# Speaking Logic

N. Shankar

Computer Science Laboratory
SRI International
Menlo Park, CA

May 20, 2014

# Why Logic?

- Computing, like mathematics, is the study of reusable abstractions.
- Abstractions in computing include numbers, lists, channels, processes, protocols, and programming languages.
- These abstractions have algorithmic value in designing, representing, and reasoning about computational processes.
- Properties of abstractions are captured by precisely stated laws through *formalization* using axioms, definitions, theorems, and proofs.
- Logic is the *medium* for expressing these abstract laws and the *method* for deriving consequences of these laws using sound reasoning principles.
- Computing is *abstraction engineering*.
- Logic is the calculus of computing.

# The Unreasonable Effectiveness of Logic in Computing

- The world is increasingly an interplay of abstractions'
- Caches, files, IP addresses, avatars, friends, likes, hyperlinks, packets, network protocols, and cyber-physical systems are all examples of abstractions in daily use.
- Such abstract entities and the relationships can be expresses clearly and precisely in logic.
- In computing, and elsewhere, we are becoming increasingly dependent on formalization as a way of managing the abstract universe.

# Where Logic has Been Effective

Logic has been *unreasonably* effective in computing, with an impact that spans

- Theoretical computer science: Algorithms, Complexity, Descriptive Complexity
- Hardware design and verification: Logic design, minimization, synthesis, model checking
- Software verification: Specification languages, Assertional verification, Verification tools
- Computer security: Information flow, Cryptographic protocols
- Programming languages: Logic/functional programming, Type systems, Semantics
- Artificial intelligence: Knowledge representation, Planning
- Databases: Data models, Query languages
- Systems biology: Process models

Our course is about the effective use of logic in computing.

# Speaking Logic

- In mathematics, logic is studied as a source of interesting (meta-)theorems, but the reasoning is typically informal.
- In philosophy, logic is studied as a minimal set of foundational principles from which knowledge can be derived.
- In computing, the challenge is to solve large and complex problems through abstraction and decomposition.
- Formal, logical reasoning is needed to achieve scale and correctness.
- We will examine how logic is used to formulate problems, find solutions, and build proofs.
- We will also examine useful metalogical properties of logics, as well as algorithmic methods for effective inference.

# Course Schedule

- The course is spread over three lectures:
  - **Lecture 1:** Propositional Logics
  - **Lecture 2:** First-Order and Higher-Order Logic
  - **Lecture 3:** Advanced topics
- The goal is to learn how to speak logic fluently through the use of propositional, modal, equational, first-order, and higher-order logic.
- This will serve as a background for the more sophisticated ideas in the main lectures in the school.
- To get the most out of the course, please do the exercises.

# A Small Puzzle [Wason]

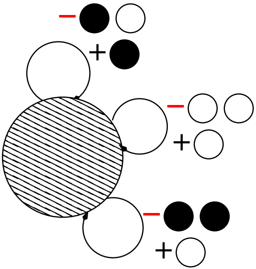
- Given four cards laid out on a table as: $\boxed{\text{D}}$, $\boxed{\text{3}}$, $\boxed{\text{F}}$, $\boxed{\text{7}}$, where each card has a letter on one side and a number on the other.
- Which cards should you flip over to determine if every card with a $\boxed{\text{D}}$ on one side has a $\boxed{\text{7}}$ on the other side?

# A Small Problem

Given a bag containing some black balls and white balls, and a stash of black/white balls. Repeatedly

1. Remove a random pair of balls from the bag
2. If they are the same color, insert a white ball into the bag
3. If they are of different colors, insert a black ball into the bag

What is the color of the last ball?



# Truthtellers and Liars [Smullyan]

- You are confronted with two gates.
- One gate leads to the castle, and the other leads to a trap
- There are two guards at the gates: one always tells the truth, and the other always lies.
- You are allowed to ask one of the guards on question with a yes/no answer.
- What question should you ask in order to find out which gate leads to the castle?

# Mr. S and Mr. P

- Two integers $m$ and $n$ are picked from the interval $[2, 99]$.
- Mr. S is given the sum $m + n$. and Mr. P is given the product $mn$.
- They then have the following dialogue:

  **S:** *I don't know m and n.*
  **P:** *Me neither.*
  **S:** *I know that you don't.*
  **P:** *In that case, I do know m and n.*
  **S:** *Then, I do too.*

- How would you determine the numbers $m$ and $n$?

# Gilbreath's Card Trick

- Start with a deck consisting of a stack of quartets, where the cards in each quartet appear in suit order $\spadesuit$, $\heartsuit$, $\clubsuit$, $\diamondsuit$:

$$\begin{array}{l}\langle 5\spadesuit\rangle, \langle 3\heartsuit\rangle, \langle Q\clubsuit\rangle, \langle 8\diamondsuit\rangle,\\ \langle K\spadesuit\rangle, \langle 2\heartsuit\rangle, \langle 7\clubsuit\rangle, \langle 4\diamondsuit\rangle,\\ \langle 8\spadesuit\rangle, \langle J\heartsuit\rangle, \langle 9\clubsuit\rangle, \langle A\diamondsuit\rangle\end{array}$$

- Cut the deck, say as $\langle 5\spadesuit\rangle, \langle 3\heartsuit\rangle, \langle Q\clubsuit\rangle, \langle 8\diamondsuit\rangle, \langle K\spadesuit\rangle$ and $\langle 2\heartsuit\rangle, \langle 7\clubsuit\rangle, \langle 4\diamondsuit\rangle, \langle 8\spadesuit\rangle, \langle J\heartsuit\rangle, \langle 9\clubsuit\rangle, \langle A\diamondsuit\rangle$.
- Reverse one of the decks as $\langle K\spadesuit\rangle, \langle 8\diamondsuit\rangle, \langle Q\clubsuit\rangle, \langle 3\heartsuit\rangle, \langle 5\spadesuit\rangle$.
- Now shuffling, for example, as

$$\begin{array}{l}\langle 2\heartsuit\rangle, \langle 7\clubsuit\rangle, \underline{\langle K\spadesuit\rangle}, \underline{\langle 8\diamondsuit\rangle},\\ \langle 4\diamondsuit\rangle, \langle 8\spadesuit\rangle, \underline{\langle Q\clubsuit\rangle}, \langle J\heartsuit\rangle,\\ \underline{\langle 3\heartsuit\rangle}, \langle 9\clubsuit\rangle, \underline{\langle 5\spadesuit\rangle}, \langle A\diamondsuit\rangle\end{array}$$

- *Each quartet contains a card from each suit. Why?*

# Pigeonhole Principle

Why can't you park $n+1$ cars in $n$ parking spaces, if each car needs its own space?

# Hard Sudoku [Wikipedia/Algorithmics_of_Sudoku]

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| | | | | | 3 | | 8 | 5 |
| | | 1 | | 2 | | | | |
| | | | 5 | | 7 | | | |
| | | 4 | | | | 1 | | |
| | 9 | | | | | | | |
| 5 | | | | | | | 7 | 3 |
| | | 2 | | 1 | | | | |
| | | | | 4 | | | | 9 |

# What is Logic?

- Logic is the art and science of effective reasoning.
- How can we draw general and reliable conclusions from a collection of facts?
- Formal logic: Precise, syntactic characterizations of well-formed expressions and valid deductions.
- Formal logic makes it possible to *calculate* consequences so that each step is verifiable by means of proof.
- Computers can be used to automate such symbolic calculations.

# Logic Basics

- Logic studies the *trinity* between *language*, *interpretation*, and *proof*.
- *Language*: What are you allowed to say?
- *Interpretation*: What is the intended meaning?
  - Meaning is usually *compositional*: Follows the syntax
  - Some symbols have fixed meaning: connectives, equality, quantifiers
  - Other symbols are allowed to vary variables, functions, and predicates
  - *Assertions* either hold or fail to hold in a given interpretation
  - A *valid* assertion holds in every interpretation
- *Proofs* are used to demonstrate validity

# Propositional Logic

- Propositional logic can be more accurately described as a logic of conditions – *propositions are always true or always false.* [Couturat, *Algebra of Logic*]
- A condition can be represented by a propositional variable, e.g., $p$, $q$, etc., so that distinct propositional variables can range over possibly different conditions.
- The conjunction, disjunction, and negation of conditions are also conditions.
- The syntactic representation of conditions is using propositional formulas:

$$\phi := P \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2$$

- $P$ is a class of propositional variables: $p_0, p_1, \ldots$.
- Examples of formulas are $p$, $p \wedge \neg p$, $p \vee \neg p$, $(p \wedge \neg q) \vee \neg p$.

- In logic, the meaning of an expression is constructed compositionally from the meanings of its subexpressions.
- The meanings of the symbols are either *fixed*, as with $\neg$, $\wedge$, and $\vee$, or allowed to vary, as with the propositional variables.
- An interpretation (truth assignment) $M$ assigns truth values $\{\top, \bot\}$ to propositional variables: $M(p) = \top \iff M \models p$.
- $M[\![A]\!]$ is the meaning of $A$ in $M$ and is computed using truth tables:

| $\phi$ | $p$ | $q$ | $\neg p$ | $p \vee q$ | $p \wedge q$ |
|---|---|---|---|---|---|
| $M_1(\phi)$ | $\bot$ | $\bot$ | $\top$ | $\bot$ | $\bot$ |
| $M_2(\phi)$ | $\bot$ | $\top$ | $\top$ | $\top$ | $\bot$ |
| $M_3(\phi)$ | $\top$ | $\bot$ | $\bot$ | $\top$ | $\bot$ |
| $M_4(\phi)$ | $\top$ | $\top$ | $\bot$ | $\top$ | $\top$ |

# Truth Tables

We can use truth tables to evaluate formulas for validity/satisfiability.

| $p$ | $q$ | $(\neg p \vee q)$ | $(\neg(\neg p \vee q) \vee p)$ | $\neg(\neg(\neg p \vee q) \vee p) \vee p$ |
|---|---|---|---|---|
| $\bot$ | $\bot$ | $\top$ | $\bot$ | $\top$ |
| $\bot$ | $\top$ | $\top$ | $\bot$ | $\top$ |
| $\top$ | $\bot$ | $\bot$ | $\top$ | $\top$ |
| $\top$ | $\top$ | $\top$ | $\top$ | $\top$ |

How many rows are there in the truth table for a formula with $n$ distinct propositional variables?

# Problems

- Define the operation of substituting a formula $A$ for a variable $p$ in a formula $B$, i.e., $B[p \mapsto A]$.
- Is the result always a well-formed formula?
- Can the variable $p$ occur in $B[p \mapsto A]$?
- What is the truth-table meaning of $B[p \mapsto A]$ in terms of the meaning of $B$ and $A$?

# Defining New Connectives

- How do you define $\wedge$ in terms of $\neg$ and $\vee$?
- Give the truth table for $A \Rightarrow B$ and define it in terms of $\neg$ and $\vee$.
- Define bi-implication $A \iff B$ in terms of $\Rightarrow$ and $\wedge$ and show its truth table.
- An $n$-ary Boolean function maps $\{\top, \bot\}^n$ to $\{\top, \bot\}$
- Show that every $n$-ary Boolean function can be defined using $\neg$ and $\vee$.
- Using $\neg$ and $\vee$ define an $n$-ary parity function which evaluates to $\top$ iff the parity is odd.
- Define an $n$-ary function which determines that the unsigned value of the little-endian input $p_0, \ldots, p_{n-1}$ is even?
- Define the $NAND$ operation, where $NAND(p, q)$ is $\neg(p \wedge q)$ using $\neg$ and $\vee$. Conversely, define $\neg$ and $\vee$ using $NAND$.

## Satisfiability and Validity

- An interpretation $M$ is a model of a formula $\phi$ if $M \models \phi$.
- If $M \models \neg\phi$, then $M$ is a *countermodel* for $\phi$.
- When $\phi$ has a model, it is said to be *satisfiable*.
- If it has no model, then it is *unsatisfiable*.
- If $\neg\phi$ is unsatisfiable, then $\phi$ is valid, i.e., alway evaluates to $\top$.
- We write $\phi \models \psi$ if every model of $\phi$ is a model of $\psi$.
- If $\phi \wedge \neg\psi$ is unsatisfiable, then $\phi \models \psi$.

# Satisfiable, Unsatisfiable, or Valid?

- Classify these formulas as satisfiable, unsatisfiable, or valid?
  - $p \vee \neg p$
  - $p \wedge \neg p$
  - $\neg p \Rightarrow p$
  - $((p \Rightarrow q) \Rightarrow p) \Rightarrow p$
- Make up some examples of formulas that are satisfiable (unsatisfiable, valid)?
- If $A$ and $B$ are satisfiable, is $A \wedge B$ satisfiable? What about $A \vee B$.
- Can $A$ and $\neg A$ both be satisfiable (unsatisfiable, valid)?

# Some Valid Laws

- $\neg(A \wedge B) \iff \neg A \vee \neg B$
- $\neg(A \vee B) \iff \neg A \wedge \neg B$
- $((A \vee B) \vee C) \iff A \vee (B \vee C)$
- $(A \Rightarrow B) \iff (\neg A \vee B)$
- $(\neg A \Rightarrow \neg B) \iff (B \Rightarrow A)$
- $\neg\neg A \iff A$
- $A \Rightarrow B \iff \neg A \vee B$
- $\neg(A \wedge B) \iff \neg A \vee \neg B$
- $\neg(A \vee B) \iff \neg A \wedge \neg B$
- $\neg A \Rightarrow B \iff \neg B \Rightarrow A$

# What Can Propositional Logic Express?

- Constraints over bounded domains can be expressed as satisfiability problems in propositional logic (SAT).
- Define a 1-bit full adder in propositional logic.
- The Pigeonhole Principle states that if $n+1$ pigeons are assigned to $n$ holes, then some hole must contain more than one pigeon. Formalize the pigeonhole principle for four pigeons and three holes.
- Formalize the statement that a graph of $n$ elements is $k$-colorable for given $k$ and $n$ such that $k < n$.
- Formalize and prove the statement that given a symmetric and transitive graph over 3 elements, either the graph is complete or contains an isolated point.
- Formalize *Sudoku* and Latin Squares in propositional logic.

# Using Propositional Logic

- Write a propositional formula for checking that a given finite automaton $\langle Q, \Sigma, q, F, \delta\rangle$ with
  - Alphabet $\Sigma$,
  - Set of states $S$
  - Initial state $q$,
  - Set of final states $F$, and
  - Transition function $\delta$ from $\langle Q, \Sigma\rangle$ to $Q$

  accepts some string of length 5.
- Describe an $N$-bit ripple carry adder with a carry-in and carry-out bits as a formula.

# Cook's Theorem

- A Turing machine consists of a finite automaton reading (and writing) symbols from a tape.
- The finite automaton (in a non-accepting state) reads the symbol at the current position of the head, and nondeterministically executes a step consisting of
  1. A new symbol to write at the head position
  2. A move (left or right) of the head from the current position
  3. A next automaton state
- Show that SAT is solvable in polynomial time (in the size of the input) by a nondeterministic Turing machine.
- Show that for any nondeterministic Turing machine and polynomial bound $p(n)$ for input of size $n$, one can (in polynomial time) construct a propositional formula which is satisfiable iff there is the Turing machine accepts the input in at most $p(n)$.

# Proof Systems

- There are three basic styles of proof systems.
- These are distinguished by their basic judgement.
  1. Hilbert systems: $\vdash A$ means the formula $A$ is provable.
  2. Natural deduction: $\Gamma \vdash A$ means the formula $A$ is provable from a set of assumption formulas $\Gamma$.
  3. Sequent Calculus: $\Gamma \vdash \Delta$ means the consequence of $\bigvee \Delta$ from $\bigwedge \Gamma$ is derivable.

# Hilbert System (H) for Propositional Logic

- The basic judgement here is $\vdash A$ asserting that a formula is *provable*.
- We can pick $\Rightarrow$ as the basic connectives
- The axioms are
  - $\overline{\vdash A \Rightarrow A}$
  - $\overline{\vdash A \Rightarrow (B \Rightarrow A)}$
  - $\overline{\vdash (A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))}$
- A single rule of inference (Modus Ponens) is given

$$\frac{\vdash A \qquad \vdash A \Rightarrow B}{\vdash B}$$

- Can you prove $((p \Rightarrow q) \Rightarrow p) \Rightarrow p$ using the above system?

# Hilbert System (H)

- Add a propositional constant $\bot$ to the formula syntax, where $[\![\bot]\!] = \bot$.
- Define negation $\neg A$ as $A \Rightarrow \bot$.
- Can you prove
  1. $\neg A \Rightarrow (A \Rightarrow \neg B)$
  2. $\neg A \Rightarrow (A \Rightarrow \bot)$
  3. $\neg\neg A \Rightarrow A$
- Are any of the axioms redundant? [Hint: See if you can prove the first axiom from the other two.]
- Write Hilbert axioms for $\wedge$ and $\vee$.

# Deduction Theorem

- We write $\Gamma \vdash A$ for a set of formulas $\Gamma$, if $\vdash A$ can be proved given $\vdash B$ for each $B \in \Gamma$.
- Deduction theorem: Show that if $\Gamma, A \vdash B$, then $\Gamma \vdash A \Rightarrow B$, where $\Gamma, A$ is $\Gamma \cup \{A\}$. [Hint: Use induction on proofs.]
- A *derived* rule of inference has the form

$$\frac{P_1, \ldots, P_n}{C}$$

  where there is a derivation in the base logic from the premises $P_1, \ldots, P_n$ to the conclusion $C$.
- An *admissible* rule of inference is one where the conclusion $C$ is provable if the premises $P_1, \ldots, P_n$ are provable.
- Every derived rule is admissible, but what is an example of an admissible rule that is not a derived one?

# Natural Deduction for Propositional Logic

- In natural deduction (ND), the basic judgement is $\Gamma \vdash A$.
- The rules are classified according to the introduction or elimination of connectives from $A$ in $\Gamma \vdash A$.
- The axiom, introduction, and elimination rules of natural deduction are
  - $\dfrac{}{\Gamma, A \vdash A}$
  - $\dfrac{\Gamma_1 \vdash A \qquad \Gamma_2 \vdash A \Rightarrow B}{\Gamma_1 \cup \Gamma_2 \vdash B}$
  - $\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}$
- Use ND to prove the axioms of the Hilbert system.
- A proof is in *normal form* if no introduction rule appears above an elimination rule. Can you ensure that your proofs are always in normal form? Can you write an algorithm to convert non-normal proofs to normal ones?

The basic judgement is $\Gamma \vdash \Delta$ asserting that $\bigwedge \Gamma \Rightarrow \bigvee \Delta$, where $\Gamma$ and $\Delta$ are sets (or bags) of formulas.

| | Left | Right |
|---|---|---|
| Ax | $\dfrac{}{\Gamma, A \vdash A, \Delta}$ | |
| $\neg$ | $\dfrac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta}$ | $\dfrac{\Gamma, A \vdash \Delta}{\Gamma \vdash \neg A, \Delta}$ |
| $\vee$ | $\dfrac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta}$ | $\dfrac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \vee B, \Delta}$ |
| $\wedge$ | $\dfrac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}$ | $\dfrac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}$ |
| $\Rightarrow$ | $\dfrac{\Gamma, B \vdash \Delta \quad \Gamma \vdash A, \Delta}{\Gamma, A \Rightarrow B \vdash \Delta}$ | $\dfrac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \Rightarrow B, \Delta}$ |
| Cut | $\dfrac{\Gamma \vdash A, \Delta \quad \Gamma, A \vdash \Delta}{\Gamma \vdash \Delta}$ | |

# Peirce's Formula

- A sequent calculus proof of Peirce's formula $((p \Rightarrow q) \Rightarrow p) \Rightarrow p$ is given by

$$
\dfrac{\dfrac{\dfrac{\overline{p \vdash p, q}\;Ax}{\vdash p, p \Rightarrow q}\;\vdash\Rightarrow \qquad \overline{p \vdash p}\;Ax}{(p \Rightarrow q) \Rightarrow p \vdash p}\;\Rightarrow\vdash}{\vdash ((p \Rightarrow q) \Rightarrow p) \Rightarrow p}\;\vdash\Rightarrow
$$

- The sequent formula that is introduced in the conclusion is the *principal* formula, and its components in the premise(s) are *side* formulas.

- Metatheorems about proof systems are useful in providing reasoning short-cuts.
- The deduction theorem for $H$ and the normalization theorem for $ND$ are examples.
- Prove that the Cut rule is admissible for the $LK$. (Difficult!)
- A bi-implication is a formula of the form $A \iff B$, and it is an equivalence when it is valid. Show that the following is a derived inference rule.

$$\frac{A \iff B}{C[p \mapsto A] \iff C[p \mapsto B]}$$

- State a similar rule for implication where

$$\frac{A \Rightarrow B}{C[p \mapsto A] \Rightarrow C[p \mapsto B]}$$

# Normal Forms for Formulas

- A formula where negation is applied only to propositional atoms is said to be in negation normal form (NNF).
- For example, $\neg(p \vee \neg q)$ can be represented as $\neg p \wedge q$.
- Show that every propositional formula built using $\neg$, $\vee$, and $\wedge$ is equivalent to one in NNF.
- A *literal* $l$ is either a propositional atom $p$ or its negation $\neg p$.
- A *clause* is a multiary disjunction of a set of literals $l_1 \vee \ldots \vee l_n$.
- A multiary conjunction of $n$ formulas $A_1, \ldots, A_n$ is $\bigwedge_{i=1}^{n} A_i$.

# Conjunctive and Disjunctive Normal Forms

- A formula that is a multiary conjunction of multiary disjunctions of literals is in conjunctive normal form (CNF).
- CNF Example:
$$\begin{array}{cl} & (\neg p \vee q \vee \neg r) \\ \wedge & (p \vee r) \\ \wedge & (\neg p \vee \neg q \vee r) \end{array}$$
- Define an algorithm for converting any propositional formula to CNF.
- A formula is in $k$-CNF if it uses at most $k$ literals per clause. Define an algorithm for converting any formula to 3-CNF.
- A formula that is a multiary disjunction of multiary conjunctions of literals is in disjunctive normal form (DNF).
- Define an algorithm for converting any formula to DNF.

# Soundness

- A proof system is *sound* if all provable formulas are valid, i.e., $\vdash A$ implies $\models A$, i.e., $M \models A$ for all $M$.
- To prove soundness, show that for any inference rule of the form
$$\frac{\vdash P_1, \ldots, \vdash P_n}{\vdash C},$$
any model of all of the premises is also a model of the conclusion.
- Since the axioms are valid, and each step preserves validity, we have that the conclusion of a proof is also valid.
- Demonstrate the soundness of the proof systems shown so far, i.e.,
  1. Hilbert system $H$
  2. Natural deduction $ND$
  3. Sequent Calculus $LK$

# Completeness

- A proof system is *complete* if all valid formulas are provable, i.e., $\models A$ implies $\vdash A$.
- A countermodel $M$ of $\Gamma \vdash \Delta$ is one where either $M \models A$ for all $A$ in $\Gamma$, and $M \models \neg B$ for all $B \in \Delta$.
- In $LK$, any countermodel of some premise of a rule is also a countermodel for the conclusion.
- We can then show that a non-provable sequent $\Gamma \vdash \Delta$ has a countermodel.
- Each non-Cut rule has premises that are simpler than its conclusion.
- By applying the rules starting from $\Gamma \vdash \Delta$ to completion, you end up with a set of premise sequents $\{\Gamma_1 \vdash \Delta_1, \ldots, \Gamma_n \vdash \Delta_n\}$ that are *atomic*, i.e., that contain no connectives.
- If an atomic sequent $\Gamma_i \vdash \Delta_i$ is unprovable, then it has a countermodel, i.e., one in which each formula in $\Gamma_i$ holds but no formula in $\Delta_i$ holds.
- Hence, $\Gamma \vdash \Delta$ has a countermodel.

# Completeness, More Generally

- A set of formulas $\Gamma$ is *consistent*, i.e., $Con(\Gamma)$ iff there is no formula $A$ in $\Gamma$ such that $\Gamma \vdash \neg A$ is provable.
- If $\Gamma$ is consistent, then $\Gamma \cup \{A\}$ is consistent iff $\Gamma \vdash \neg A$ is not provable.
- If $\Gamma$ is consistent, then at least one of $\Gamma \cup \{A\}$ or $\Gamma \cup \{\neg A\}$ must be consistent.
- A set of formulas $\Gamma$ is *complete* if for each formula $A$, it contains $A$ or $\neg A$.

# Completeness

- Any consistent set of formulas $\Gamma$ can be made complete as $\hat{\Gamma}$.
- Let $A_i$ be the $i$'th formula in some enumeration of $PL$ formulas. Define

$$
\begin{array}{rcl}
\Gamma_0 & = & \Gamma \\
\Gamma_{i+1} & = & \Gamma_i \cup \{A_i\}, \text{ if } Con(\Gamma_i \cup \{A_i\}) \\
 & = & \Gamma_i \cup \{\neg A_i\}, \text{ otherwise.} \\
\hat{\Gamma} & = & \Gamma_\omega = \bigcup_i \Gamma_i
\end{array}
$$

- Ex: Check that $\hat{\Gamma}$ yields an interpretation $\mathcal{M}_{\hat{\Gamma}}$ satisfying $\Gamma$.
- Is it enough to just enumerate as $A_i$, the propositional variables in $\Gamma$?
- If $\Gamma \vdash \Delta$ is unprovable, then $\Gamma \cup \overline{\Delta}$ is consistent, and has a model.

# Compactness

- A logic is *compact* if any set of sentences $\Gamma$ is satisfiable iff all finite subsets of it are, i.e., if it is *finitely satisfiable*.
- Propositional logic is compact — hard direction is showing that every finitely satisfiable set is satisfiable.
- Zorn's lemma states that if in a partially ordered set $A$, every chain $L$ has an upper bound $\widehat{L}$ in $A$, then $A$ has a maximal element.
- Given a finitely satisfiable set $\Gamma$, the set of finitely satisfiable extensions satisfies the conditions of Zorn's lemma.
- Hence there is a maximal extension $\widehat{\Gamma}$ that is finitely satisfiable.
- For any atom $p$, exactly $p \in \widehat{\Gamma}$ or $\neg p \in \widehat{\Gamma}$. Why?
- We can similarly define the model $M_{\widehat{\Gamma}}$ to show that $\widehat{\Gamma}$ is satisfiable.

# Interpolation

Craig's interpolation property states that given two sets of formulas $\Gamma_1$ and $\Gamma_2$ in propositional variables $\Sigma_1$ and $\Sigma_2$, respectively, $\Gamma_1 \cup \Gamma_2$ is unsatisfiable iff there is a formula $A$ in propositional variables $\Sigma_1 \cap \Sigma_2$ such that $\Gamma_1 \models A$ and $\Gamma_2, A$ is unsatisfiable.

| | |
|---|---|
| $Ax_1$ | $\dfrac{}{[\bot] \vdash \Gamma, P, \overline{P}; \Delta}$ |
| $Ax_2$ | $\dfrac{}{[\top] \vdash \Gamma; P, \overline{P}, \Delta}$ |
| $Ax_3$ | $\dfrac{}{[P] \vdash \Gamma, P; \overline{P}, \Delta}$ |
| $\neg\neg$ | $\dfrac{[I] \vdash P, \Delta}{[I] \vdash \neg\neg P, \Delta}$ |
| $\vee$ | $\dfrac{[I] \vdash A, B, \Delta}{[I] \vdash A \vee B, \Delta}$ |
| $\neg\vee_1$ | $\dfrac{[I_1] \vdash \Gamma, \neg A; \Delta \quad [I_2] \vdash \Gamma, \neg B; \Delta}{[[I_1 \vee I_2] \vdash \Gamma, \neg(A \vee B); \Delta}$ |
| $\neg\vee_2$ | $\dfrac{[I_1] \vdash \Gamma; \neg A, \Delta \quad [I_2] \vdash \Gamma; \neg B, \Delta}{[I_1 \wedge I_2] \vdash \Gamma; \neg(A \vee B), \Delta}$ |

# Resolution

- We have already seen that any propositional formula can be written in CNF as a conjunction of clauses.
- Input $K$ is a set of clauses.
- Tautologies, i.e., clauses containing both $l$ and $\bar{l}$, are deleted from initial input.

| | |
|---|---|
| **Res** | $\dfrac{K, l \vee \Gamma_1, \bar{l} \vee \Gamma_2}{K, l \vee \Gamma_1, \bar{l} \vee \Gamma_2, \Gamma_1 \vee \Gamma_2}$ $\Gamma_1 \vee \Gamma_2 \notin K$, $\Gamma_1 \vee \Gamma_2$ is not tautological |
| **Contrad** | $\dfrac{K, l, \bar{l}}{\bot}$ |

# Resolution: Example

$$\dfrac{\dfrac{\dfrac{\dfrac{(K_0 =)\ \neg p \vee \neg q \vee r, \quad \neg p \vee q, \quad p \vee r, \quad \neg r}{(K_1 =)\ \neg q \vee r, \quad K_0}\ \textbf{Res}}{(K_2 =)\ q \vee r, \quad K_1}\ \textbf{Res}}{(K_3 =)\ r, \quad K_2}\ \textbf{Res}}{\bot}\ \textbf{Contrad}$$

Show that resolution is a sound and complete procedure for checking satisfiability.

# CDCL Informally

- Goal: Does a given set of clauses $K$ have a satisfying assignment?
- If $M$ is a total assignment such that $M \models \Gamma$ for each $\Gamma \in K$, then $M \models K$.
- If $M$ is a partial assignment at level $h$, then *propagation* extends $M$ at level $h$ with the *implied literals* $l$ such that $l \vee \Gamma \in K \cup C$ and $M \models \neg\Gamma$.
- If $M$ detects a conflict, i.e., a clause $\Gamma \in K \cup C$ such that $M \models \neg\Gamma$, then the conflict is *analyzed* to construct a conflict clause that allows the search to be continued from a prior level.
- If $M$ cannot be extended at level $h$ and no conflict is detected, then an unassigned literal $l$ is *selected* and assigned at level $h + 1$ where the search is continued.

# Conflict-Driven Clause Learning (CDCL) SAT

| Name | Rule | Condition |
|---|---|---|
| Propagate | $\dfrac{h, \langle M \rangle, K, C}{h, \langle M, l[\Gamma] \rangle, K, C}$ | $\Gamma \equiv l \vee \Gamma' \in K \cup C$<br>$M \models \neg\Gamma'$ |
| Select | $\dfrac{h, \langle M \rangle, K, C}{h+1, \langle M; l[] \rangle, K, C}$ | $M \not\models l$<br>$M \not\models \neg l$ |
| Conflict | $\dfrac{0, \langle M \rangle, K, C}{\bot}$ | $M \models \neg\Gamma$<br>for some $\Gamma \in K \cup C$ |
| Backjump | $\dfrac{h+1, \langle M \rangle, K, C}{h', \langle M_{\leq h'}, l[\Gamma'] \rangle, K, C \cup \{\Gamma'\}}$ | $M \models \neg\Gamma$<br>for some $\Gamma \in K \cup C$<br>$\langle h', \Gamma' \rangle$<br>$= analyze(\psi)(\Gamma)$<br>for $\psi = h, \langle M \rangle, K, C$ |

# CDCL Example

- Let $K$ be $\{p \vee q, \neg p \vee q, p \vee \neg q, s \vee \neg p \vee q, \neg s \vee p \vee \neg q, \neg p \vee r, \neg q \vee \neg r\}$.
- 

| step | $h$ | $M$ | $K$ | $C$ | $\Gamma$ |
|---|---|---|---|---|---|
| select $s$ | 1 | $; s$ | $K$ | $\emptyset$ | $\_$ |
| select $r$ | 2 | $; s; r$ | $K$ | $\emptyset$ | $\_$ |
| propagate | 2 | $; s; r, \neg q[\neg q \vee \neg r]$ | $K$ | $\emptyset$ | $\_$ |
| propagate | 2 | $; s; r, \neg q, p[p \vee q]$ | $K$ | $\emptyset$ | $\_$ |
| conflict | 2 | $; s; r, \neg q, p$ | $K$ | $\emptyset$ | $\neg p \vee q$ |

# CDCL Example (contd.)

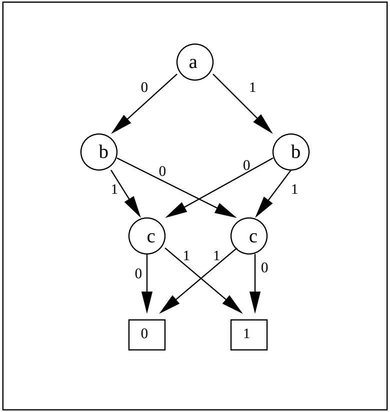
| step | $h$ | $M$ | $K$ | $C$ | $\Gamma$ |
|---|---|---|---|---|---|
| conflict | 2 | $; s; r, \neg q, p$ | $K$ | $\emptyset$ | $\neg p \vee q$ |
| backjump | 0 | $\emptyset$ | $K$ | $q$ | _ |
| propagate | 0 | $q[q]$ | $K$ | $q$ | _ |
| propagate | 0 | $q, p[p \vee \neg q]$ | $K$ | $q$ | _ |
| propagate | 0 | $q, p, r[\neg p \vee r]$ | $K$ | $q$ | _ |
| conflict | 0 | $q, p, r$ | $K$ | $q$ | $\neg q \vee \neg r$ |

Show that CDCL is sound and complete.

# ROBDD

- Boolean functions map $\{0,1\}^n$ to $\{0,1\}$.
- We have already seen how $n$-ary Boolean functions can be represented by propositional formulas of $n$ variables.
- ROBDDs are a canonical representation of boolean functions as a decision diagram where
  1. Literals are uniformly ordered along every branch: $f(x_1,\ldots,x_n) = \mathtt{IF}(x_1, f(\top, x_2,\ldots,x_n), f(\bot, x_2,\ldots,x_n))$
  2. Common subterms are identified
  3. Redundant branches are removed: $\mathtt{IF}(x_i, A, A) = A$
- Efficient implementation of boolean operations: $f_1.f_2$, $f_1 + f_2$, $-f$, including quantification.
- Canonical form yields free equivalence checks (for convergence of fixed points).

ROBDD for even parity boolean function of $a$, $b$, $c$.




Construct an algorithm to compute $f_1 \odot f_2$, where $\odot$ is $\wedge$ or $\vee$.
Construct an algorithm to compute $\exists \overline{x}.f$.

**initially**
$\mathtt{try}[1] = \mathtt{critical}[1] = \mathtt{turn} = \mathtt{false}$
**transition**

$$\begin{array}{rcl}
\neg\mathtt{try}[1] & \rightarrow & \mathtt{try}[1] := \mathtt{true}; \\
 & & \mathtt{turn} := \mathtt{false}; \\
\neg\mathtt{try}[2] \vee \mathtt{turn} & \rightarrow & \mathtt{critical}[1] := \mathtt{true}; \\
\mathtt{critical}[1] & \rightarrow & \mathtt{critical}[1] := \mathtt{false}; \\
 & & \mathtt{try}[1] := \mathtt{false};
\end{array}$$

$\|$

**initially**
$\mathtt{try}[2] = \mathtt{critical}[2] = \mathtt{false}$
**transition**

$$\begin{array}{rcl}
\neg\mathtt{try}[2] & \rightarrow & \mathtt{try}[2] := \mathtt{true}; \\
 & & \mathtt{turn} := \mathtt{true}; \\
\neg\mathtt{try}[1] \vee \neg\mathtt{turn} & \rightarrow & \mathtt{critical}[2] := \mathtt{true}; \\
\mathtt{critical}[2] & \rightarrow & \mathtt{critical}[2] := \mathtt{false}; \\
 & & \mathtt{try}[2] := \mathtt{false};
\end{array}$$

# Model Checking Transition Systems

- A transition system is given as a triple $\langle W, I, N\rangle$ of states $W$, an initialization predicate $I$, and a transition relation $N$.
- Symbolic Model Checking: Fixpoints such as $\mu X.I \sqcup post(N)(X)$ which is the set of reachable states can be constructed as an ROBDD.
- Bounded Model Checking: $I(s_0) \wedge \bigwedge_{i=0}^{k} N(s_i, s_{i+1})$ represents the set of possible $(k+1)$-step computations and $\neg P(s_{k+1})$ represents the possible violations of state predicate $P$ at the state $s_{k+1}$.
- $k$-Induction: A variant of bounded model checking can be used to prove properties:
  - Base: Check that $P$ holds in the first $k$ states of the computation
  - Induction: If $P$ holds for any sequence of $k$ steps in a computation, it holds in the $k+1$-th state.
- Prove the mutual exclusion property by $k$-induction.

# Interpolation-Based Model Checking

- Interpolation: The unsatisfiability of the BMC query yields an interpolant $Q$ such that $I(s_0) \wedge N(s_0, s_1)$ and $\bigwedge_{i=1}^{k} N(s_i, s_{i+1}) \wedge \neg P(s_{k+1})$ are jointly unsatisfiable.
- The proof yields an interpolant $Q(s_1)$.
- Let $I'(s_0)$ be $I(s_0) \vee Q(s_0)$.
- If $I(s_0) = I'(s_0)$ then this is an invariant. Otherwise, repeat the process with $I$ replaced by $I'$.
- Prove the mutual exclusion property using interpolation-based model checking.

# Equality Logic (EL)

In the process of creeping toward first-order logic, we introduce a modest but interesting extension of propositional logic.
In addition to propositional atoms, we add a set of constants $\tau$ given by $c_0, c_1, \ldots$ and equalities $c = d$ for constants $c$ and $d$.

$$\phi \quad := \quad P \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \tau_1 = \tau_2$$

The structure $M$ now has a domain $|M|$ and maps propositional variables to $\{\top, \bot\}$ and constants to $|M|$.

$$M[\![c = d]\!] \quad = \quad \begin{cases} \top, & \text{if } M[\![c]\!] = M[\![d]\!] \\ \bot, & \text{otherwise} \end{cases}$$

# Proof Rules for Equality Logic

- | | |
  |---|---|
  | Reflexivity | $\Gamma \vdash a = a, \Delta$ |
  | Symmetry | $\dfrac{\Gamma \vdash a = b, \Delta}{\Gamma \vdash b = a, \Delta}$ |
  | Transitivity | $\dfrac{\Gamma \vdash a = b, \Delta \qquad \Gamma \vdash b = c, \Delta}{\Gamma \vdash a = c, \Delta}$ |

- Show that the above proof rules (on top of propositional logic) are sound and complete.
- Show that Equality Logic is decidable.
- Adapt the above logic to reason about a partial ordering relation $\leq$, i.e., one that is reflexive, transitive, and anti-symmetric ($x \leq y \wedge y \leq x \Rightarrow x = y$).

# Term Equality Logic (TEL)

- One further extension is to add function symbols from a signature $\Sigma$ that assigns an arity to each symbol.
- Function symbols are used to form terms $\tau$, so that constants are just 0-ary function symbols.

$$\begin{array}{rcl}\tau & := & f(\tau_1, \ldots, \tau_n), \text{ for } n \geq 0 \\ \phi & := & P \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \tau_1 = \tau_2\end{array}$$

- For an $n$-ary function $f$, $M(f)$ maps $|M|^n$ to $|M|$.

$$\begin{array}{rcl}M[\![a = b]\!] & = & M[\![a]\!] = M[\![b]\!] \\ M[\![f(a_1, \ldots, a_n)]\!] & = & (M[\![f]\!])(M[\![a_1]\!], \ldots, M[\![a_n]\!])\end{array}$$

- We need one additional proof rule.

| | |
|---|---|
| Congruence | $\dfrac{\Gamma \vdash a_1 = b_1, \Delta \ldots \Gamma \vdash a_n = b_n, \Delta}{\Gamma \vdash f(a_1, \ldots, a_n) = f(b_1, \ldots, b_n), \Delta}$ |

# Term Equality Proof Examples

Let $f^n(a)$ represent $\underbrace{f(\ldots f}_{n}(a)\ldots)$.

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{f^3(a) = f(a) \vdash f^3(a) = f(a)}\,Ax}{f^3(a) = f(a) \vdash f^4(a) = f^2(a)}\,C}{f^3(a) = f(a) \vdash f^5(a) = f^3(a)}\,C \qquad \dfrac{}{f^3(a) = f(a) \vdash f^3(a) = f(a)}\,Ax}{f^3(a) = f(a) \vdash f^5(a) = f(a)}\,T
$$

Show soundness and completeness of the above system.

# Equational Logic

- Equational Logic is a heavily used fragment of first-order logic.
- It consists of term equalities $s = t$, with proof rules
  1. Reflexivity: $\frac{}{s=s}$
  2. Symmetry: $\frac{s=t}{t=s}$
  3. Transitivity: $\frac{r=s \quad s=t}{r=t}$
  4. Congruence: $\frac{s_1=t_1,\ldots,s_n=t_n}{f(s_1,\ldots,s_n)=f(t_1,\ldots,t_n)}$
  5. Instantiation: $\frac{s=t}{\sigma(s)=\sigma(t)}$, for substitution $\sigma$.
- We say $\Gamma \vdash s = t$ when the equality $s = t$ can be derived from the equalities in $\Gamma$.
- Show that equational logic is sound and complete.

# Equational Logic

## Use equational logic to formalize

1. Semigroups: A set $G$ with an associative binary operator .
2. Monoids: A set $M$ with associative binary operator . and unit 1
3. Groups: A monoid with an right-inverse operator $x^{-1}$
4. Commutative groups and semigroups
5. Rings: A set $R$ with commutative group $\langle R, +, -, 0\rangle$, semigroup $\langle R, .\rangle$, and distributive laws $x.(y+z) = x.y + x.z$ and $(y+z).x = y.x + z.x$
6. Semilattice: A commutative semigroup $\langle S, \wedge\rangle$ with idempotence $x \wedge x = x$
7. Lattice: $\langle L, \wedge, \vee\rangle$ where $\langle L, \wedge\rangle$ and $\langle L, \vee\rangle$ are semilattices, and $x \vee (x \wedge y) = x$ and $x \wedge (x \vee y) = x$.
8. Distributive lattice: A lattice with $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$.
9. Boolean algebra: Distributive lattice with constants 0 and 1 and unary operation $-$ such that $x \wedge 0 = 0$, $x \vee 1 = 1$, $x \wedge -x = 0$, and $x \vee -x = 1$.

# Equational Logic

- Prove that every group element has a left inverse.
- For a lattice, define $x \leq y$ as $x \wedge y = x$. Show that $\leq$ is a partial order (reflexive, transitive, and antisymmetric).
- Show that a distributive lattice satisfies $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$.
- Prove the de Morgan laws, $-(x \vee y) = -x \wedge -y$ and $-(x \wedge y) = -x \vee -y$ for Boolean algebras.

# First-Order Logic

We can now complete the transition to first-order logic by adding

$$
\begin{array}{lcl}
\tau & := & X \\
 & & \mid \; f(\tau_1, \ldots, \tau_n), \text{ for } n \geq 0 \\
\phi & := & \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \tau_1 = \tau_2 \\
 & & \mid \forall x.\phi \mid \exists x.\phi \mid q(\tau_1, \ldots, \tau_n), \text{ for } n \geq 0
\end{array}
$$

Terms contain variables, and formulas contain atomic and quantified formulas.

# Semantics for Variables and Quantifiers

$M[\![q]\!]$ is a map from $D^n$ to $\{\top, \bot\}$, where $n$ is the arity of predicate $q$.

$$\begin{array}{rcl}
M[\![x]\!]\rho & = & \rho(x) \\
M[\![q(a_1, \ldots, a_n)]\!]\rho & = & M[\![q]\!](M[\![a_1]\!]\rho, \ldots, M[\![a_n]\!]\rho) \\
M[\![\forall x.A]\!]\rho & = & \begin{cases} \top, & \text{if } M[\![A]\!]\rho[x := d] \text{ for all } d \in D \\ \bot, & \text{otherwise} \end{cases} \\
M[\![\exists x.A]\!]\rho & = & \begin{cases} \top, & \text{if } M[\![A]\!]\rho[x := d] \text{ for some } d \in D \\ \bot, & \text{otherwise} \end{cases}
\end{array}$$

Atomic formulas are either equalities or of the form $q(a_1, \ldots, a_n)$.

# First-Order Logic

- | | Left | Right |
  |---|---|---|
  | $\forall$ | $\dfrac{\Gamma, A[t/x] \vdash \Delta}{\Gamma, \forall x.A \vdash \Delta}$ | $\dfrac{\Gamma \vdash A[c/x], \Delta}{\Gamma \vdash \forall x.A, \Delta}$ |
  | $\exists$ | $\dfrac{\Gamma, A[c/x] \vdash \Delta}{\Gamma, \exists x.A \vdash \Delta}$ | $\dfrac{\Gamma \vdash A[t/x], \Delta}{\Gamma \vdash \exists x.A, \Delta}$ |

- Constant $c$ must be chosen to be new so that it does not appear in the conclusion sequent.
- Demonstrate the soundness of first-order logic.
- A theory consists of a signature $\Sigma$ for the function and predicate symbols and non-logical axioms.
- If a $T$ is obtained from $S$ by extending the signature and adding axioms, then $T$ is conservative with respect to $S$, if all the formulas in $S$ provable in $T$ are also provable in $S$.

# Using First-Order Logic

- Prove $\exists x.(p(x) \Rightarrow \forall y.p(y))$.
- Give at least two satisfying interpretations for the statement $(\exists x.p(x)) \implies (\forall x.p(x))$.
- A sentence is a formula with no free variables. Find a sentence $A$ such that both $A$ and $\neg A$ are satisfiable.
- Write a formula asserting the unique existence of an $x$ such that $p(x)$.
- Define operations for collecting the free variables $vars(A)$ in a given formula $A$, and substituting a term $a$ for a free variable $x$ in a formula $A$ to get $A\{x \mapsto a\}$.
- Is $M[\![A\{x \mapsto a\}]\!]\rho = M[\![A]\!]\rho[x := M[\![a]\!]\rho]$? If not, show an example where it fails. Under what condition does the equality hold?
- Show that any quantified formula is equivalent to one in *prenex normal form*, i.e., where the only quantifiers appear at the head of the formula and the body is purely a propositional combination of atomic formulas.

# More Exercises

- Prove
  1. $\neg\forall x.A \iff \exists x.\neg A$
  2. $(\forall x.A \wedge B) \iff (\forall x.A) \wedge (\forall x.B)$
  3. $(\exists x.A \vee B) \iff (\exists x.A) \vee (\exists x.B)$
  4. $((\forall x.A) \vee (\forall x.B)) \Rightarrow (\forall x.A \vee B)$
- Write the axioms for a partially ordered relation $\leq$.
- Write the axioms for a bijective (1-to-1, onto) function $f$.
- Write a formula asserting that for any $x$, there is a unique $y$ such that $p(x, y)$.
- Can you write first-order formulas whose models
  1. Have exactly (at most, at least) three elements?
  2. Are infinite
  3. Are finite but unbounded
- Can you write a first-order formula asserting that
  1. A relation is transitively closed
  2. A relation is the transitive closure of another relation.

- The quantifier rules for sequent calculus require copying.
- Proof branches can be extended without bound.
- Ex: Show that $LK$ is sound: $\vdash A$ implies $\models A$.
- The Henkin closure $H(\Gamma)$ is the smallest extension of a set of sentences $\Gamma$ that is Henkin-closed, i.e., contains $B \Rightarrow A(c_B)$ for every $B \in H(\Gamma)$ of the form $\exists x : A$. ($c_B$ is a fresh constant.)
- Any consistent set of formulas $\Gamma$ has a *consistent* Henkin closure $H(\Gamma)$.
- As before, any consistent, Henkin closed set of formulas $\Gamma$ has a complete, Henkin-closed extension $\hat{\Gamma}$.
- Ex: Construct an interpretation $M_{\widehat{H(\Gamma)}}$ from $\widehat{H(\Gamma)}$ and show that it is a model for $\Gamma$.

# Herbrand's Theorem

- For any sentence $A$ there is a quantifier-free sentence $A_H$ (the Herbrand form of $A$) such that $\vdash A$ in $LK$ iff $\vdash A_H$ in $TEL_0$.
- The Herbrand form is a *dual* of Skolemization where each universal quantifier is replaced by a term $f(\overline{y})$, where $\overline{y}$ is the set of governing existentially quantified variables.
- Then, $\exists x : (p(x) \Rightarrow \forall y : p(y))$ has the Herbrand form $\exists x.p(x) \Rightarrow p(f(x))$, and the two formulas are equi-valid.
- How do you prove the latter formula?

# Herbrand's Theorem

- Herbrand terms are those built from function symbols in $A_H$ (adding a constant, if needed).
- Show that if $A_H$ is of the form $\exists \overline{x}.B$, then $\vdash A_H$ iff $\bigvee_{i=0}^{n} \sigma_i(B)$, for some Herbrand term substitutions $\sigma_1, \ldots, \sigma_n$.
- [Hint: In a cut-free sequent proof of a prenex formula, the quantifier rules can be made to appear below all the other rules. Such proofs must have a quantifier-free mid-sequent above which the proof is entirely equational/propositional.]
- Show that if a formula has a counter-model, then it has one built from Herbrand terms (with an added constant if there isn't one).

# Skolemization

- Consider a formula of the form $\forall x.\exists y.q(x, y)$.
- It is equisatisfiable with the formula $\forall x.q(x, f(x))$ for a new function symbol $f$.
- If $M \models \forall x.\exists y.q(x, y)$, then for any $c \in |M|$, there is $d_c \in |M|$ such that $M[\![q(x, y)]\!]\{x \mapsto c, y \mapsto d_c\}$. let $M'$ extend $M$ so that $M(f)(c) = d_c$, for each $c \in |M|$: $M' \models \forall x.q(x, f(y))$.
- Conversely, if $M \models \forall x.q(x, f(y))$, then for every $c \in |M|$, $M[\![q(x, y)]\!]\{x \mapsto c, y \mapsto M(f)(c)\}$.
- Prove the general case that any prenex formula can be Skolemized by replacing each existentially quantified variable $y$ by a term $f(\overline{x})$, where $f$ is a distinct, new function symbol for each $y$, and $\overline{x}$ are the universally quantified variables *governing* $y$.

# Unification

- A substitution is a map $\{x_1 \mapsto a_1, \ldots, x_n \mapsto a_n\}$ from a finite set of variables $\{x_1, \ldots, x_n\}$ to a set of terms.
- Define the operation $\sigma(a)$ of applying a substitution (such as the one above) to a term $a$ to replace any free variables $x_i$ in $t$ with $a_i$.
- Define the operation of composing two substitutions $\sigma_1 \circ \sigma_2$ as $\{x_1 \mapsto \sigma_1(a_1), \ldots, x_n \mapsto \sigma_1(a_n)\}$, if $\sigma_2$ is of the form $\{x_1 \mapsto a_1, \ldots, x_n \mapsto a_n\}$.
- Given two terms $f(x, g(y, y))$ and $f(g(y, y), x)$ (possibly containing free variables), find a substitution $\sigma$ such that $\sigma(a) \equiv \sigma(b)$.
- Such a $\sigma$ is called a unifier.
- Not all terms have such unifiers, e.g., $f(g(x))$ and $f(x)$.
- A substitution $\sigma_1$ is more general than $\sigma_2$ if the latter can be obtained as $\sigma \circ \sigma_1$, for some $\sigma$.
- Define the operation of computing the most general unifier, if there is one, and reporting failure, otherwise.

# Resolution Example

- To prove $(\exists y.\forall x.p(x,y)) \Rightarrow (\forall x.\exists y.p(x,y))$
- Negate: $(\exists y.\forall x.p(x,y)) \wedge (\exists x.\forall y.\neg p(x,y))$
- Prenexify: $\exists y_1.\forall x_1.\exists x_2.\forall y_2.p(x_1,y_1) \wedge \neg p(x_2,y_2)$
- Skolemize: $\forall x_1, y_2.p(x_1,c) \wedge \neg p(f(x_1),y_2)$
- Distribute and clausify: $\{p(x_1,c), \neg p(f(x_3),y_2)\}$
- Unify and resolve with unifier $\{x_1 \mapsto f(x_3), y_2 \mapsto c\}$
- Yields an empty clause
- Now try to show $(\forall x.\exists y.p(x,y)) \Rightarrow (\exists y.\forall x.p(x,y))$.

- The natural numbers consist of $0, s(0), s(s(0))$, etc.
- Clearly, $0 \neq s(x)$, for any $x$.
- Also, $s(x) = s(y) \Rightarrow x = y$, for any $x$ and $y$.
- Next, we would like to say that this is all there is, i.e., every domain element is reachable from 0 through applications of $s$.
- This requires induction:
  $P(0) \wedge (\forall n. P(n) \Rightarrow P(n+1)) \Rightarrow (\forall n. P(n))$, for every property $P$.
- But there is no way to write this — there are uncountably many properties (subset of natural numbers) but only finitely many formulas.
- Induction is therefore given as a scheme, an infinite set of axioms, with the template

$$A\{x \mapsto 0\} \wedge (\forall x. A \Rightarrow A\{x \mapsto s(x)\}) \Rightarrow (\forall x. A).$$

- We still need to define $+$ and $\times$. How?
- How do you define the relations $x < y$ and $x \leq y$?

- Prove that
  1. $\forall x. x = 0 \vee (\exists y. s(y) = x)$
  2. $\forall x, y, z. (x + y) + z = x + (y + z)$
  3. $\forall x, y. x + y = y + x$
  4. $\forall x, y. x < y \implies \neg(y < x)$

# Set Theory

Set theory can be axiomatized using axiom schemes, using a membership relation $\in$:

- Extensionality: $x = y \iff (\forall z. z \in x \iff z \in y)$
- The existence of the empty set $\forall x. \neg x \in \emptyset$
- Pairs: $\forall x, y. \exists z. \forall u. u \in z. \iff u = x \vee u = y$ (Define the singleton set containing the empty set. Construct a representation for the ordered pair of two sets.)
- Union: How? (Define a representation for the finite ordinals using singleton, or using singleton and union.)
- Separation: $\{x \in y | A\}$, for any formula $A$, $y \notin vars(A)$. (Define the intersection and disjointness of two sets.)
- Infinity: There is a set containing all the finite ordinals.
- Power set: For any set, we have the set of all its subsets.
- Regularity: Every set has an element that is disjoint from it.
- Replacement: There is a set that is a superset of the image $Y$ of a set $X$ with respect to a functional $(\forall x \in X. \exists! y. A(x, y))$ rule $A(x, y)$.

# Using Set Theory

- Can two different sets be empty?
- For your definition of ordered pairing, define the first and second projection operations.
- Define the Cartesian product $x \times y$ of two sets, as the set of ordered pairs $\langle u, v \rangle$ such that $u \in x$ and $v \in y$.
- Define a subset of $x \times y$ to be functional if it does not contain any ordered pairs $\langle u, v \rangle$ and $\langle u, v' \rangle$ such that $v \neq v'$.
- Define the function space $y^x$ of the functions that map elements of $x$ to elements of $y$.
- Define the join of two relations, where the first is a subset of $x \times y$ and the second is a subset of $y \times z$.

# Incompleteness

- Can all mathematical truths (valid sentences) be formally proved?
- *No.* There are valid statements about numbers that have no proof. (Gödel's first incompleteness theorem)
- Suppose $Z$ is some formal theory claiming to be a sound and complete formalization of arithmetic, i.e., it proves all and only valid statements about numbers.
- Gödel showed that there is a valid but unprovable statement.

# The First Incompleteness Theorem

- The expressions of $Z$ can be represented as numbers as can the proofs.
- The statement "$p$ is a proof of $A$" can then be represented by a formula $Pf(x, y)$ about numbers $x$ and $y$.
- If $p$ is represented by the number $\underline{p}$ and $A$ by $\underline{A}$, then $Pf(\underline{p}, \underline{A})$ is provable iff $p$ is a proof of $A$.
- Numbers such as $\underline{A}$ are representable as numerals in $Z$ and these numerals can also be represented by numbers, $\underline{\underline{A}}$.
- Then $\exists x.Pf(x, y)$ says that the statement represented by $y$ is *provable*. Call this $Pr(y)$.

# The Undecidable Sentence

- Let $S(x)$ represent the numeric encoding of the operation such that for any number $k$, $S(k)$ is the encoding of the expression obtained by substituting the numeral for $k$ for the variable '$x$' in the expression represented by the number $k$.
- Then $\neg Pr(S(x))$ is represented by a number $k$, and the undecidable sentence $U$ is $\neg Pr(S(k))$.
- $\underline{U}$ is $S(k)$, i.e., the sentence obtained by substituting the numeral for $k$ for '$x$' in $\neg Pr(S(x))$ which is represented by $k$.
- Since $U$ is $\neg Pr(\underline{U})$, we have a situation where either
  1. $U$, i.e., $\neg Pr(\underline{U})$, is provable, but from the numbering of the proof of $U$, we can also prove $Pr(\underline{U})$.
  2. $\neg U$, i.e., $Pr(\underline{U})$ is provable, but clearly none of $Pf(0, \underline{U})$ $Pf(1, \underline{U})$, $\dots$, is provable (since otherwise $U$ would be provable), an $\omega$-inconsistency, or
  3. Neither $U$ nor $\neg U$ is provable: an incompleteness.

- The negation of the sentence $U$ is $\Sigma_1$, and $Z$ can verify $\Sigma_1$-completeness (*every valid $\Sigma_1$-sentence is provable*).
- Then
$$\vdash Pr(\underline{U}) \Rightarrow Pr(\underline{Pr(\underline{U})}).$$
- But this says $\vdash Pr(\underline{U}) \Rightarrow Pr(\underline{\neg U})$.
- Therefore $\vdash Con(Z) \Rightarrow \neg Pr(\underline{U})$.
- Hence $\neg \vdash Con(Z)$, by the first incompleteness theorem.
- **Exercise:** The theory $Z$ is consistent if $A \wedge \neg A$ is not provable for any $A$. Show that $\omega$-consistency is stronger than consistency. Show that the consistency of $Z$ is adequate for proving the first incompleteness theorem.

# Higher-Order Logic

- Thus far, variables ranged over ordinary datatypes such as numbers, and the functions and predicates were fixed (constants).
- Second-order logic allows free and bound variables to range over the functions and predicates of first-order logic.
- In $n$'th-order logic, the arguments (and results) of functions and predicates are the functions and predicates of $m$'th-order logic for $m < n$.
- This kind of strong typing is required for consistency, otherwise, we could define $R(x) = \neg x(x)$, and derive $R(R) = \neg R(R)$.
- Higher-order logic, which includes $n$'th-order logic for any $n > 0$, can express a number of interesting concepts and datatypes that are not expressible within first-order logic: transitive closure, fixpoints, finiteness, etc.

# Types in Higher-Order Logic

- Base types: e.g., `bool`, `nat`, `real`
- Tuple types: $[T_1, \ldots, T_n]$ for types $T_1$, ..., $T_n$.
- Tuple terms: $(a_1, \ldots, a_n)$
- Projections: $\pi_i(a)$
- Function types: $[T_1 \rightarrow T_2]$ for domain type $T_1$ and range type $T_2$.
- Lambda abstraction: $\lambda(x : T_1) : a$
- Function application: $f\ a$.

# Semantics of Higher Order Types

$$
\begin{array}{rcl}
[\![\mathtt{bool}]\!] & = & \{0, 1\} \\
[\![\mathtt{real}]\!] & = & \mathbf{R} \\
[\![[T_1, \ldots, T_n]]\!] & = & [\![T_1]\!] \times \ldots \times [\![T_n]\!] \\
[\![[T_1 \rightarrow T_2]]\!] & = & [\![T_2]\!]^{[\![T_1]\!]}
\end{array}
$$

# Higher-Order Proof Rules

| | |
|---|---|
| $\beta$-reduction | $\dfrac{}{\Gamma \vdash (\lambda(x : T) : a)(b) = a[b/x], \Delta}$ |
| Extensionality | $\dfrac{\Gamma \vdash (\forall(x : T) : f(x) = g(x)), \Delta}{\Gamma \vdash f = g, \Delta}$ |
| Projection | $\dfrac{}{\Gamma \vdash \pi_i(a_1, \ldots, a_n) = a_i, \Delta}$ |
| Tuple Ext. | $\dfrac{\Gamma \vdash \pi_1(a) = \pi_1(b), \Delta, \ldots, \Gamma \vdash \pi_n(a) = \pi_i(b), \Delta}{\Gamma \vdash a = b, \Delta}$ |

# Using Higher-Order Logic

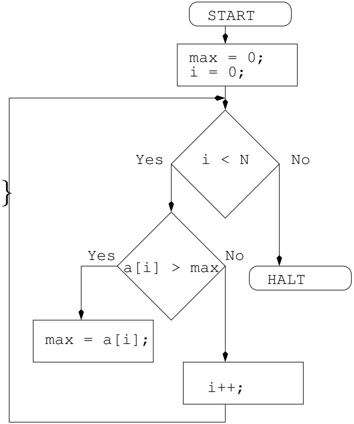
- Define universal quantification using equality in higher-order logic.
- Express and prove Cantor's theorem (there is no injection from a type $T$ to a $[T \to bool]$) in higher-order logic.
- Write the induction principle for Peano arithmetic in higher-order logic.
- Write a definition for the transitive closure of a relation in higher-order logic.
- Describe the modal logic CTL in higher-order logic.
- State and prove the Knaster-Tarski theorem.

# Floyd's method for Flowchart programs

- A flowchart has a *start* vertex with a single outgoing edge, a *halt* vertex with a single incoming edge.
- Each vertex corresponds to a program block or a decision conditions.
- Each edge corresponds to an assertion; the start edge is the flowchart *precondition*, and the halt edge is the flowchart *postcondition*.
- *Verification conditions* check that for each vertex, each incoming edge assertion through the block implies the outgoing edge assertion.
- *Partial correctness*: If each verification condition has been discharged, then every halting computation starting in a state satisfying the precondition terminates in a state satisfying the postcondition.
- *Total correctness*: If there is a ranking function mapping states to ordinals that strictly decreases for any cycle in the flowchart, then every computation terminates in the halt

# Floyd's Method

```
max = 0;
i = 0;
{i ≤ N ∧ ∀(j < i): a[j] ≤ max}
while (i < N){
 if (a[i] > max){
   max = a[i];
   }
 i++;
 }
{∀(j < N): a[j] ≤ max}
```




Precondition is true, and postcondition is $\forall(\mathtt{j} < \mathtt{N}) : \mathtt{a[j]} \leq \mathtt{max}$.

The *loop invariant* is $\mathtt{i} \leq \mathtt{N} \wedge \forall(\mathtt{j} < \mathtt{i}) : \mathtt{a[j]} \leq \mathtt{max}$.

- A Hoare triple has the form $\{P\}S\{Q\}$, where $S$ is a program statement in terms of the program variables drawn from the set $Y$ and $P$ and $Q$ are assertions containing logical variables from $X$ and program variables.
- A program statement is one of
  1. A *skip* statement *skip*.
  2. A *simultaneous assignment* $\overline{y} := \overline{e}$ where $\overline{y}$ is a sequence of $n$ distinct program variables, $e$ is a sequence of $n$ $\Sigma[Y]$-terms.
  3. A *conditional* statement $e \; ? \; S_1 \; : \; S_2$, where $C$ is a $\Sigma[Y]$-formula.
  4. A *loop* *while* $e$ *do* $S$.
  5. A sequential composition $S_1; S_2$.

# Hoare Logic

| | |
|---|---|
| Skip | $\{P\}skip\{P\}$ |
| Assignment | $\{P[\overline{e}/\overline{y}]\}\overline{y} := \overline{e}\{P\}$ |
| Conditional | $\dfrac{\{C \wedge P\}S_1\{Q\} \quad \{\neg C \wedge P\}S_2\{Q\}}{\{P\}C\ ?\ S_1\ :\ S_2\{Q\}}$ |
| Loop | $\dfrac{\{P \wedge C\}S\{P\}}{\{P\}while\ C\ do\ S\{P \wedge \neg C\}}$ |
| Composition | $\dfrac{\{P\}S_1\{R\} \quad \{R\}S_2\{Q\}}{\{P\}S_1; S_2\{Q\}}$ |
| Consequence | $\dfrac{P \Rightarrow P' \quad \{P'\}S\{Q'\} \quad Q' \Rightarrow Q}{\{P\}S\{Q\}}$ |

# Hoare Logic Semantics

- Both assertions and statements contain operations from a first-order signature $\Sigma$.
- An assignment $\sigma$ maps program variables in $Y$ to values in $dom(M)$.
- A program expression $e$ has value $M[\![e]\!]\sigma$.
- The meaning of a statement $M[\![S]\!]$ is given by a sequence of states (of length at least 2).
  1. $\sigma \circ \sigma \in M[\![skip]\!]$, for any state $\sigma$.
  2. $\sigma \circ \sigma[M[\![\overline{e}]\!]\sigma/\overline{y}] \in M[\![\overline{y} := \overline{e}]\!]$, for any state $\sigma$.
  3. $\psi_1 \circ \sigma \circ \psi_2 \in M[\![S_1; S_2]\!]$ for $\psi_1 \circ \sigma \in M[\![S_1]\!]$ and $\sigma \circ \psi_2 \in M[\![S_2]\!]$
  4. $\psi \in M[\![C\ ?\ S_1\ :\ S_2]\!]$ if either $M[\![C]\!]\psi[0] = \top$ and $\psi \in M[\![S_1]\!]$, or $M[\![C]\!]\psi[0] = \bot$ and $\psi \in M[\![S_2]\!]$
  5. $\sigma \circ \sigma \in M[\![while\ C\ do\ S]\!]$ if $M[\![C]\!]\sigma = \bot$
  6. $\psi_1 \circ \sigma \circ \psi_2 \in M[\![while\ C\ do\ S]\!]$ if $M[\![C]\!](\psi_1[0]) = \top$, $\psi_1 \circ \sigma \in M[\![S]\!]$, and $\sigma \circ \psi_2 \in M[\![while\ C\ do\ S]\!]$

# Soundness of Hoare Logic

- $\{P\}S\{Q\}$ is *valid* in a $\Sigma$-structure $M$ if for every sequence $\sigma \circ \psi \circ \sigma' \in M[\![S]\!]$ and any assignment $\rho$ of values in $dom(M)$ to logical variables in $X$, either
  1. $M[\![Q]\!]^{\rho}_{\sigma'} = \top$, or
  2. $M[\![P]\!]^{\rho}_{\sigma} = \bot$.
- Informally, every computation sequence for $S$ either ends in a state satisfying $Q$ or starts in a state falsifying $P$.
- Demonstrate the soundness of the Hoare calculus.

# Completeness of Hoare Logic

- The proof of a valid triple $\{P\}S\{Q\}$ can be decomposed into
  1. The valid triple $\{wlp(S)(Q)\}S\{Q\}$, and
  2. The valid assertion $P \Rightarrow wlp(S)(Q)$
- $wlp(S)(Q)$ (the *weakest liberal precondition*) is an assertion such that for any $\psi \in M[\![S]\!]$ with $|\psi| = n + 1$ and $\rho$, either $M[\![Q]\!]^{\rho}_{\psi_n} = \bot$ or $M[\![wlp(S)(Q)]\!]^{\rho}_{\psi_0} = \top$.
- Show that for any $S$ and $Q$, the valid triple $\{wlp(S)(Q)\}S\{Q\}$ can be proved in the Hoare calculus. (Hint: Use induction on $S$.)
- First-order arithmetic over $\langle +, ., 0, 1\rangle$ is sufficient to express $wlp(S)(Q)$ since it can code up sequences of states representing computations.

# Conclusions: Speak Logic!

- Logic is a powerful tool for
  1. Formalizing concepts
  2. Defining abstractions
  3. Proving validities
  4. Solving constraints
  5. Reasoning by calculation
  6. Mechanized inference
- The power of logic is when it is used as an aid to effective reasoning.
- > *Logic can become enormously difficult, and it would undoubtedly be well to produce more assurance in its use. . . . We may some day click off arguments on a machine with the same assurance that we now enter sales on a cash register.*
  >
  > *Vannevar Bush, As We May Think*
- The machinery of logic has made it possible to solve large and complex problems; formal verification is now a practical technology.

# References


- Barwise, *Handbook of Mathematical Logic*
- Johnstone, *Notes on logic and set theory*
- Ebbinghaus, Flum, and Thomas, *Mathematical Logic*
- Kees Doets, *Basic Model Theory*
- Huth and Ryan, *Logic in Computer Science: Modelling and Reasoning about Systems*
- Girard, Lafont, and Taylor, *Proofs and Types*
- Shankar, *Automated Reasoning for Verification*, ACM Computing Surveys, 2009