*Proceedings Fifteenth National Conference on Artificial Intelligence 1998 (AAAI-98), Madison, Wisconsin, July 1998.*

# Multimodal Reasoning for Automatic Model Construction

**Reinhard Stolle and Elizabeth Bradley***†
University of Colorado at Boulder
Department of Computer Science
Boulder, Colorado 80309–0430
{stolle,lizb}@cs.colorado.edu

### Abstract

This paper describes a program called PRET that automates system identification, the process of finding a dynamical model of a black-box system. PRET performs both structural identification and parameter estimation by integrating several reasoning modes: qualitative reasoning, qualitative simulation, numerical simulation, geometric reasoning, constraint reasoning, resolution, reasoning with abstraction levels, declarative meta-level control, and a simple form of truth maintenance.

Unlike other modeling programs that map structural or functional descriptions to model fragments, PRET combines hypotheses about the mathematics involved into candidate models that are intelligently tested against observations about the target system.

We give two examples of system identification tasks that this automated modeling tool has successfully performed. The first, a simple linear system, was chosen because it facilitates a brief and clear presentation of PRET's features and reasoning techniques. In the second example, a difficult real-world modeling task, we show how PRET models a radio-controlled car used in the University of British Columbia's soccer-playing robot project.

## Introduction

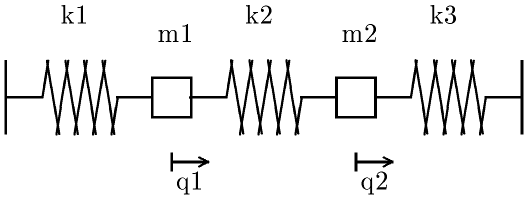
Models are powerful tools that are used to understand physical systems. The abstraction level of a model is an essential part of its power as a reasoning aid. Abstract models are simple: they account for major properties of the physical system. Less-abstract models are more complicated, allowing them to capture the features of the physical system more accurately and in more detail, but at the cost of increased complexity during model construction and usage. Typically, in this abstraction hierarchy, the model of choice is the one that is just detailed enough to account for the properties and perspectives that are of interest for the task at hand.

The process of inferring an internal model of a system's dynamics from external observations—often called *system identification*—is a routine and difficult problem faced by engineers in a variety of domains (Aström & Eykhoff 1971; Ljung 1987). The first stage of the system identification process, *structural identification*, identifies the form of the model, or skeleton of the equation, such as $a\ddot{\theta} + b\sin\theta = 0$ for a simple pendulum. In the second system identification stage, *parameter estimation*, the parameter values $a$ and $b$ are determined.

The program PRET (Bradley & Stolle 1996) automates both stages of the system identification process; it finds a system of ODEs that models a given physical system. Inputs are *observations* about that system, user-supplied *hypotheses* about the desired model, and *specifications*. Fig. 1 shows a physical system that consists of two masses and three springs and the call that instructs PRET to construct a model of that system. It is important to note that this simple linear example does not by any means exercise PRET's power.

Observations are measured automatically by sensors and/or interpreted by the user; they may be symbolic or numeric and take on a variety of formats and degrees of precision. For example, an observation might inform PRET that the system to be modeled is autonomous; another observation could state that the state variable $q$ oscillates and that this oscillation is damped. Observations can also be physical measurements made directly and automatically on the system (Bradley & Easley 1998). Hypotheses about the physics involved, e.g., a hypothesis about friction, are supplied to PRET by the user; these may conflict and need not be mutually exclusive, whereas observations are always held to be true. Finally, specifications indicate the quantities

* Supported by NSF NYI #CCR-9357740, ONR #N00014-96-1-0720, and a Packard Fellowship in Science and Engineering from the David and Lucile Packard Foundation.




```
(find-model
  (domain mechanics)
  (state-variables <q1> <q2>)
  (point-coordinates <q1> <q2>)
  (hypotheses
    (<force> (* m1 (deriv (deriv <q1>))))
    (<force> (* m2 (deriv (deriv <q2>))))
    (<force> (* k1 <q1>))
    (<force> (* k2 (- <q1> <q2>)))
    (<force> (* k3 <q2>))
    (<force> (* r1 (deriv <q1>)))
    (<force> (* r2 (square (deriv <q1>))))
    (<force> (* r3 (deriv <q2>)))
    (<force> (* r4 (square (deriv <q2>)))))
  (observations
    (autonomous)
    (oscillation <q1>)
    (oscillation <q2>)
    (numeric (<time> <q1> <q2>)
             ((0 .1 .1) (.1 .1099 .1103) ...  )))
  (specifications
    (resolution <q1> absolute 1e-3 (0 1))
    (resolution <q2> absolute 1e-3 (0 1))))
```

Figure 1: A `find-model` call that instructs Pret to model the `Springs&Masses` system shown above. In this example, the user first sets up the problem, then hypothesizes nine different force terms, makes four observations about the position coordinates $q_1$ and $q_2$, and finally specifies resolutions and ranges.

of interest and their resolutions— for example, a specification might impose a microsecond resolution over 120 seconds of system evolution.

When modeling physical systems, human engineers make use of a variety of well-established modeling techniques. A modeler's reasoning about a given physical system and possible candidate models takes place at an abstract level first and resorts to more detailed reasoning later in the modeling process. Pret's goal is to mimic this strategy and attempt to find the right model at the right abstraction level as quickly as possible. Therefore, the challenge in designing Pret was to work out a formalism that meets two requirements: first, it must facilitate easy formulation of the various reasoning techniques; and second, it must allow Pret to reason about which techniques are appropriate in which situation. This paper describes Pret's reasoning modes, how they are integrated, and how they solve these problems.

The next section gives a brief overview of Pret. In the following section, we present the various reasoning modes, their interaction, and their integration, and give two examples that illustrate how Pret works. Finally, we give an evaluation, point to problems and future directions, and discuss related work.

## Overview of Pret

Pret incorporates two types of knowledge:

- *Domain rules* that apply in individual domains, such as Kirchhoff's voltage law for electronic circuits, and
- *ODE rules* that capture mathematical precepts that are true for all ODEs, such as the definition of an equilibrium point.

The high-level control flow of Pret is a variant of "generate-and-test." In the "generate" phase, the domain rules are used to construct candidate models from the user's hypotheses. Since Pret tries to find a model that accounts for all observations without being more complex than necessary, candidate models are generated in order of increasing complexity. In the "test" phase, Pret uses its ODE rules to generate new knowledge about the physical system and about the current candidate model. A model is valid if the known facts about the system to be modeled are consistent with the known facts about the model. Any inconsistency is a reason to discard the model. Currently, the first model in this generate-and-test sequence that is consistent with the observations is returned as the result.

Pret judges models according to the opportunistic paradigm "valid if not proven invalid:" if a model is bad, there must be a reason for it. Or, conversely, if there is no reason to discard a model, it is a valid model. Therefore, Pret's central task is to quickly find inconsistencies between a candidate model and the target system. The remainder of the paper focuses on the techniques that Pret uses to accomplish this task.

## Reasoning Modes

In order to find inconsistencies between candidate models and the target system quickly, Pret employs a number of very different techniques. None of these techniques is new; human engineers routinely use them when modeling dynamical systems, and every one has been used in at least one automated modeling tool (see the Related Work section). The particular set of techniques used here, the multimodal reasoning framework that integrates them, and the system architecture that lets Pret decide which one is appropriate in which situation, make the approach taken here different—and powerful. This particular combination of tactics—designed for and focused upon a particular problem domain—constitutes the novel contributions of the work described in this paper.

Using the paradigm "valid if not proven invalid," it is important to rule out invalid candidate models as quickly as possible. To accomplish this, PRET uses abstract knowledge first and detailed knowledge later. In order to achieve this behavior, PRET chooses among, invokes, and interprets the results of several reasoning modes. In the following subsections, we describe these reasoning modes and show how PRET orchestrates their selection, invocation, and interactions.

### Qualitative Reasoning

Reasoning about abstract features of the physical system or the candidate model is typically faster than reasoning about their detailed properties. Therefore, PRET uses a "high-level first" strategy: it tries to rule out a model by purely *qualitative* techniques before advancing to more expensive semi-numerical or numerical techniques. Often, only a few steps of inexpensive qualitative reasoning (QR) suffice to quickly discard a model. Some of these qualitative rules in turn make use of other tools, e.g., symbolic algebra facilities from the commercial package Maple. For example, PRET's ODE theory includes the qualitative rule that oscillating linear systems must be of at least second order. If the user's observations imply an oscillation, any model whose order is less than two can be discarded without performing more-complex operations such as, for example, a numerical integration of the model.

PRET's QR features are not only important for accelerating the search for inconsistencies between the physical system and the model; they also allow the user to express incomplete information (Kuipers 1992). For example, the user might not know the exact value of a friction coefficient, but he or she might know that it is constant and positive. This example also shows the importance and power of interaction among the various reasoning modes. A qualitative result about the sign of a parameter can be used by the constraint reasoner, and vice versa. PRET's logic inference system is designed to facilitate exactly this type of interaction (see Section "Storing and Reusing Intermediate Results").

### Geometric Reasoning

Verification of a *valid* candidate model with respect to a numeric observation requires point-by-point comparison with a numeric integration of the ODE (see the section entitled "Parameter Estimation and Numerical Simulation"). However, *invalid* models can often be ruled out without such an expensive numeric procedure; many inappropriate models, for instance, can be discarded by reasoning about purely qualitative information that can be inferred inexpensively from numeric information. In order to achieve this behavior, PRET processes the observations—curve fitting, recognition of linear regions, and so on—using Maple functions and simple phase-portrait analysis techniques, both of which yield high-level results that can then be used much as qualitative observations are (Bradley & Easley 1998).

### Qualitative Simulation

Before PRET resorts to the numerical level, it attempts to establish contradictions quickly and cheaply by reasoning about the qualitative states of the physical system (Kuipers 1992). PRET's qualitative envisioning module constrains the possible ranges of parameters in the candidate model. If the constraints become inconsistent—i.e., the range of a parameter becomes the empty set—the model is ruled out. Currently, the qualitative states contain only sign information $(-, 0, +)$. For example, for the model $ax + by = 0$ the state $(x, y) = (+, +)$ constrains $(a, b)$ to the possibilities $(+, -)$ or $(0, 0)$ or $(-, +)$.

PRET does not do full qualitative simulation (Kuipers 1986). Instead, it only envisions the state space of all possible combinations of qualitative values of state variables and parameters. This strategy is faster than full qualitative simulation, but it is also less accurate; it may let invalid models pass the test. These invalid models are later ruled out by the numeric simulator. However, for the models that do fail the qualitative envisioning test, this test is much cheaper than a numeric simulation and point-by-point comparison would be.

### Parameter Estimation and Numerical Simulation

If the observations contain a numerical time series, the model must match the time series to within the resolution specified in the `find-model` call. If a candidate model cannot be discarded by qualitative means, PRET integrates the model (an ODE system) with fourth-order Runge-Kutta, comparing the result to the numeric time-series observation. Typically, however, models contain parameters whose values must be determined before the numeric integration can take place. For example, for the model $a\ddot{\theta} + b \sin\theta = 0$ of a simple pendulum the parameter values $a$ and $b$ must be determined. Parameter estimation for nonlinear systems is a (very difficult) global optimization task; PRET solves this problem by combining qualitative reasoning and local numerical methods (an orthogonal distance regression-based nonlinear least-squares solver, specifically). The nonlinear parameter estimation reasoner (NPER) that implements these ideas is described in (Bradley, O'Gallagher, & Rogers 1997).[1] The NPER uses knowledge derived during the structural identification phase to guide the parameter estimation process—for example, using constraint reasoning (e.g., the sign of a friction coefficient) or the results of the qualitative envisionment to choose good initial values, thereby avoiding local minima in the regression landscape.

## Constraint Reasoning

Often, information *between* the purely qualitative and the purely numeric levels is also available. If a linear system oscillates, for example, the imaginary parts of at least one pair of the roots of its model's characteristic polynomial must be nonzero. Thus, if the model $a\ddot{x} + b\dot{x} + cx = 0$ is to match an oscillation observation, the coefficients must satisfy the inequality $4ac > b^2$. PRET uses expression inference to merge and simplify such *constraints*. However, this approach works only for linear and quadratic expressions and some special cases of higher order. We are investigating techniques for reasoning about more-general expressions, e.g., (Faltings & Gelle 1997). For example, if the candidate model $\ddot{x} + a\dot{x}^4 + b\dot{x}^2 = 0$ is to match a `conservative` observation, the coefficients $a$ and $b$ must take on values such that the divergence $-4a\dot{x}^3 - 2b\dot{x}$ is zero, below a certain resolution threshold, for the specified range of interest of $x$.

## SLD-based resolution

PRET's search for an inconsistency between the observations and a particular candidate model is based on SLD resolution. The language in which observations and the ODE theory are expressed is that of generalized Horn clause intuitionistic logic (GHCIL) (McCarty 1988). Roughly, GHCIL clauses are Horn clauses that also allow embedded implications in their bodies.

The special atomic formula `falsum` may only appear as the head of a clause. Such clauses express fundamental reasons for inconsistencies, e.g., that a system cannot be oscillating *and* non-oscillating at the same time. This concept of *negation as inconsistency* (Gabbay & Sergot 1986) is the only form of negation in our paradigm. Negation as failure, for example, which is the standard form of negation in PROLOG, is particularly undesirable for our purposes. Since we do not require the user to supply all possible[2] observations, the absence of knowledge cannot be used to generate new knowledge.

A model is ruled out if and only if a contradiction exists between a mathematical property of the physical system (e.g., a sensor observation that includes an oscillation `(oscillation <x>)`) and a mathematical property of the model (e.g., `(no-oscillation <x>)`, derived from a root-locus analysis of an ODE model). For every candidate model, PRET combines basic facts about the target system, basic facts about the candidate model, and basic facts and rules from the ODE theory into one set of clauses, and then checks that set for inconsistency, i.e., tries to derive `falsum` from it.

The basic facts about the target system are the observations. They have two potential sources—the user and the sensors—and may be *descriptive*, *graphical*, or *numeric*. The former use special descriptive keywords, the second are sketches drawn on a computer screen with a mouse, and the third simply specify data points. Only the first and the last are currently implemented. The basic facts about the current model are obtained by a collection of SCHEME[3] "model observer" functions that identify mathematical properties of the model, e.g., "the ODE is linear in $x$." These functions essentially implement the basic operations found in any differential equations text.

In the course of trying to prove the `falsum`, PRET's inference engine expands both sets of properties by applying the ODE rules; for example, if the system is known to be autonomous, its model cannot explicitly contain the variable `<time>`. This is one of the advantages of the declarative reasoning framework: an expert user of PRET can easily extend and specialize the ODE theory represented in the knowledge base.

The special predicate `scheme-eval` provides the link between the inference engine and PRET's model observer functions. It also provides the link to all modules that implement other reasoning modes.

For a more detailed discussion of PRET's logic system, see (Stolle & Bradley 1996).

## Declarative Meta Level Control

The *control strategy* of a SLD resolution theorem prover is defined by the function that selects the literal that is resolved and by the function that chooses the resolving clause. PRET provides meta-level language constructs that allow the implementer of the ODE theory to specify the *control strategy* that is to be used (Davis 1980; Gallaire & Lasserre 1982; Beckstein, Stolle, & Tobermann 1996). The intuition here is, again, that the search should be guided towards a cheap and quick proof of a contradiction. As an example, consider the following (simplified) excerpt

[1] How to use qualitative reasoning (QR) to navigate in the parameter space for models of one particular class of *linear* systems is exemplified in (Capelo, Ironi, & Tentoni 1996).

[2] Here, *possible* means *expressible with the implemented observation vocabulary.*

[3] PRET is written in SCHEME.

from the knowledge base.

$$\begin{array}{rcl} \textit{stable} & \leftarrow & \textit{linear}, \textit{all\_roots\_in\_left\_half\_plane}. \\ \textit{stable} & \leftarrow & \textit{non\_linear}, \textit{stable\_in\_all\_basins}. \\ \textit{hot}(L) & \leftarrow & \textit{linear}, \textit{goal}(L, \textit{stable}). \end{array}$$

A linear dynamical system has a unique equilibrium point, and the stability of that point—and therefore of the system as a whole—can be determined by examining the system's eigenvalues: a simple symbolic manipulation of the coefficients of the equation. *Nonlinear* systems can have arbitrary numbers of equilibrium sets, which are expensive to find and evaluate. Thus, if a system is known to be linear, its *overall* stability is easy to establish, whereas evaluating the stability of a nonlinear system is far more complicated and expensive. Therefore, PRET's meta control predicate `hot` is used in this example to prioritize a stability check in the case where this check is cheap and easy—i.e., if the system is known to be linear.

For a more detailed discussion of PRET's meta control constructs, see (Hogan, Stolle, & Bradley 1998).

### Reasoning at Different Abstraction Levels

Every rule in the ODE theory is assigned a natural number that indicates its level of abstraction: the lower the *abstraction level number*, the more abstract the rule. Whereas the meta predicates described in the previous paragraph specify *dynamic* control, the ODE rule abstraction levels express *static* control information. The theorem prover proceeds to a higher abstraction level number only if the attempt to prove the `falsum` with ODE rules with lower abstraction level numbers fails. For example, the `scheme-eval`-rule that triggers numerical integration has a higher abstraction level number than the `scheme-eval`-rule that calls the qualitative simulation.

PRET tries to build complete proofs on a more-abstract level before even considering less-abstract rules. Since abstract reasoning usually involves less detail, this approach leads to short and quick proofs of the `falsum` whenever possible.

### Storing and Reusing Intermediate Results

PRET reuses previously derived knowledge in three ways. First, knowledge about the physical system is global, whereas knowledge about a candidate model is local to that model. Therefore, PRET reuses knowledge that is independent of the current candidate model.

Second, knowledge is reused within the process of reasoning about one particular model. Every time the reasoning proceeds to a less-abstract level, PRET needs all information that has already been derived at the more abstract level. To avoid duplication of effort, PRET stores this information rather than rederiving it. The user declares a number of predicates as *relevant* (Beckstein & Tobermann 1992) which causes all succeeding subgoals with this predicate to be stored for later reuse.[4] Currently, PRET recognizes special cases and generalizations of previously proved formulae, but it maintains no contexts or labels for intermediate results.

Finally, other—non-logic-based—reasoning modes typically use knowledge that has been generated by previous inferences, which may in turn have triggered other reasoning modules. To facilitate this, PRET gives these modules access to the set of formulae that have been derived so far. As described above, for example, the nonlinear parameter estimation reasoner (NPER) uses knowledge derived in the structural identification phase.

## Example Applications

In this section we trace PRET's actions on the example of Fig. 1. The four friction forces are omitted here for space reasons, and the numeric observation has the form `(numeric (<time> <q1> <q2>) (eval *data*))`. The keyword `eval` causes the variable `*data*` to be evaluated in the calling environment. Bound to this variable is a time series that was generated by Runge-Kutta integration of the system

$$\begin{array}{rcl} \ddot{q}_1 & = & -0.1\, q_1 - 0.2\,(q_1 - q_2) \\ \ddot{q}_2 & = & 0.2\,(q_1 - q_2) - 0.3\, q_2. \end{array}$$

The first candidate model is $k_1 q_1 = 0$. A SCHEME function called on the ODE establishes the fact `(order <q1> 0)` which expresses that the order of the highest derivative of $q_1$ in this model is zero. This fact conflicts with facts inferred from the observation `(oscillation <q1>)`, so this model is ruled out. The way PRET handles this first candidate model demonstrates the power of its abstract-reasoning-first approach; only a few steps of inexpensive qualitative reasoning suffice to let it quickly discard the model.

PRET tries all combinations of `<force>` hypotheses at single point coordinates, but all these models are ruled out for qualitative or numeric reasons. It then proceeds with ODE systems that consist of *two* force balances—one for each point coordinate. One example of a candidate model of this type is

$$\begin{array}{rcl} k_1 q_1 + m_1 \ddot{q}_1 & = & 0 \\ m_2 \ddot{q}_2 & = & 0 \end{array}$$

[4] The set of previously derived relevant formulae is currently implemented as a hash table.

None of the implemented rules discards this model by purely qualitative means, so PRET invokes its nonlinear parameter estimation reasoner (NPER) which finds no appropriate values for the coefficients $k_1$, $m_1$, and $m_2$ such that any ODE solution matches the numeric time series. Therefore, this candidate model is also ruled out.[5] After having discarded a variety of candidate models in a similar manner, PRET tries the model

$$\begin{aligned} k_1 q_1 + k_2(q_1 - q_2) + m_1 \ddot{q}_1 &= 0 \\ k_3 q_2 + k_2(q_1 - q_2) + m_2 \ddot{q}_2 &= 0 \end{aligned}$$

Again, it calls the NPER, this time successfully. It then substitutes the returned parameter values for the constants $k_1$, $k_2$, $k_3$, $m_1$, and $m_2$ and integrates the resulting ODE system with fourth-order Runge-Kutta, comparing the result to the numeric time-series observation. The difference between the integration and the observation stays within the specified resolution, so the numeric comparison yields no contradiction and this candidate model and the parameter values are returned as the answer.

We have chosen the simple spring-mass example of Fig. 1 to make this presentation brief and clear. Linear systems of this type are easy to model; no engineer would use a software tool to do generate-and-test and guided search to find an ODE model of a system so simple and well-understood. This example is representative neither of PRET's power nor of its intended applications—nonlinear, high-dimensional, black-box dynamical systems. Modeling these types of systems is where PRET's mixture of exact and approximate techniques, quantitative and qualitative reasoning, and precise and heuristic knowledge becomes truly powerful.

PRET has also been successfully used to solve a real-world problem: modeling the radio-controlled (R/C) cars used in the University of British Columbia's soccer-playing robot project. These commercially acquired cars cannot be controlled without an accurate ODE model of their dynamics—something that is not part of the specifications sheet. The sensor data consists of the car's position in an $(x, y)$-plane and its heading (orientation) $\theta$.

PRET goes through the structural identification process that we have already seen in the `Springs&Masses` example, using force balances to assemble hypotheses into models, examining hypothesis combinations in order of increasing complexity, and discarding models that are inconsistent with the observations, always taking advantage of its abstract-reasoning-first approach. The result, in this example, is the model:

$$\begin{aligned} \dot{x} &= v \cos\theta \\ \dot{y} &= v \sin\theta \\ \dot{\theta} &= \rho v \\ \dot{v} &= \alpha + \gamma v \end{aligned}$$

where $v$ is the car's velocity, $\theta$ its orientation, $\rho$ the position of the steering wheel, $\alpha$ acceleration, and $\gamma$ friction. Following the structural identification phase, PRET calls the NPER; as in the previous example, qualitative knowledge derived during the earlier phases of the modeling process are used during parameter estimation—for example, symbolic algebra, constraint propagation, and divided differences are used to compute the initial values and bounds that are passed to the local least-squares solver that lies at the heart of the NPER.

PRET's solution to the modeling problem surprised the University of British Columbia analyst. The model did not match his intuition because he had omitted some important information from the specification—including the fact that the car *started from rest*. In order to work around noise in sensor data, PRET's NPER is designed to filter data and adjust boundary conditions; in this case, this reasoning led to a numerically successful ODE model with a negative initial condition for the velocity—a conflict with the expert's implicit mental model of the situation. Some reflection on this discrepancy led the analyst to realize that the system dynamics might include a delay. Thus, the correct `find-model` call for this example should contain an `observation` that the initial velocity was zero and a `hypothesis` that incorporates a delay between the application of force and the acceleration of the car.

We include this anecdote to emphasize that PRET is an engineer's tool, not a scientific discovery system. Its goal is to construct the simplest ODE that accounts for the observations and specifications that are *explicit* in the `find-model` call, not to infer physics that the user left implicit. The interaction between human expert and automated modeling tool was unexpectedly fruitful in this example; not only did PRET construct an accurate model of the target system, but it actually helped the expert identify what was wrong with the observations and model fragments that he suggested. For a more detailed discussion see (Bradley, O'Gallagher, & Rogers 1997).

[5] In certain cases, this approach may result in a departure from the paradigm *valid if not proven invalid*; unless we trust that the parameter estimator *always* finds a set of coefficients if such a set exists, this amounts to *negation as failure*. This is the appropriate decision here because a user who supplies a numeric time series is certainly interested in a numerically accurate ODE model.

## Evaluation and Future Work

PRET's declarative knowledge representation system is well-suited for encoding ODE theory and control knowledge. Mathematical truths about ODEs are naturally expressed as logical implications. Since the person who implements or maintains this knowledge base is an expert in mathematics and/or engineering—not in logic programming—an approach where control knowledge is separated from object level knowledge and represented declaratively is ideal for this application.

The model tester's cheap-first strategy works well on most junior/senior-level engineering textbook examples and even, as described in the previous section, on some research-level problems. Candidate models that do not trigger qualitative envisioning or numeric parameter estimation and simulation are ruled out symbolically in less than a second,[6] even though the SCHEME code is uncompiled and mostly unoptimized. The time needed for qualitative envisionment and parameter estimation ranges between seconds and minutes, and this is inescapable: a human modeler would face the same expenses when resorting to these techniques. PRET's design goal is not to speed up these techniques, but to avoid them whenever possible.

Currently, the model generator (which was not the focus of this paper) is impractical if more than about ten hypotheses are under consideration, since the number of possible models grows exponentially with the number of hypotheses. This problem can even arise if there are only a few hypotheses in the `find-model` call, as PRET resorts to power series expansions in state variables if it runs out of user-provided hypotheses, and such methods can generate dozens or hundreds of ODE fragments in no time at all. We are investigating solutions to this complexity problem that preferentially select promising hypothesis combinations. Hypotheses could be explicitly prioritized by the user, for instance, or PRET could use case-based reasoning and a repertoire of typical hypothesis combinations (e.g., coriolis and centripetal forces usually appear together, and both have recognizable forms). Other existing modeling tools use energy-based approaches and/or bond graphs (Karnopp & Rosenberg 1975) to find structural models (Amsterdam 1992). This approach may be useful in our case: typical bond graphs and bond graph fragments may help select terms from a large number of hypotheses. It is not clear, however, how similarity of system behavior (the *solution* of an ODE) translates to similarity of the involved model *fragments*.

[6] Several algebraic operations call Maple; interaction with this package is realized through file I/O which can, in sum, take longer than a second. In principle, however, SCHEME could communicate with Maple without file I/O, or the algebraic operations could be implemented in SCHEME directly.

Developing a more sophisticated way of navigating in the space of possible models is an important, interesting, and promising task. Since the state of PRET's inference engine is represented explicitly, it is trivial to generate explanations (proof trees) for failures of models. These explanations can then be used to guide the generation of better models. In the literature this approach has been termed *discrepancy-driven refinement* (e.g., (Addanki, Cremonini, & Penberthy 1991)).

More research is also needed concerning the form in which the user provides hypotheses and observations. In other modeling programs, this input is called *scenario description*. Since PRET's intended target systems are high-dimensional black-box systems, a modeling task's "scenario" must be described in mathematical terms—state variables, ODE fragments, etc.—rather than in terms of traditional qualitative physics—U-tubes, capacitors, springs, and so on. In the R/C car modeling task, for example, the formulation of the ODE makes use of two reference frames simultaneously. PRET's actual formulation of the model is much more complicated. Allowing the user to express hypotheses in appropriate reference frames—and transforming smoothly between different reference frames—is a current focus of our research effort. This is a fundamentally difficult problem; learning how to select the appropriate reference frame in which to reason is one of the most important and difficult concepts one learns in a physics course, so it is not surprising that this is a difficult (perhaps even impossible) task to automate well.

## Related Work

The work described in this paper draws upon ideas and techniques from almost a dozen areas of mathematics, engineering, and computer science; citing more than the few most important and/or most closely related publications in each of these areas would yield an excessive bibliography, so this section—particularly the paragraph on the very active area of reasoning about physical systems—is necessarily abridged. For space reasons, references to related work that appear in the body of the paper will not be repeated here.

In general, *modeling* underlies most approaches to reasoning about physical systems. Strictly speaking, every formalization of the properties of a physical system constitutes a *model* of that system. The spectrum of models ranges from those that use a language that is very close to the physics (e.g., QPT/QPE (Forbus 1984)) to models that use a language that is well suited to describe the system mathematically (e.g., ODEs). QSIM (Kuipers 1986) is a qualitative realization of the mathematics end of this spectrum. PRET resides somewhere in the middle. Its inputs are partially expressed in terms of physics and its reasoning uses concepts from physics. However, its output—the model of the physical system that it constructs—is purely mathematical: an ODE. This choice makes PRET both precise and broadly applicable, as all domains of science and engineering use ODE models.[7] It also imposes some constraints—notably continuity—on the types of systems to which PRET can be applied. There has been recent attention to discontinuities in physical system models; see, for example, (Mosterman & Biswas 1996). PRET models the continuous sections between discontinuities.

Most automated modeling programs (Forbus 1984; Addanki, Cremonini, & Penberthy 1991; Falkenhainer & Forbus 1991; Amsterdam 1992; Kuipers 1993; Nayak 1995) use domain knowledge in order to map a structural description of the system to model fragments. PRET, however, uses only general mathematics in its analysis and does not rely on knowledge specific to the domain in question. Also, PRET's aim is not to discover the underlying physics of the system, but rather to find the simplest ODE that is consistent with the observed behavior. We emphasize, once again, that PRET is not a scientific discovery program (Langley *et al.* 1987) whose goal is to discover new physical phenomena; it is an engineering tool that identifies (already understood) phenomena and models them.

In other automated modeling systems, one of the important features of the formalized domain theory is that it suggests how structural components of the physical system map to candidate model fragments. PRET's true targets are physical systems that do not have a well-defined domain theory, such as complex industrial devices and processes.[8] The R/C car was a first step in this direction; the next test case, on which we are currently working, is a complex machine tool that takes a plastic blank and a prescription and automatically produces an eyeglass lens.

PRET aims to integrate quantitative and qualitative information. Many good papers have reported work in this area, among which are (Williams 1991; Kay & Kuipers 1993).

Statically ordering rules according to their expected computational complexity is a common strategy in rule-based systems, and dynamic meta control has been in use for two decades (Gallaire & Lasserre 1979; Davis 1980). However, we are not aware of any automated modeling program that uses *dynamic* meta control to choose among various reasoning techniques.

Recently, there has been an interesting discussion on the need of domain-dependent control information in *any* application. Theoretically, there is no need for domain-dependent control because control knowledge can be factorized into domain-independent control information and domain-dependent modal information that encodes the structure of the search space (Ginsberg & Geddis 1991). While this elegant result is true for logic programming in general, our particular project (and others as well, e.g., (Minton 1996)) suggested a different approach. Having to think about control in terms of the structure of the search space is exactly what we want to avoid. The implementer of the knowledge base should instead approach it from the viewpoint of his/her domain—in our case as an engineer: which rules are more abstract than others, which rules or goals trigger expensive calls to other packages, and so on.

## Conclusion

System identification (SID) is a necessary first step in many control-theory problems; without a reasonable model of the dynamics, very few systems admit any form of control. PRET automates the system identification process by building an AI layer on top of a set of the kinds of traditional SID techniques that human experts use to solve these problems: regression, curve-fitting, matrix methods, fast-fourier transforms and filtering, root-locus plots, etc. Integrating a collection of techniques like this—a heterogeneous group that is diverse both in methods and in reasoning levels—was an important design goal for the inference system described in this paper. The successful design of that framework allows PRET to intelligently assess the task at hand, and then automatically choose, invoke, and interpret the results of appropriate lower-level methods.

## Acknowledgements

Matt Easley, Apollo Hogan, Brian LaMacchia, Agnes O'Gallagher, Janet Rogers, Ray Spiteri, and Tom Wrensch contributed ideas, code, and/or R/C car data to this project. We would also like to thank the reviewers and the QR community for helpful comments.

[7]Many of these domains also use partial differential equation models. PRET is not designed to work with PDEs, but it *can* produce useful ODE truncations of PDE-governed systems, thus automating another hard and useful part of the modeling art.

[8]In (Münker *et al.* 1997), chemical processes are modeled using a PRET-like approach.

## References


Addanki, S.; Cremonini, R.; and Penberthy, J. S. 1991. Graphs of models. *Artif. Intell.* 51:145–177.

Amsterdam, J. 1992. *Automated Qualitative Modeling of Dynamic Physical Systems.* Ph.D. Dissertation, MIT.

Astrom, K. J., and Eykhoff, P. 1971. System identification — a survey. *Automatica* 7:123–167.

Beckstein, C., and Tobermann, G. 1992. Evolutionary logic programming with RISC. In *4th Intl. Workshop on Logic Programming Environments.*

Beckstein, C.; Stolle, R.; and Tobermann, G. 1996. Meta-programming for generalized horn clause logic. In *META-96.*

Bradley, E., and Easley, M. 1998. Reasoning about sensor data for automated system identification. *Intelligent Data Analysis.* In press. Also in *IDA-97.*

Bradley, E., and Stolle, R. 1996. Automatic construction of accurate models of physical systems. *Annals of Mathematics and Artif. Intell.* 17:1–28.

Bradley, E.; O'Gallagher, A.; and Rogers, J. 1997. Global solutions for nonlinear systems using qualitative reasoning. In *QR-97.*

Capelo, A. C.; Ironi, L.; and Tentoni, S. 1996. The need for qualitative reasoning in automated modeling: A case study. In *QR-96.*

Davis, R. 1980. Meta-rules: Reasoning about control. *Artif. Intell.* 15(3).

Falkenhainer, B., and Forbus, K. D. 1991. Compositional modeling: Finding the right model for the job. *Artif. Intell.* 51:95–143.

Faltings, B., and Gelle, E. 1997. Local consistency for ternary numeric constraints. In *IJCAI-97.*

Forbus, K. D. 1984. Qualitative process theory. *Artif. Intell.* 24:85–168.

Gabbay, D. M., and Sergot, M. J. 1986. Negation as inconsistency I. *J. Logic Progr.* 3(1):1–36.

Gallaire, H., and Lasserre, C. 1979. Controlling knowledge deduction in a declarative approach. In *IJCAI-79.*

Gallaire, H., and Lasserre, C. 1982. Metalevel control for logic programs. In Clark, K., and Tärnlund, S., eds., *Logic Programming.* London: Academic Press.

Ginsberg, M., and Geddis, D. 1991. Is there any need for domain-dependent control information? In *AAAI-91.*

Hogan, A.; Stolle, R.; and Bradley, E. 1998. Putting declarative meta control to work. Technical Report CU-CS-856-98, University of Colorado at Boulder.

Karnopp, D., and Rosenberg, R. 1975. *System Dynamics: A Unified Approach.* New York: John Wiley & Sons.

Kay, H., and Kuipers, B. 1993. Numerical behavior envelopes for qualitative models. In *AAAI-93.*

Kuipers, B. J. 1986. Qualitative simulation. *Artif. Intell.* 29(3):289–338.

Kuipers, B. J. 1992. *Qualitative Reasoning: Modeling and Simulation with incomplete knowledge.* Reading, MA: Addison-Wesley.

Kuipers, B. J. 1993. Reasoning with qualitative models. *Artif. Intell.* 59:125–132.

Langley, P.; Simon, H. A.; Bradshaw, G. L.; and Zytkow, J. M., eds. 1987. *Scientific Discovery: Computational Explorations of the Creative Process.* Cambridge, MA: MIT Press.

Ljung, L., ed. 1987. *System Identification; Theory for the User.* Englewood Cliffs, N.J.: Prentice-Hall.

McCarty, L. T. 1988. Clausal intuitionistic logic I. Fixed-point semantics. *J. Logic Progr.* 5:1–31.

Minton, S. 1996. Is there any need for domain-dependent control information? A reply. In *AAAI-96.*

Mosterman, P. J., and Biswas, G. 1996. A formal hybrid modeling scheme for handling discontinuities in physical system models. In *AAAI-96.*

Münker, B.; Hellinger, S.; Schaich, D.; and King, R. 1997. Application of QR techniques within an integrated tool for automated modelling of chemical reaction systems. In *QR-97.*

Nayak, P. 1995. *Automated Modeling of Physical Systems.* Berlin: Springer. LNCS 1003.

Stolle, R., and Bradley, E. 1996. A customized logic paradigm for reasoning about models. In *QR-96.*

Williams, B. C. 1991. A theory of interactions: unifying qualitative and quantitative algebraic reasoning. *Artif. Intell.* 51:39–94.