SE 2F03 Fall 2005

02 Propositional Logic

Instructor: W. M. Farmer

Revised: 26 September 2005

What is Propositional Logic?

- **Propositional logic** is the study of the truth or falsehood of **propositions** or **sentences** constructed using truth-functional **connectives**.
  - Also called **sentential logic**.
  - Began with the work of the Stoic philosophers, particularly Chrysippus, in the late 3rd century BCE.

- Most other logics are extensions of propositional logic.

- Applications:
  - Logical arguments.
  - Logical circuits (e.g., electronic circuits).
  - Boolean constraint modeling.

Propositional Symbols and Connectives

- A **propositional symbol** is a symbol that denotes an atomic proposition.

- A **propositional connective** is a symbol used to construct a **propositional formula** that denotes a compound proposition.
  - Each connective denotes an n-ary **truth function**
    $$f : B \times \cdots \times B \to B$$
    where $0 \geq n$ and $B = \{\mathsf{t}, \mathsf{f}\}$.

- Common propositional connectives:
  - 0-ary: T (truth), F (falsehood).
  - Unary: $\neg$ (negation).
  - Binary: $\wedge$ (conjunction), $\vee$ (disjunction), $\Rightarrow$ (implication), $\Leftrightarrow$ (bi-implication), $\mid$ (Sheffer's stroke).

Truth Tables (1)

- The truth-valued function that a propositional connective denotes can be represented by a **truth table**.

- Examples:

	T
	t

	F
	f

p	$(\neg p)$
t	f
f	t

p	q	$(p \wedge q)$
t	t	t
t	f	f
f	t	f
f	f	f

p	q	$(p \vee q)$
t	t	t
t	f	t
f	t	t
f	f	f

p	q	$(p \Rightarrow q)$
t	t	t
t	f	f
f	t	t
f	f	t

p	q	$(p \Leftrightarrow q)$
t	t	t
t	f	f
f	t	f
f	f	t

p	q	$(p \mid q)$
t	t	f
t	f	t
f	t	t
f	f	t

Truth Tables (2)

- Truth tables can be used to analyze the meaning of propositional formulas.

- Example (**Rule of Contraposition**):

p	q	$((p \Rightarrow q) \Leftrightarrow ((\neg q) \Rightarrow (\neg p)))$					
t	t	t	t	f	t	f	t
t	f	f	t	t	f	f	t
f	t	t	t	f	t	t	t
f	f	t	t	t	t	t	t

- A propositional formula A is a **tautology** and is **valid** if all of the final entries in the truth table for A are t.

- A propositional formula A is **satisfiable** if some of the final entries in the truth table for A are t.

- A and B are **logically equivalent** if $(A \Leftrightarrow B)$ is valid.

What is a Logic?

- Informally, a logic is a system of reasoning.

- Formally, a **logic** is a family of **formal languages** with:
  1. A common syntax.
  2. A common semantics.
  3. A notion of **logical consequence**.

- A logic may include a **proof system** for **proving** that a given formula is a logical consequence of a given set of formulas.

- Examples:
  - Propositional logic.
  - First-order logic.
  - Simple type theory (higher-order logic).

PROP: Syntax

- PROP is a simple version of propositional logic.

- The single language L of PROP is the pair $\{\mathcal{A}, \mathcal{B}\}$ where:
  - $\mathcal{A} = \{p_0, p_1, p_2, \ldots\}$ is a set of propositional symbols.
  - $\mathcal{B} = \{\neg, \Rightarrow\}$ is a set of propositional connectives.

- A **formula** of L is a string of symbols inductively defined by the following formation rules:
  1. Each $p \in \mathcal{A}$ is a formula of L.
  2. If A and B are formulas of L, then so are $(\neg A)$ and $(A \Rightarrow B)$.

- $\mathcal{B}$ is a **complete** set of propositional connectives, i.e., every truth function can be represented by a formula using only the members of $\mathcal{B}$.
  - $\{|\}$ is also complete.

PROP: Abbreviations

We will employ the following abbreviations:

T	denotes	$(p_0 \Rightarrow p_0)$.
F	denotes	$(\neg \mathsf{T})$.
$(A \vee B)$	denotes	$((\neg A) \Rightarrow B)$.
$(A \wedge B)$	denotes	$(\neg((\neg A) \vee (\neg B)))$.
$(A \Leftrightarrow B)$	denotes	$((A \Rightarrow B) \wedge (B \Rightarrow A))$.
$(A \mid B)$	denotes	$(\neg(A \wedge B))$.

PROP: Semantics

- Let $L = \{\mathcal{A}, \mathcal{B}\}$ be the language of PROP and $f_\neg, f_\Rightarrow$ be the truth values denoted by $\neg, \Rightarrow$, respectively.

- A **model** for L is an (interpretation) function I that assigns a truth value in $\{\mathsf{t}, \mathsf{f}\}$ to each $p \in \mathcal{A}$.

- The **valuation function** for I is the function V that maps formulas of L to $\{\mathsf{t}, \mathsf{f}\}$ and satisfies the following conditions:

  1. If $p \in \mathcal{A}$, then $V(p) = I(p)$.
  2. If A is a formula of L, then $V((\neg A)) = f_\neg(V(A))$.
  3. If A and B are formulas of L, then $V((A \Rightarrow B)) = f_\Rightarrow(V(A), V(B))$.

Proof Systems

- A **proof system** is a system of axioms and rules for constructing **formal proofs**.

- Proof systems come in several different styles.

- Two of the most popular styles are:

  1. **Hilbert style**.
  2. **Natural deduction**.

Hilbert-Style Proof Systems

- A **Hilbert-style proof system H** for a language L consists of:
  1. A set of formulas of L called **logical axioms**.
  2. A set of **rules of inference**.

- A **proof** of A from Σ in **H** is a finite sequence $B_1, \ldots, B_n$ of formulas of L with $B_n = A$ such that each B_i is either a logical axiom, a member of Σ, or follows from earlier B_j by one of the rules of inference.

- Hilbert-style proof systems are easy to understand but hard to use!

PROP: A Hilbert-Style Proof System

Let **H** be the following Hilbert-style proof system for PROP:

- The **logical axioms** of **H** are all formulas of L that are instances of the following three schemas:

  A1: $(A \Rightarrow (B \Rightarrow A))$.
  A2: $((A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C)))$.
  A3: $((\neg A \Rightarrow \neg B) \Rightarrow (B \Rightarrow A))$.

- The single **rule of inference** of **H** is **modus ponens**:

  MP: From A and $(A \Rightarrow B)$, infer B.

Metatheorems of Propositional Logic

- **Deduction Theorem**. $\Sigma \cup \{A\} \vdash_{\mathbf{H}} B$ implies $\Sigma \vdash_{\mathbf{H}} A \Rightarrow B$.

- **Soundness Theorem**. $\Sigma \vdash_{\mathbf{H}} A$ implies $\Sigma \models A$.

- **Completeness Theorem**. $\Sigma \models A$ implies $\Sigma \vdash_{\mathbf{H}} A$.

- **Soundness and Completeness Theorem (second form)**. Σ is consistent in **H** iff Σ is satisfiable.

- **Compactness Theorem**. If Σ is finitely satisfiable, then Σ is satisfiable.

Natural Deduction Systems

- A **natural deduction system** is a proof system consisting of a set of **introduction** and **elimination** rules.
  - An introduction rule introduces a logical symbol into the conclusion.
  - An elimination rule eliminates a logical symbol from a premise.

- **Reasoning from assumptions** (i.e., the deduction theorem) is formalized as the elimination rule for $\Rightarrow$.

- Huth and Ryan present in *Logic in Computer Science* a natural deduction system for propositional logic that is sound and complete.

- Natural deductions systems are harder to understand than Hilbert-style systems but much easier to use!

Normal Forms (1)

- A **literal** is a propositional symbol or the negation of a propositional symbol.

- A propositional formula is in **conjunctive normal form (CNF)** if it is a conjunction of disjunctions of literals.

- A propositional formula is in **disjunctive normal form (DNF)** if it is a disjunction of conjunctions of literals.

- **Theorem.** Every propositional formula is logically equivalent to:

  1. A formula in CNF.
  2. A formula in DNF.

- A CNF and a DNF of a formula can be “read” off of the formula’s truth table.

Normal Forms (2)

- **Lemma**.
  1. A disjunction D of literals is valid iff, for some propositional symbol p, D contains both p and $\neg p$.
  2. A conjunction C of literals is satisfiable iff, for all propositional symbols p, C does not contain both p and $\neg p$.

- **Theorem**.
  1. Validity of a formula in CNF can be checked in linear time.
  2. Satisfiability of a formula in DNF can be checked in linear time.

- **Proposition.** A formula A is valid [satisfiable] iff $\neg A$ is not satisfiable [valid].

Automated Reasoning Software

- Decision procedures:
  - **Validity checkers**.
  - **Satisfiability checkers**.

- Theorem proving systems.
  - **Automatic theorem provers** (automatically search for a proof of a given conjecture).
  - **Proof checkers** (automatically check the correctness of a given proof).
  - **Interactive theorem provers** (help the user to develop a proof of a given conjecture).