Lecture 13: Review of Logic

Kenneth M. Anderson
Foundations of Software Engineering
CSCI 5828 - Spring Semester, 1999

Today's Lecture

- Two different types of logic systems
  - propositional logic
  - predicate logic

Propositional Logic

- A proposition is a statement that is either true or false, but not both
- Propositional Logic is the language of propositions
  - It consists of well-formed formulas constructed from atomic formulas and logical connectives
  - The meaning of a proposition is determined by the truth values assigned to its assertions

Example

- P = "program does not terminate"
- Q = "alarm rings forever"
- P $\Rightarrow$ Q (If the program does not terminate then alarm rings forever)
    - P Q P $\Rightarrow$ Q
    - T T T
    - T F F
    - F T/F T

Formal Language Definition

- terminals = { P, Q, R, …, $\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$, (,)};
- nonterminals = { atomic formula, sentence };
- atomic formula = P | Q | R | … ;
- sentence = atomic formula | (, sentence,) | $\neg$, sentence | sentence, $\vee$, sentence | sentence, $\wedge$, sentence | sentence, $\Rightarrow$, sentence | sentence, $\Leftrightarrow$, sentence;

Example

- sentence
- (sentence)
- (sentence $\vee$ sentence)
- (atomic formula $\vee$ (sentence))
- (P $\vee$ ($\neg$ sentence))
- (P $\vee$ ($\neg$ atomic formula))
- (P $\vee$ ($\neg$ Q))

Truth Table

- Defines values of the logical connectives

P	Q	$\neg$P	P$\vee$Q	P$\wedge$Q	P$\Rightarrow$Q	P$\Leftrightarrow$Q
T	T	F	T	T	T	T
T	F	F	T	F	F	F
F	T	T	T	F	T	F
F	F	T	F	F	T	T

Semantics

- A sentence is true if it evaluates to true after assigning a set of truth values to its atomic propositions
- P and Q are equivalent if they evaluate to the same truth values for every interpretation
  - This is indicated $P \equiv Q$
- A sentence F is satisfiable if it evaluates to true for at least one assignment of truth values, otherwise it is called contradictory

Semantics, continued

- If, for a list of sentences L, every assignment that makes the sentences of L true also makes P true, we say that P is a semantic consequence of L
  - This is written: $L \models P$
- A sentence true for all assignments is called a tautology, the reverse is called a contradiction
- All other sentences are called contingent; they depend on the truth values of their constituents for their truth values
- Note: L |- P means that proposition P can be syntactically derived from L via means of the rules discussed next

Proofs

- A proof is a mechanism for showing that a given claim Q is a logical consequence of some premises $P_1 \ldots P_k$ or
  - $P_1, P_2, \ldots, P_k \models Q$ or
  - $P_1 \wedge P_2 \wedge \ldots \wedge P_k \Rightarrow Q$ or
  - $\neg(P_1 \wedge P_2 \wedge \ldots \wedge P_k) \vee Q$
- In order to establish the proof this final form must be shown to be a tautology

Logical Equivalences

- double negation
  - $\neg\neg p \Leftrightarrow p$
- commutative
  - $(p \vee q) \Leftrightarrow (q \vee p)$
  - $(p \wedge q) \Leftrightarrow (q \wedge p)$
  - $(p \Leftrightarrow q) \Leftrightarrow (q \Leftrightarrow p)$
- associative
  - $(p \vee q) \vee r \Leftrightarrow p \vee (q \vee r)$
  - $(p \wedge q) \wedge r \Leftrightarrow p \wedge (q \wedge r)$
- distributive
  - $p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$
  - $p \wedge (q \vee r) \Leftrightarrow (p \wedge q) \vee (p \wedge r)$
- DeMorgan laws
  - $\neg(p \vee q) \Leftrightarrow (\neg p \wedge \neg q)$
  - $\neg(p \wedge q) \Leftrightarrow (\neg p \vee \neg q)$
- Implication
  - $(p \Rightarrow q) \Leftrightarrow (\neg p \vee q)$
  - $(p \Rightarrow q) \Leftrightarrow \neg(p \wedge \neg q)$

Deduction Rules

- $\vee$-Introduction
  - $A \Rightarrow A \vee B$
- $\wedge$-Introduction
  - $A, B \Rightarrow A \wedge B$
- $\neg$-Introduction
  - $(A \text{ |- false}) \Rightarrow \neg A$
- $\Rightarrow$-Introduction
  - $(A \text{ |- } B) \Rightarrow (A \Rightarrow B)$
- $\Leftrightarrow$-Introduction
  - $(A \text{ |- } B), (B \text{ |- } A) \Rightarrow (A \Leftrightarrow B)$
- $\vee$-Elimination
  - $A \vee B, A \text{ |- } y, B \text{ |- } y \Rightarrow y$
- $\wedge$-Elimination
  - $A \wedge B \Rightarrow A, B$
- $\neg$-Elimination
  - $\neg\neg A \Rightarrow A, \neg A, A \Rightarrow \text{false}$
- $\Rightarrow$-Elimination
  - $A, (A \Rightarrow B) \Rightarrow B$
- $\Leftrightarrow$-Elimination
  - $A \Leftrightarrow B \Rightarrow (A \Rightarrow B), (B \Rightarrow A)$

Prove $P \vee (Q \wedge R) \vdash (P \vee Q) \wedge (P \vee R)$

1. P	Premise
2. $P \vee Q$	$\vee$-Introduction
3. $P \vee R$	$\vee$-Introduction
4. $(P \vee Q) \wedge (P \vee R)$	$\wedge$-Introduction
5. $Q \wedge R$	Premise
6. Q	$\wedge$-Elimination
7. $P \vee Q$	$\vee$-Introduction
8. R	$\wedge$-Elimination and 5
9. $P \vee R$	$\vee$-Introduction
10. $(P \vee Q) \wedge (P \vee R)$	$\wedge$-Introduction, 7, 9
11. $P \vee (Q \wedge R) \vdash (P \vee Q) \wedge (P \vee R)$	$\vee$-Elimination and 1-4, 5-10

By Truth Table

P	Q	R	(P	$\vee$	(Q	$\wedge$	R))	$\Leftrightarrow$	((P	$\vee$	Q)	$\wedge$	(P	$\vee$	R))
T	T	T	T	**T**	T	T	T	**T**	T	T	T	**T**	T	T	T
T	T	F	T	**T**	T	F	F	**T**	T	T	T	**T**	T	T	F
T	F	T	T	**T**	F	F	T	**T**	T	T	F	**T**	T	T	T
T	F	F	T	**T**	F	F	F	**T**	T	T	F	**T**	T	T	F
F	T	T	F	**T**	T	T	T	**T**	F	T	T	**T**	F	T	T
F	T	F	F	**F**	T	F	F	**T**	F	T	T	**F**	F	F	F
F	F	T	F	**F**	F	F	T	**T**	F	F	F	**F**	F	T	T
F	F	F	F	**F**	F	F	F	**T**	F	F	F	**F**	F	F	F
1	1	1	1	3	1	2	1	7	1	4	1	6	1	5	1

Tautology concludes proof

Resolution Rules

- Resolution
  - $(A \vee P), (B \vee \neg P) \Rightarrow A \vee B$
- Chain Rule
  - $(A \Rightarrow P), (P \Rightarrow B) \Rightarrow (A \Rightarrow B)$
- Modus Ponens
  - $P, (P \Rightarrow A) \Rightarrow A$

Proof by Contradiction

- To establish
  - $P_1, P_2, \ldots, P_k \models Q$
- Negate Q
- Transform Ps and Q to conjunctive normal form
  - Example $(P \vee Q) \wedge (Q \vee S) \wedge \ldots$
- Apply resolution (and other) rules repeatedly until P and $\neg P$ are derived
- These negate and the proof is achieved

Example

- Assume $P \Rightarrow Q, R \vee P$
  - Show that $R \Rightarrow S \mid\!- S \vee Q$
- Premises are $P \Rightarrow Q$, $R \vee P$, and $R \Rightarrow S$
- CNF: $\neg P \vee Q$, $R \vee P$, $\neg R \vee S$
- Negation of conclusion: $\neg(S \vee Q) \Leftrightarrow \neg S \wedge \neg Q$

Example, continued

1. $\neg P \vee Q$	• Premise
2. $R \vee P$	• Premise
3. $\neg R \vee S$	• Premise
4. $\neg S \wedge \neg Q$	• Negation of Conclusion
5. $\neg S$	• $\wedge$-elimination
6. $\neg Q$	• $\wedge$-elimination
7. $R \vee Q$	• (1), (2), resolution
8. $\neg R$	• (3), (5)
9. Q	• (7), (8)
10. NIL	• (6), (9)

Consistency

- Propositional logic is consistent
  - All provable statements are semantically true
    - That is if a set of premises S syntactically entail a proposition P then there is an interpretation in which P can be reasoned about from S.
    - Formally, if S |- P, then S |= P

Completeness

- Propositional logic is complete
  - All semantically true statements are provable
    - That is, if a set of premises S semantically entails a proposition P, then P can be derived formally (syntactically).
    - Formally, S |= P, then S |- P
- One important consequence
  - Decidability
    - Given a finite set of propositions S and a proposition P, there is an algorithm that determines whether or not S |= P

Why is decidability important?

- When a specification S is created with propositional logic
  - decidability confirms that S can be analyzed to demonstrate whether a property P holds in S or not.

Library Example

- S: a book is on the stacks
- R: a book is on reserve
- L: a book is on loan
- Q: a book is requested
- Constraints
  - A book can be in only one of three states S, R, and L
  - If a book is on the stacks or on reserve then it can be requested

Library Example, continued

- Constraints specified as propositions
  - $S \Leftrightarrow \neg(R \vee L)$
  - $R \Leftrightarrow \neg(S \vee L)$
  - $L \Leftrightarrow \neg(S \vee R)$
  - $S \vee R \Rightarrow Q$
- Homework 1 (submit via e-mail by Lec. 15)
  - Prove "if a book is on loan then it is not requested" is a logical consequence

Predicate Logic

- Propositional Logic cannot specify the relationships between objects
  - It can only assert that particular properties hold or do not hold within a set of propositions
- Predicate Logic has the power to do so
  - consists of
    - constants, predicates, variables, and functions

Examples

- constants
  - computer, mary, 2, …
- variables
  - x, y, z
- predicates
  - mammal(x), parent(x, y)
- functions
  - father(x), sqrt(x)

Formal syntax of predicate logic

- wff = proposition | predicate | ¬wff | quantified-wff | (, wff, op, wff,);
- proposition = P | Q| R | …;
- predicate = predicate_name, (, term_list,);
- predicate_name = IDENTIFIER;
- term_list = term | term, “,”, term_list;
- term = CONSTANT | variable | function, (term_list,);
- variable = VARNAME; function = IDENTIFIER;
- quantified-wff = quantifier, “•”, wff;
- quantifier = ∃, variable | ∀, variable;
- op = ∧ | ∨ | ⇔ | ⇒

Example well-formed formulas

- ∀ x • ∃ y • (less(square(x), y))
- ∀ x • ∀ y • (likes(x, y) ⇒ marry(x, y))
- ∃ x • ∃ y • (airline(x) ∧ city(y) ∧ flies(x, y))
- ∀ x • ∃ y,z • (airline(x) ∧ city(y) ∧ city(z) ∧ flies(x, y) ∧ flies(x, z) ⇒ (y=z))
  - ∀ x • ∃!y • (airline(x) ∧ city(y) ∧ flies(x, y))
  - ∃! is a shorthand to express uniqueness
- Note: predicates are Boolean *n*-ary functions

Binding Variables

- x and y are bound
  - ∀ x : jobs • ∃ y : queues • (¬executing(x) ⇒ has(y, x))
- only y is bound
  - ∃ y • on(x, y)
- When all variables are bound, we call the wff a closed formula
- All closed formulas can be interpreted as a proposition

Example use of predicate logic

- Consider lines and points on a plane
  - (1) two lines meet at a unique point
  - (2) there is a unique line through any two points
  - line(x) = x is a line
  - point(x) = x is a point
  - lies_on(x, y) = point x is contained in line y

Example, continued

- domain distinction
  - (a) $\forall$ x • (point(x) $\vee$ line(x));
  - (b) $\forall$ x • ($\neg$(point(x) $\wedge$ line(x)));
- incidence
  - $\forall$ x, y • (lies_on(x, y) $\Rightarrow$ (point(x) $\wedge$ line(y)));

Example, continued

- equality for lines
  - $\exists$ x1, x2 • ($\neg$(x1 = x2) $\wedge$ lies_on(x1, y1) $\wedge$ lies_on(x1, y2) $\wedge$ lies_on(x2, y1) $\wedge$ lies_on(x2, y2)) $\Rightarrow$ y1 = y2;

Example, continued

- unique line
  - $\forall$ x, y • ((point(x) $\wedge$ point(y) $\wedge$ $\neg$(x=y)) $\Rightarrow$ $\exists$!z•(lies_on(x, z) $\wedge$ lies_on(y,z)));
- unique intersection
  - $\forall$ x, y • ((line(x) $\wedge$ line(y) $\wedge$ $\neg$(x=y)) $\Rightarrow$ $\exists$!z•(lies_on(z, x) $\wedge$ lies_on(z, y)));