (ba and bd).

Without a model the robot can only move by applying random actions, i.e. randomly choose the speed of each wheel. Doing so it collects some data and after a certain time period it learns from the collected data. The learning is supervised. The attributes are robot's actions and observations. The class variable is defined by qualitative relations (described later) between the actions and observations.

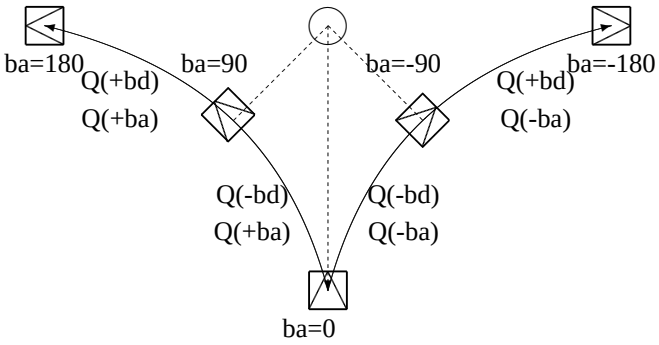


Figure 3: The various angles when robot is turning left and right from $ba = 0$

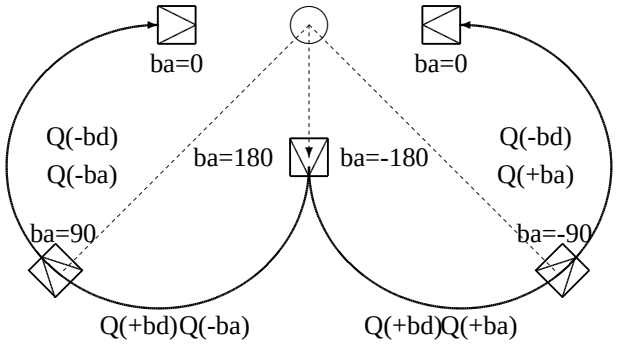


Figure 4: The various angles when robot is turning left and right from $ba = 180$ or $ba = -180$

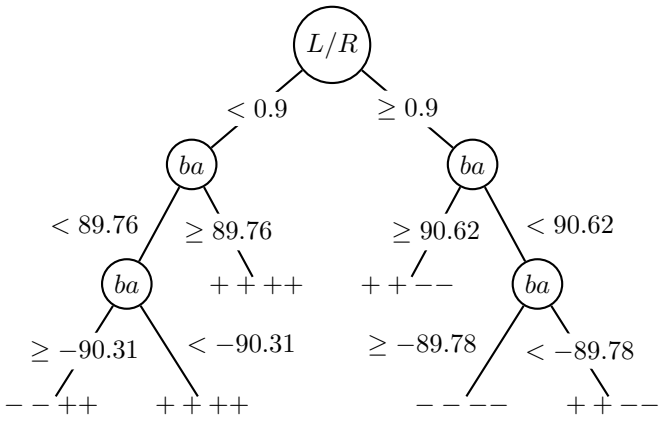


Figure 5: The "almost ideal" model of the robot in our domain.

The robot learns its first model from a small dataset collected by random movement. This initial model is not very accurate and useful. Nevertheless, it enables the robot to use it for making predictions about further actions.

The ability to make predictions enables the choice among learning strategies. A learning strategy determines the next action. The most primitive learning strategy is random strategy, in which the robot chooses one of its three possible actions at random. Random movement is thus defined by actions rather than by robot's positions. The latter is not even possible since in our case the robot is not aware of its coordinates and can not choose to navigate in any coordinate system.

The robot is supposed to learn the relations between its actions and observations. In our simple example, the robot has two actions (L and R) and two observation variables (ba and bd), so it should learn $bd = Q(sS_L)$, $bd = Q(sS_R)$, $ba = Q(sS_L)$ and $ba = Q(sS_R)$, where sign s is $+$ or $-$ and $L = \dot{S}_L, R = \dot{S}_R$, where S_L and S_R are the paths of the left and the right wheel respectively. In these equations, Q stands for qualitative relation as described in section Parametric Padé. In the paper, we use a shorter notation, e.g. "$++--$", giving only the signs s in the above mentioned order. So "$++--$" means: $bd = Q(+S_L)$, $bd = Q(+S_R)$, $ba = Q(-S_L)$ and $ba = Q(-S_R)$. In words: ball distance is increasing when S_L and S_R are increasing (i.e. $L, R > 0$), and ball angle is decreasing when S_L and S_R are increasing. We define the class C of this domain as a 4-tuple of signs as just described. Figures 3 and 4 clearly shows the regions of different values of class C.

Qualitative models that the robot is learning are in the form of qualitative trees (qtree) and qualitative nondeterministic finite automata (envisionment). The robot uses algorithm pPadé with decision trees to learn qualitative trees while it builds an envisionment from the temporal sequence of its actions and observations. The initial set of attributes includes L, R, ba, bd and the class C. To this set, pPadé adds a newly constructed attribute L/R, obtained by the chain rule, dividing the derivatives of each wheel's path w.r.t. time. The attribute L/R describes the qualitative relation between both speeds and can, as we shall see, explain the left and right turns. Using the chain rule for attribute construction is a general principle and is not specifically added to this domain.

There is no relation between the qualitative tree and the envisionment. They are merely a different perspective to the same data. While the qualitative tree is used for prediction, the envisionment is used for planning new experiments for the robot to explore new and less explored regions. Similar to a qualitative tree, the envisionment is gradually learned by the robot.

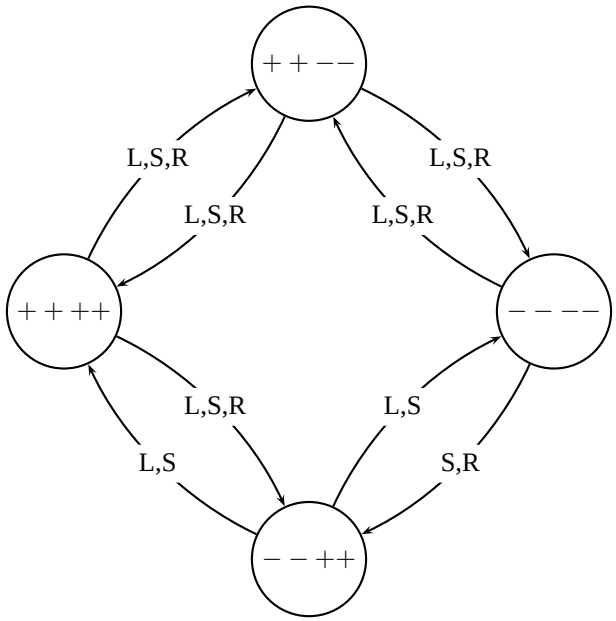


Figure 6: The envisionment learned by the robot.

Exploration strategies

The robot's exploration algorithm includes three strategies that strive to guide the learning towards the final goal. First and most primitive is the *random strategy*. Using this strategy, the robot moves randomly choosing the actions from its set of available actions. The second strategy we call *uniform strategy*; it is used when the robot wants to sample the actions so that their distribution is uniform. Uniform distribution of actions assures that the robot is not biased towards one of the actions, e.g. going straight ahead all the time. At first glance it may seem that uniform and random strategies are the same, but the difference lies in the fact that uniform strategy also accounts for the action executed using persistent strategy. The third strategy is called *persistent strategy*. The robot, using this strategy, keeps executing the same action for some time. Doing so it is collecting more learning examples of the same kind.

The robot uses random strategy only for its first ten moves when it has no knowledge about its environment and the random choice is the best it can make. After it collects the first ten learning examples it can already build a first model and

start using it. At this time, it changes the strategy to *uniform* and enters the main loop in which it is updating and improving the model.

The main loop starts with choosing the next action based on the current strategy (either uniform or persistent). After the robot picks the action it uses the current qualitative tree to make the prediction using the current state and the action. When it makes the prediction it executes the action and observes the result. It compares its own prediction with the actual observation. If they match, the robot continues with persistent strategy, otherwise the robot is "surprised" and motivated for further exploration of the unknown behaviors. The reason for the mismatch is the false prediction of qualitative behavior, i.e. the signs in the class value were predicted wrongly. The robot updates the envisionment with a new state and transition and also updates the qualitative tree. After it updates the model, the robot starts designing a new experiment and planning its actions so that it could carry out the designed experiment. For this purpose it maintains a frequency table of class values and it observes the difference between the number of examples in the current envisionment state and the one with the lowest frequency in the table. If the number of examples in the current envisionment state is greater than a threshold, it selects uniform strategy and picks persistent otherwise. This finishes one iteration of the loop and starts a new one.

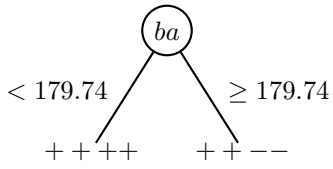


Figure 7: The model created by the robot after 19 steps.

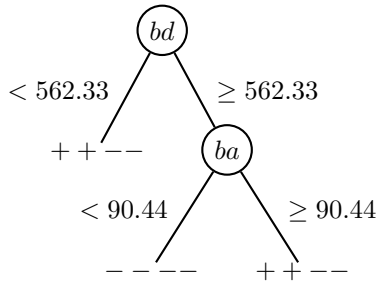


Figure 8: The model created by the robot after 1000 steps.

Results

The exploration algorithm from the previous section enables the robot to learn by experimentation in an efficient way. To confirm the latter, in this section we compare our approach to the pure random strategy. Again, we stress that random strategy does not mean random sampling of the coordinate space but rather choosing the actions randomly.

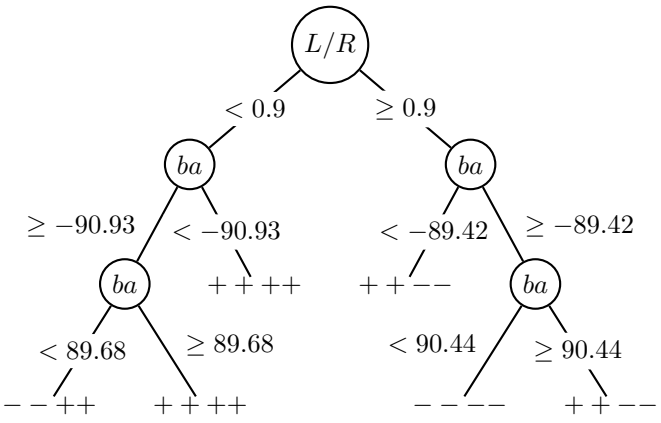


Figure 9: The final model created by the robot after 2674 steps.

In random strategy we use a parameter *duration* which defines the frequency for choosing a random action. If *duration* $= 1$ the action is chosen randomly on each simulation step while for *duration* $= n$ it is chosen only each n-th step and maintained the same in between. The latter is actually not a pure random strategy but rather a mixture of random and persistent. We use it for comparison anyway since the pure random strategy performs extremely poor.

We ran 3 runs of each random strategy, varying *duration* and 9 runs with different initial positions of the robot with our exploration algorithm. We manually determined the point at which the robot learned the desired model. We measured the time it had needed to learn the model in the number of steps it performed until that state. Table 1 presents the results over different runs, the averages and standard errors.

The results show that the robot learns significantly better and faster with our exploration algorithm as opposed to the pure random strategy or random-persistent strategies. We have no formal proof to explain why persistent strategy works. Nevertheless, it is clear from the way humans experiment that we pursue one direction until there arises a reason or motivation to change it.

Parametric Padé

Algorithm Padé, as described in (Žabkar, Bratko, & Demšar 2007), discovers monotonic relations in static domains. It does so by computing partial derivatives from numerical data and is used together with an appropriate machine learning algorithm, e.g. decision trees, to build a qualitative model. However, it is quite limited in the diversity of the domain types it can handle. For example, it can not handle a temporal data set well. Here, we introduce a motivation for a complementary method which we call parametric Padé, abbreviated pPadé. The parameter in pPadé is time which allows pPadé to learn in dynamic domains. We should note here that Padé also works with other parametrizations but time. We only give here time as an example.

Time is not always an important attribute. For example, it is always true that "the larger the piece of ice, the heavier

	Random		our exploration strategy	
Run	Stepsize	Steps taken to reach best model	Stepsize	Steps taken to reach best model
1	1	Not until 30000	1	2674
2		Not until 30000	1	3685
3		Not until 30000	1	1991
4	10	Not until 30000	1	2078
5		Not until 30000	1	3530
6		Not until 30000	1	3254
7	100	15967[a]	1	7317
8		Not until 30000	1	4866
9		27654[b]	1	2843

[a] Even this does not result in the ideal model, but very close to it

[b] This resulted in a model separated at the root by L instead of L/R

Table 1: Comparison between random action selection and our exploration strategy presented here.

it is". In Padé's notation this qualitative proportionality is written as *weight = Q(+volume)*. However, a lot of things change over time and for these time obviously is important. Yet, it should not be treated as any other attribute but rather as a parameter, i.e. the temporal dimension is somehow hidden. For example, it is well known that parametric equations $x(t) = \cos(t)$, $y(t) = \sin(t)$ represent a unit circle for $t \in [0, 2\pi]$. While we observe the circle in xy-plane, parameter t remains hidden. Derivatives w.r.t. time can take advantage of the chain rule:

$$\frac{dy}{dx} = \frac{dy}{dt}\frac{dt}{dx} = \frac{\dot{y}}{\dot{x}}$$

In temporal data sets, it is possible to compute the derivatives of the attributes w.r.t. time t and by using the chain rule, obtain the derivatives w.r.t. other variables as well. Doing so, we overcome the problem of high dimensionality. As opposed to ordinary Padé, where the derivatives are computed in the space of dimensionality n ($n =$ number of attributes), all the derivatives in parametric Padé are computed w.r.t. time.

The input for pPadé is a temporal data set, e.g. a set of points in xy-space (Fig. 10(a)) each having a time stamp. The first four columns of Table 2 present the example data set which we use to illustrate how pPadé works.

The goal in this toy example is to obtain the qualitative behavior of the class variable c w.r.t. attribute x.

The temporal diagram of attributes x and y is shown in Fig. 10(b). First, pPadé computes $\dot{x}$, $\dot{y}$ and $\dot{c}$. pPadé approximates the derivative $\dot{x}$ at t_i as:

$$\dot{x}_i = \frac{x_{i+1} - x_i}{t_{i+1} - t_i}.$$

Simple divided differences can be substituted with more robust, noise resistant linear regression, localy weighted regression (LWR) (Atkeson, Moore, & Schaal 1997) or LOESS (Cleveland 1979; Cleveland & Devlin 1988). However, a machine learning algorithm that is subsequently applied to these approximations also tends to eliminate noise.

pPadé uses the chain rule to compute dc/dx as $\dot{c}/\dot{x}$ and similarly of dc/dy. Table 2 presents the computed derivatives and qualitative behavior of c w.r.t. attributes x and y. The signs of dc/dx are also shown in Fig. 10(c). On the other hand, Figure 10(d) shows why it is not possible to correctly asses the desired derivatives in xy-plane, namely the points' neighbors do not respect the time but rather the Euclidean distance in the plane alone.

Related work

The problem we tackled in this paper is addressed in many different research fields which include but are not limited to robotics, AI, psychology and cognitive sciences. We only mention those that are directly related to model building.

Similar to our approach, (Modayil & Kuipers 2007) present an algorithm for learning qualitative models from robot's actions and observations, but their qualitative models are in the form of object control laws while we use qualitative trees and envisionment. An interesting approach using probability estimates is described in (Hart, Grupen, & Jensen 2005). Work by (Barto, Singh, & Chentanez 2004) in intrinsically motivated learning shows how reusable rules can be learned, but only in a playroom domain with much more data than we require. (Kuipers *et al.* 2006) describes a methodology that bootstraps knowledge from low-level sensorimotor primitives and then uses this knowledge to navigate in its environment. (Stoytchev 2005) proposes a novel approach to representing and learning tool affordances by a robot by pushing objects, but with very limited and specific exploratory behaviors.

Conclusion and further work

We showed a simple example of a robot that is capable of learning by making experiments in its environment. The exploration algorithm that we presented proved to be a useful tool for the autonomous learner that has to design, plan and execute the experiments in order to obtain some knowledge about how its actions influence its observations in the given world. One of the contributions in our opinion is the use of qualitative models only and the combination of qualitative tree and the envisionment. Both models do not only suffice to support the robot in its actions, but also offer insights into the knowledge that the robot acquired in the learning

process. Further, we believe that our approach can be generalized to other more complex domains and that it can scale well due to the simplicity of learning the qualitative models.

The algorithm for autonomous learning can be further improved by elaborating the planning part and the design of experiments. Applying this procedure in other domains and with real robots may give rise to new ideas for further development. We are already very close to running a real robot with this algorithm.

Acknowledgment

The work described in this article has been funded by the European Commission's Sixth Framework Programme under contract no. 029427 as part of the Specific Targeted Research Project XPERO ("Robotic Learning by Experimentation").

References

Atkeson, C.; Moore, A.; and Schaal, S. 1997. Locally weighted learning. *Artificial Intelligence Review* 11:11–73.

Barto, A. G.; Singh, S.; and Chentanez, N. 2004. Intrinsically motivated learning of hierarchical collections of skills. *International Conference on Developmental Learning*.

Cleveland, W., and Devlin, S. 1988. Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association* 83:596–610.

Cleveland, W. 1979. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association* 74:829–836.

Hart, S.; Grupen, R.; and Jensen, D. 2005. A relational representation for procedural task knowledge. In *Proc. 20th National Conf. on Artificial Intelligence*.

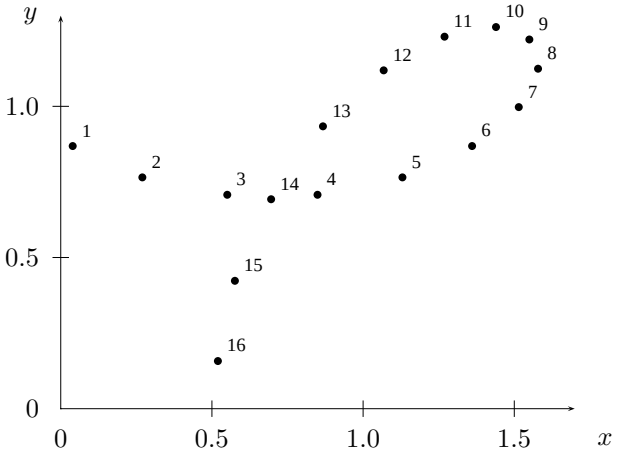
Kuipers, B.; Beeson, P.; Modayil, J.; and Provost, J. 2006. Bootstrap learning of foundational representations.

Modayil, J., and Kuipers, B. 2007. Where do actions come from? autonomous robot learning of objects and actions. *AAAI Spring Symposium Series 2007, Control Mechanisms for Spatial Knowledge Processing in Cognitive / Intelligent Systems*.

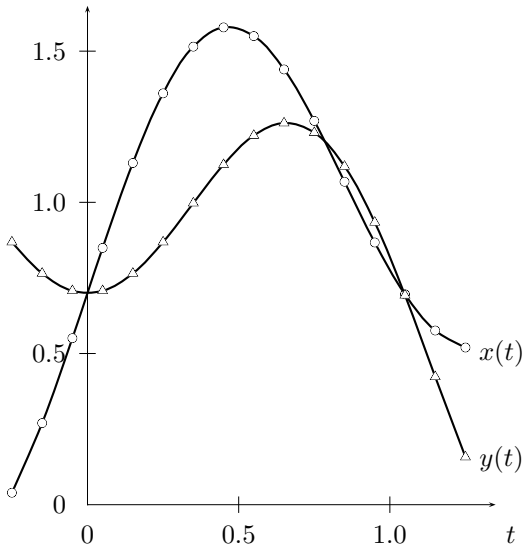
Stoytchev, A. 2005. Behavior-grounded representation of tool affordances. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Žabkar, J.; Bratko, I.; and Demšar, J. 2007. Learning qualitative models through partial derivatives by pad. In *Proceedings of the 21th International Workshop on Qualitative Reasoning*.

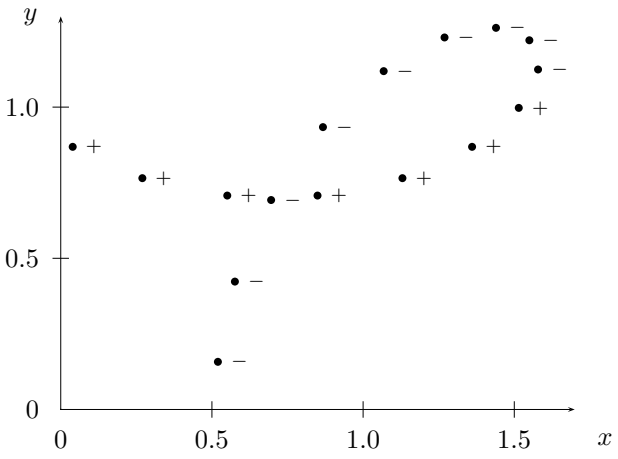
Zupan, B.; Leban, G.; and Demšar, J. 2004. Orange: Widgets and visual programming, a white paper.



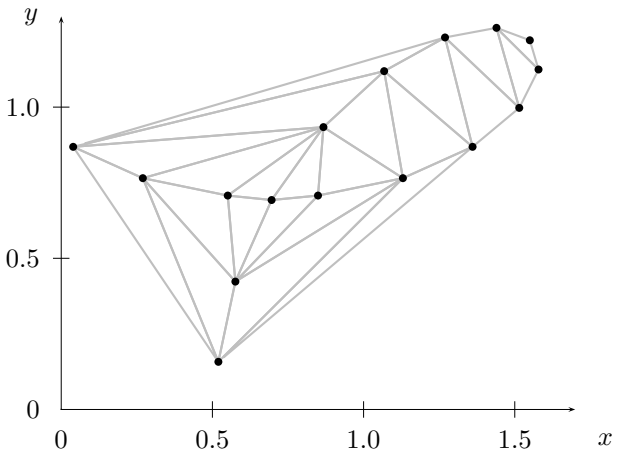
(a) Sampled parametric curve: $x(t) = \sin(3t)\cos(t) + .7$, $y(t) = \sin(3t)\sin(t) + .7$, class variable c takes the values at specified points from $1, \ldots, 16$.



(b) Attributes x and y as functions of t.



(c) Derivatives' signs of dc/dx for each point considering time, calculated using parametric Padé.



(d) Delaunay triangulation of the attribute space. Each point's neighbors could be defined by connections in this graph. So defined neighbors are not good for calculating the derivatives because real neighbors are implied by time.

Figure 10: Illustration of parametric Padé.

t	x	y	c	$\dot{x}$	$\dot{y}$	$\dot{c}$	$\dot{c}/\dot{x}$	$\dot{c}/\dot{y}$	$\dot{y}/\dot{x}$	$c = Q(x)$	$c = Q(y)$
-0.25	0.039	0.868	1	2.30	-1.03	10	4.34	-9.64	-0.44	+	-
-0.15	0.269	0.765	2	2.80	-0.57	10	3.56	-17.38	-0.20	+	-
-0.05	0.550	0.707	3	2.98	0	10	3.35	∞	0	+	o
0.05	0.849	0.707	4	2.80	0.57	10	3.56	17.38	0.20	+	+
0.15	1.130	0.765	5	2.30	1.03	10	4.34	9.64	0.44	+	+
0.25	1.360	0.868	6	1.54	1.28	10	6.47	7.76	0.83	+	+
0.35	1.514	0.997	7	0.63	1.26	10	15.68	7.87	1.99	+	+
0.45	1.578	1.124	8	-0.28	0.96	10	-34.79	10.34	-3.36	-	+
0.55	1.549	1.221	9	-1.10	0.41	10	-9.06	24.30	-0.37	-	+
0.65	1.439	1.262	10	-1.70	-0.31	10	-5.87	-31.41	0.18	-	-
0.75	1.269	1.230	11	-2.01	-1.11	10	-4.96	-8.97	0.55	-	-
0.85	1.068	1.118	12	-2.00	-1.85	10	-4.97	-5.40	0.92	-	-
0.95	0.867	0.933	13	-1.71	-2.41	10	-5.83	-4.14	1.40	-	-
1.05	0.695	0.692	14	-1.19	-2.69	10	-8.34	-3.70	2.25	-	-
1.15	0.576	0.422	15	-0.56	-2.65	10	-17.78	-3.76	4.71	-	-
1.25	0.519	0.157	16	-0.56	-2.65	10	-17.78	-3.76	4.71	-	-

Table 2: The input data (columns 1-4) and the output of parametric Padé (columns 5-12).