Propositional Calculus - Hilbert system H

Moonzoo Kim CS Division of EECS Dept. KAIST

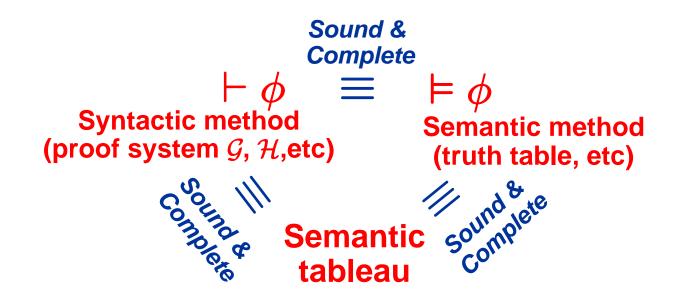
moonzoo@cs.kaist.ac.kr http://pswlab.kaist.ac.kr/courses/cs402-07



Review

Goal of logic

- To check whether given a formula ϕ is valid
- To prove a given formula ϕ





Review (cont.)

Remember the following facts

Although we have many binary operators ({∨,∧,→,←,↔, ↓, ↑,⊕}), ↑ can replace all other binary operators through semantic equivalence. Similarly, {→, ¬} is an adequate set of binary operators.

• $\nvDash \phi$ does not necessarily mean $\vDash \neg \phi$

- Deductive proof cannot disprove \u03c6 (i.e. claiming that there does not exist a proof for \u03c6) while semantic method can show both validity and satisfiability of \u03c6
- Very few logics have decision procedure for validity check (i.e., truth table). Thus, we use deductive proof in spite of the above weakness.
- A proof tree in G grows up while a proof tree in H shrinks down according to characteristics of its inference rules

Thus, a proof in \mathcal{G} is easier than a proof in \mathcal{H} in general



Sound verification tools

- Suppose that
 - there is a target software S
 - there is a formal requirement R
- We can make a state machine (automata) of S, say A_S
 - A state of A_S consists of all variables including a program counter.
- Any state machine can be encoded into a predicate logic formual ϕ_{a_s}
 - We will see this encoding in the first order logic classes
- Program verification is simply to prove $\phi_{A_S} \vDash \mathbb{R}$
- For this purpose, we use a formal verification tool V so that $\phi_{A_S} \vdash_V R$
 - We call V is sound whenever S has a bug, V always detects the bug

•
$$\phi_{A_S} \nvDash \mathsf{R} \Rightarrow \phi_{A_S} \nvDash_{\mathsf{V}} \mathsf{R} \text{ (iff } \phi_{A_S} \vdash_{\mathsf{V}} \mathsf{R} \Rightarrow \phi_{A_S} \vDash \mathsf{R} \text{)}$$

- We call V is complete whenever V detects a bug, that bug is a real bug. ■ $\phi_{A_S} \nvDash_V \mathsf{R} \to \phi_{A_S} \nvDash_\mathsf{R}$ (iff $\phi_{A_S} \vDash \mathsf{R} \Rightarrow \phi_{A_S} \vdash_V \mathsf{R}$)
- In reality, most formal verification tools are just sound, not complete (I.e., formal verification tools may raise false alarms). However, for debugging purpose, soundness is great.



The Hilbert system ${\boldsymbol{\mathcal H}}$

- Def 3.9 \mathcal{H} is a deductive system with three axiom schemes and one rule of inference.
 - For any formulas A,B,C, the following formulas are axioms (in fact axiom schemata):
 - Axiom1: \vdash (A \rightarrow (B \rightarrow A))
 - Axiom2: \vdash (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))
 - Axiom3: $\vdash (\neg B \rightarrow \neg A) \rightarrow (A \rightarrow B))$
 - The rule of inference is called modus ponens (MP). For any formulas A,B

$$\begin{array}{c|c} -A & \vdash A \rightarrow B \\ & \vdash B \end{array}$$

- Note that axioms used in a proof in H are usually very long because the MP rule reduces a length of formula (see Thm 3.10)
 - at least one premise ($\vdash A \rightarrow B$) is longer than conclusion (B)



\mathcal{G} v.s. \mathcal{H}

- G is a deductive system for a set of formulas while
 H is a deductive system for a single formula
- G has one form of axiom and many rules (for 8 αrules and 7 β-rules) while H has several axioms (in fact axiom schemes) but only one rule



Derived rules

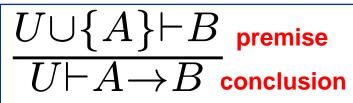
- Def. 3.12 Let U be a set of formulas and A a formula. The notation U ⊢ A means that the formulas in U are assumptions in the proof of A. If A_i ∈ U, a proof of U ⊢ A may include an element of the form U ⊢ A_i
- Rule 3.13 Deduction rule

$$\frac{U \cup \{A\} \vdash B}{U \vdash A \to B}$$

 Note that deduction rule increase the size of a formula, thus making a proof easier



Soundness of deduction rule



Thm 3.14 The deduction rule is a sound derived rule

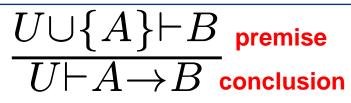
- By induction on the length **n** of the proof $U \cup \{A\} \vdash B$
 - For n=1, B is proved in one step, so B must be either an element of $U \cup \{A\}$ or an axiom of \mathcal{H}
 - If B is A, then $\vdash A \rightarrow B$ by Thm 3.10 ($\vdash A \rightarrow A$), so certainly U $\vdash A \rightarrow B$.
 - Otherwise (i.e., $B \in U$ or B is an axion), the following is a proof of $U \vdash A \rightarrow B$

$$\frac{\mathsf{U} \vdash \mathsf{B} \qquad \mathsf{U} \vdash \mathsf{B} \rightarrow (\mathsf{A} \rightarrow \mathsf{B})}{\mathsf{U} \vdash \mathsf{A} \rightarrow \mathsf{B}} \mathsf{MP}$$



Soundness of deduction rule

MP



- For n>1, the last step in the proof of $U \cup \{A\} \vdash B$ is either
 - a one-step inference of B
 - the result holds by the proof for n =1
 - an inference of B using MP.
 - there is a formula C such that formula I in the proof is $U \cup \{A\} \vdash C$ and formula j is $U \cup \{A\} \vdash C \rightarrow B$, for I, j < n. By the inductive hypothesis $U \vdash A \rightarrow C$ and $U \vdash A \rightarrow (C \rightarrow B)$. A proof of $U \vdash A \rightarrow B$ is given by

$$\begin{array}{ccc} \mathsf{U}\vdash\mathsf{A}\to\mathsf{B} & \mathsf{U}\vdash\mathsf{A}\to(\mathsf{C} & \mathsf{U}\vdash\mathsf{A}\to(\mathsf{C}\to\mathsf{B}) \\ & \to\mathsf{B} \end{array}$$

 $U \vdash A \rightarrow B$



Theorems and derived rules in \mathcal{H}

- Note that any theorem of the form $U \vdash A \rightarrow B$ justifies a derived rule of the form $\underbrace{U \vdash A}_{U \vdash B}$ simply by using MP on
- Rule 3.15 Contrapositive rule
 by Axiom 3 ⊢ (¬B→¬A) → (A→B))

$$\frac{U \vdash \neg B \rightarrow \neg A}{U \vdash A \rightarrow B}$$

- Rule 3.17 Transitivity rule ■ by Thm 3.16 \vdash (A \rightarrow B) \rightarrow [(B \rightarrow C) \rightarrow (A \rightarrow C)] $U \vdash A \rightarrow C$ $U \vdash A \rightarrow C$
- Rule 3.19 Exchange of antecedent rule
 by Thm 3.18 \begin{aligned} U & (A \rightarrow (B \rightarrow C)] \rightarrow [(B \rightarrow (A \rightarrow C)] & U & (B \rightarrow (A \rightarrow C)] \rightarrow U & (B \rightarrow C)] & U & (B \rightarrow (A \rightarrow C)) \rightarrow U & (B \rightarrow C)] & U & (B \rightarrow (A \rightarrow C)) \rightarrow U & (B \rightarrow C)] & U & (B \rightarrow (A \rightarrow C)) \rightarrow U & (B \rightarrow C)] & U & (B \rightarrow (A \rightarrow C)) \rightarrow U & (B \rightarrow C)] & U & (B \rightarrow (A \rightarrow C)) \rightarrow U & (B \rightarrow C)] & U & (B \rightarrow (A \rightarrow C)) \rightarrow U & (B \rightarrow C)] & U & (B \rightarrow (A \rightarrow C)) \rightarrow U & (B \rightarrow C)] & U & (B \rightarrow (A \rightarrow C)) \rightarrow U & (B \rightarrow C)] & U & (B \rightarrow (A \rightarrow C)) \rightarrow U & (B \rightarrow C)] & U & (B \rightarrow (A \rightarrow C)) \rightarrow U & (B \rightarrow C)] & U & (B \rightarrow (A \rightarrow C)) \rightarrow U & (B \rightarrow (A \rightarrow C)) \rightarrow U & (B \rightarrow (A \rightarrow C)) & U & (B \rightarrow (A \rightarrow C)) \rightarrow U & (B \rightarrow (A \rightarrow C)) & (B \rightarrow (A \rightarrow (A \rightarrow C))) & (B \rightarrow (A \rightarrow (A \rightarrow C))) & (B \rightarrow (A \rightarrow (A \rightarrow C))) & (B \rightarrow (A \rightarrow (A