

Speaking Logic

N. Shankar

Computer Science Laboratory
SRI International
Menlo Park, CA

May 20, 2013

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*.
- Logic is the medium for expressing these abstract laws.
- It also supplies sound reasoning principles for deriving the consequences of these laws.
- Logic is thus 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 expressed clearly and precisely in logic.
- 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

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

- 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 complex problems through abstraction and decomposition.
- This kind of problem solving requires *using* rigorous, possibly formal, logical reasoning.
- 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.

- The course is spread over six hours:
 - **Monday:** Propositional Logics
 - **Tuesday:** First-Order Logic
 - **Wednesday:** Set Theory and Higher-Order Logic
- 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]

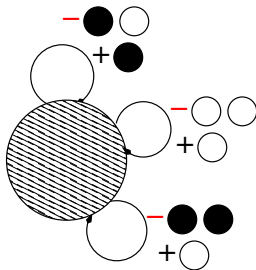
- Given four cards laid out on a table as: \boxed{D} , $\boxed{3}$, \boxed{F} , $\boxed{7}$, where each card has a letter on one side and a number on the other.
- Which cards should you flip over to determine if every card with a \boxed{D} on one side has a $\boxed{7}$ on the other side?

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?



Truth-tellers 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?

- 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 ♠, ♥, ♣, ♦:

$\langle 5♠ \rangle, \langle 3♥ \rangle, \langle Q♣ \rangle, \langle 8♦ \rangle,$
 $\langle K♠ \rangle, \langle 2♥ \rangle, \langle 7♣ \rangle, \langle 4♦ \rangle,$
 $\langle 8♠ \rangle, \langle J♥ \rangle, \langle 9♣ \rangle, \langle A♦ \rangle$

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

$\langle 2♥ \rangle, \langle 7♣ \rangle, \underline{\langle K♠ \rangle}, \underline{\langle 8♦ \rangle},$
 $\langle 4♦ \rangle, \langle 8♠ \rangle, \underline{\langle Q♣ \rangle}, \underline{\langle J♥ \rangle},$
 $\underline{\langle 3♥ \rangle}, \underline{\langle 9♣ \rangle}, \underline{\langle 5♠ \rangle}, \underline{\langle A♦ \rangle}$

- 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 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 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, \dots
- Examples of formulas are p , $p \wedge \neg p$, $p \vee \neg p$, $(p \wedge \neg q) \vee \neg p$.



Meaning

- 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 $\{\top, \perp\}$ 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:

ϕ	p	q	$\neg p$	$p \vee q$	$p \wedge q$
$M_1(\phi)$	\perp	\perp	\top	\perp	\perp
$M_2(\phi)$	\perp	\top	\top	\top	\perp
$M_3(\phi)$	\top	\perp	\perp	\top	\perp
$M_4(\phi)$	\top	\top	\perp	\top	\top

Truth Tables

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

p	q	$(\neg p \vee q)$	$(\neg(\neg p \vee q) \vee p)$	$\neg(\neg(\neg p \vee q) \vee p) \vee p$
\perp	\perp	\top	\perp	\top
\perp	\top	\top	\perp	\top
\top	\perp	\perp	\top	\top
\top	\top	\top	\top	\top

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



- 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 $\{\top, \perp\}^n$ to $\{\top, \perp\}$
- 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, \dots, p_{n-1} is even?
- Define the *NAND* operation, where $NAND(p, q)$ is $\neg(p \wedge q)$ using \neg and \vee .

Satisfiability and Validity

- An interpretation M is a model of a formula ϕ if $M \models \phi$.
- If $M \models \neg\phi$, then M is a *countermodel* for ϕ .
- When ϕ has a model, it is said to be *satisfiable*.
- If it has no model, then it is *unsatisfiable*.
- If $\neg\phi$ is unsatisfiable, then ϕ is valid, i.e., always evaluates to \top .
- We write $\phi \models \psi$ if every model of ϕ is a model of ψ .
- 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 Σ ,
 - Set of states S
 - Initial state q ,
 - Set of final states F , and
 - Transition function δ from $\langle Q, \Sigma \rangle$ to Qaccepts 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 reads the symbol at the current position of the head, and
 - 1 Writes a new symbol at the head position
 - 2 Moves the head by a step either to the left or right of the current position
 - 3 Transitions to the next state of the finite automaton
- A Turing machine is nondeterministic if the transition function computes a set of states.
- 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 tape of size n , one can (in polynomial time) construct a propositional formula which is satisfiable iff there is a terminating computation of the Turing machine on the input.

- There are three basic styles of proof systems.
- These are distinguished by their basic judgement.
 - 1 Hilbert systems: A formula is provable.
 - 2 Natural deduction: A formula is provable from a set of formulas.
 - 3 Sequent Calculus: Some consequent formula is a consequence of the antecedent formulas.

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
 - $\frac{}{\vdash A \Rightarrow A}$
 - $\frac{}{\vdash A \Rightarrow (B \Rightarrow A)}$
 - $\frac{}{\vdash (A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))}$
- A single rule of inference (Modus Ponens) is given

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

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



Hilbert System (H)

- Add a propositional constant \perp to the formula syntax, where $\llbracket \perp \rrbracket = \perp$.
- Define negation $\neg A$ as $A \Rightarrow \perp$.
- Can you prove
 - 1 $\neg A \Rightarrow (A \Rightarrow \neg B)$
 - 2 $\neg A \Rightarrow (A \Rightarrow \perp)$
 - 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 Γ , 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 Γ, A is $\Gamma \cup \{A\}$. [Hint: Use induction on proofs.]
- A *derived* rule of inference has the form

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

where there is a derivation in the base logic from the premises P_1, \dots, P_n to the conclusion C .

- An *admissible* rule of inference is one where the premises P_1, \dots, P_n must be provable for C to be 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

- $$\frac{\overline{\Gamma, A \vdash A}}{\Gamma_1 \vdash A} \quad \Gamma_2 \vdash A \Rightarrow B}{\Gamma_1 \cup \Gamma_2 \vdash B}$$
- $$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}$$

- 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.

Sequent Calculus (LK) for Propositional Logic

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

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



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

$$\frac{\frac{\frac{\overline{p \vdash p, q} \text{ Ax}}{\vdash p, p \Rightarrow q} \text{ } \Rightarrow \vdash}{\vdash p \vdash p} \text{ Ax}}{\vdash (p \Rightarrow q) \Rightarrow p \vdash p} \Rightarrow \vdash}{\vdash ((p \Rightarrow q) \Rightarrow p) \Rightarrow p} \text{ } \Rightarrow \vdash$$

- 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

- 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 $l_1 \vee \dots \vee l_n$.
- A multiary conjunction of n formulas A_1, \dots, 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:
$$\begin{aligned} & (\neg p \vee q \vee \neg r) \\ & \wedge (p \vee r) \\ & \wedge (\neg p \vee \neg q \vee r) \end{aligned}$$
- 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.



- 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

$$\frac{\vdash P_1, \dots, \vdash P_n}{\vdash C},$$

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 Γ , 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 $\{\Gamma_1 \vdash \Delta_1, \dots, \Gamma_n \vdash \Delta_n\}$ 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 Γ_i holds but no formula in Δ_i holds.
- Hence, $\Gamma \vdash \Delta$ has a countermodel.

Completeness, More Generally

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

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

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

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

- A logic is *compact* if any set of sentences Γ is satisfiable if all finite subsets of it are, i.e., if it is finitely satisfiable.
- Propositional logic is compact.
- If Γ is not satisfiable, then it is not consistent.
- The completeness argument can be adapted to a proof of compactness by replacing $Con(\Gamma)$ by the finite satisfiability of Γ .
- Then the set $\hat{\Gamma}$ yields a complete finitely satisfiable set and hence it does not contain both A and $\neg A$ for any A .
- Since $\hat{\Gamma}$ is a superset of Γ , the latter is also satisfiable.

- Craig's interpolation property states that given two sets of formulas Γ_1 and Γ_2 in propositional variables Σ_1 and Σ_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 Γ_2, A is unsatisfiable.
- An alternative statement of interpolation is that if $\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2$, then there is a formula C in the intersection of $\text{vars}(\Gamma_1 \cup \Delta_1) \cap \text{vars}(\Gamma_2 \cup \Delta_2)$ such that $\Gamma_1 \vdash C, \Delta_1$ and $\Gamma_2, C \vdash \Delta_2$.
- Show a way of annotating a sequent proofs so that each sequent has an interpolant.

- 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	$\frac{K, l \vee \Gamma_1, \bar{l} \vee \Gamma_2}{K, l \vee \Gamma_1, \bar{l} \vee \Gamma_2, \Gamma_1 \vee \Gamma_2} \quad \Gamma_1 \vee \Gamma_2 \notin K$ $\Gamma_1 \vee \Gamma_2 \text{ is not tautological}$
Contrad	$\frac{K, l, \bar{l}}{\perp}$

Resolution: Example

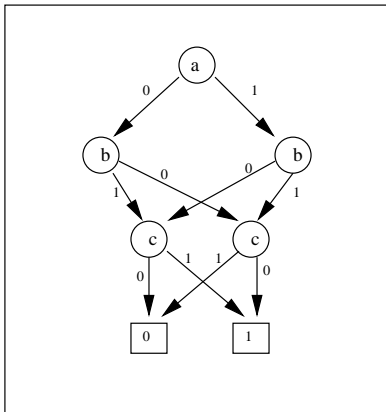
$$\begin{array}{l} (K_0 =) \neg p \vee \neg q \vee r, \neg p \vee q, p \vee r, \neg r \\ \hline (K_1 =) \neg q \vee r, K_0 \\ \hline (K_2 =) q \vee r, K_1 \\ \hline (K_3 =) r, K_2 \\ \hline \perp \end{array} \begin{array}{l} \text{Res} \\ \text{Res} \\ \text{Res} \\ \text{Contrad} \end{array}$$

- 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:

$$f(x_1, \dots, x_n) = \text{IF}(x_1, f(\top, x_2, \dots, x_n), f(\perp, x_2, \dots, x_n))$$
 - 2 Common subterms are identified
 - 3 Redundant branches are removed: $\text{IF}(x_i, A, A) = A$
- Efficient implementation of boolean operations: $f_1 \cdot f_2$, $f_1 + f_2$, $\neg f$, including quantification.
- Canonical form yields free equivalence checks (for convergence of fixed points).

ROBDD for Even Parity

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



Construct an algorithm to compute $f_1 \odot f_2$, where \odot is \cdot or $+$.



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 τ given by c_0, c_1, \dots and equalities $c = d$ for constants c and d .

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

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

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



Proof Rules for Equality Logic

Reflexivity	$\Gamma \vdash a = a, \Delta$
Symmetry	$\frac{\Gamma \vdash a = b, \Delta}{\Gamma \vdash b = a, \Delta}$
Transitivity	$\frac{\Gamma \vdash a = b, \Delta \quad \Gamma \vdash b = c, \Delta}{\Gamma \vdash a = c, \Delta}$

- 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 Σ that assigns an arity to each symbol.
- Function symbols are used to form terms τ , so that constants are just 0-ary function symbols.

$$\tau := f(\tau_1, \dots, \tau_n), \text{ for } n \geq 0$$

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

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

$$M[a = b] = M[a] = M[b]$$

$$M[f(a_1, \dots, a_n)] = (M[f])(M[a_1], \dots, M[a_n])$$

- We need one additional proof rule.

Congruence	$\frac{\Gamma \vdash a_1 = b_1, \Delta \dots \Gamma \vdash a_n = b_n, \Delta}{\Gamma \vdash f(a_1, \dots, a_n) = f(b_1, \dots, b_n), \Delta}$
------------	---

Term Equality Proof Examples

Let $f^n(a)$ represent $\underbrace{f(\dots f(a)\dots)}_n$.

$$\frac{\frac{\frac{f^3(a) = f(a) \vdash f^3(a) = f(a)}{f^3(a) = f(a) \vdash f^4(a) = f^2(a)} C}{f^3(a) = f(a) \vdash f^5(a) = f^3(a)} C \quad \frac{f^3(a) = f(a) \vdash f^3(a) = f(a)}{f^3(a) = f(a) \vdash f^3(a) = f(a)} Ax}{f^3(a) = f(a) \vdash f^5(a) = f(a)} T$$

Show soundness and completeness of the above system.



- Equational Logic 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: $\frac{s_1=t_1, \dots, s_n=t_n}{f(s_1, \dots, s_n)=f(t_1, \dots, t_n)}$

5 Instantiation: $\frac{s=t}{\sigma(s)=\sigma(t)}$, for substitution σ .

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

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

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

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

$$M[x]\rho = \rho(x)$$

$$M[q(a_1, \dots, a_n)]\rho = M[q](M[a_1]\rho, \dots, M[a_n]\rho)$$

$$M[\forall x.A]\rho = \begin{cases} \top, & \text{if } M[A]\rho[x := d] \text{ for all } d \in D \\ \perp, & \text{otherwise} \end{cases}$$

$$M[\exists x.A]\rho = \begin{cases} \top, & \text{if } M[A]\rho[x := d] \text{ for some } d \in D \\ \perp, & \text{otherwise} \end{cases}$$

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

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

- 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 Σ 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.

- 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.

- Equational logic (or Universal Algebra) can be seen as a fragment of first-order logic where sequents are of the form $\Gamma \vdash A$, where the sequent formulas are all of the form $a = b$ or $\forall \bar{x}. a = b$, where \bar{x} is a sequence of variables x_1, \dots, x_n and $\forall \bar{x}. a = b$ is $\forall x_1. \dots \forall x_n. a = b$.
- Use equational logic to formalize (i.e., find axioms whose models are)
 - 1 Semigroups: A set G with an associative binary operator \cdot .
 - 2 Monoids: A set M with associative binary operator \cdot and unit 1 .
 - 3 Groups: A monoid with an right-inverse operator x^{-1}
- Prove that every group element has a left inverse.

Completeness of First-Order Logic

- 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 Γ 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 Γ has a *consistent* Henkin closure $H(\Gamma)$.
- As before, any consistent, Henkin closed set of formulas Γ has a complete, Henkin-closed extension $\widehat{\Gamma}$.
- Ex: Construct an interpretation $M_{\widehat{H(\Gamma)}}$ from $\widehat{H(\Gamma)}$ and show that it is a model for Γ .



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 $f(\bar{y})$, where \bar{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 $\exists \bar{x}. B$, then $\vdash A_H$ iff $\bigvee_{i=0}^n \sigma_i(B)$, for some Herbrand term substitutions $\sigma_1, \dots, \sigma_n$.
- [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).



- 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 $M \models q(x, y) \{x \mapsto c, y \mapsto d_c\}$. let M' extend M so that $M'(f)(c) = d_c$, for each $c \in |M|$: $M' \models \forall x.q(x, f(x))$.
- Conversely, if $M \models \forall x.q(x, f(x))$, then for every $c \in |M|$, $M \models q(x, y) \{x \mapsto c, y \mapsto M(f)(c)\}$.
- Prove the general case that any prenex formula can be Skolemized by replacing each existentially quantified variable y by a term $f(\bar{x})$, where f is a distinct, new function symbol for each y , and \bar{x} are the universally quantified variables governing y .

Unification

- A substitution is a map $\{x_1 \mapsto a_1, \dots, x_n \mapsto a_n\}$ from a finite set of variables $\{x_1, \dots, x_n\}$ 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 $\{x_1 \mapsto \sigma_1(a_1), \dots, x_n \mapsto \sigma_1(a_n)\}$, if σ_2 is of the form $\{x_1 \mapsto a_1, \dots, x_n \mapsto a_n\}$.
- Given two terms $f(x, g(y, y))$ and $f(g(y, y), x)$ (possibly containing free variables), find a substitution σ such that $\sigma(a) \equiv \sigma(b)$.
- Such a σ is called a unifier.
- Not all terms have such unifiers, e.g., $f(g(x))$ and $f(x)$.
- A substitution σ_1 is more general than σ_2 if the latter can be obtained as $\sigma \circ \sigma_1$, for some σ .
- Define the operation of computing the most general unifier, if there is one, and reporting failure, otherwise.



- 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 $(\forall x.\exists y.p(x, y)) \Rightarrow (\exists y.\forall x.p(x, y))$.

Dedekind–Peano Arithmetic

- 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 has can also 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: $\forall x, y.\exists z.\forall u \in z.u = x \vee u = y$ (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.)
- Comprehension: $\{x \in y \mid A\}$, for any formula A , $y \notin \text{vars}(A)$. (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 a minimal element (that is disjoint from it).
- Replacement: The image Y of a set X with respect to a functional $(\forall x \in X.\exists! y.A(x, y))$ rule $A(x, y)$, is a set.

- 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 join of two relations, where the first is a subset of $x \times y$ and the second is a subset of $y \times z$.
- Define the function space y^x of the functions that map elements of x to elements of y .

- 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 ω -inconsistency, or
 - 3 Neither U nor $\neg U$ is provable: an incompleteness.

Second Incompleteness Theorem

- The negation of the sentence U is Σ_1 , and Z can verify Σ_1 -completeness (every *valid* Σ_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 ω -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: $[T_1, \dots, T_n]$ for types T_1, \dots, T_n .
- Tuple terms: (a_1, \dots, 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

$$\llbracket \text{bool} \rrbracket = \{0, 1\}$$

$$\llbracket \text{real} \rrbracket = \mathbf{R}$$

$$\llbracket [T_1, \dots, T_n] \rrbracket = \llbracket T_1 \rrbracket \times \dots \times \llbracket T_n \rrbracket$$

$$\llbracket [T_1 \rightarrow T_2] \rrbracket = \llbracket T_2 \rrbracket^{\llbracket T_1 \rrbracket}$$



Higher-Order Proof Rules

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

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 $[T \rightarrow bool]$) 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.

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.

- 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