Some Questions for Recap

- What is a formal method?
- Can you give the syntax of propositional logic?
- Give a wff of PL of length at least 5!
- Translate the following into PL:
  \[ ((p \land q) \lor (\lnot p \land \lnot q)) \]
- What are the literals in the above formula?
- Can you describe the semantics of PL?
- What is the complexity of boolean satisfiability?
Propositional Calculus

• Various calculi have been proposed
  ▪ boolean satisfiability (SAT) algorithms
  ▪ tableau systems, natural deduction,
  ▪ enumeration of valid formulæ

• Hilbert-style axiom system

  \[ \vdash (\varphi \rightarrow (\psi \rightarrow \varphi)) \]  
  (weakening)

  \[ \vdash ((\varphi \rightarrow (\psi \rightarrow \xi)) \rightarrow ((\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \xi))) \]  
  (distribution)

  \[ \vdash (\neg \neg \varphi \rightarrow \varphi) \]  
  (excluded middle)

  \[ \varphi, (\varphi \rightarrow \psi) \vdash \psi \]  
  (modus ponens)

• Derivability
  ▪ All substitution instances of axioms are *derivable*
  ▪ If all antecedents of a rule are derivable, so is the consequent
An Example Derivation

Show ⊢ (p → p)

(1) ⊢ (p → ((p → p) → p)) → ((p → (p → p)) → (p → p)) (dis)
(2) ⊢ (p → ((p → p) → p)) (wea)
(3) ⊢ ((p → (p → p)) → (p → p)) (1,2,mp)
(4) ⊢ (p → (p → p)) (wea)
(5) ⊢ (p → p) (3,4,mp)
Correctness and Completeness

- **Correctness:** $\vdash \varphi \Rightarrow \models \varphi$
  
  Only valid formulæ can be derived
  
  - Induction on the length of the derivation
  - Show that all axiom instances are valid, and that the consequent of (mp) is valid if both antecedents are

- **Completeness:** $\models \varphi \Rightarrow \vdash \varphi$
  
  All valid formulæ can be derived
  
  - Show that consistent formulæ are satisfiable
    
    $\lnot \vdash \varphi \Rightarrow \lnot \models \varphi$
Consistency and Satisfiability

• A finite set $\Phi$ of formulæ is **consistent**, if $\neg \vdash \wedge_{\psi \in \Phi} \psi$

• Extension lemma: If $\Phi$ is a finite consistent set of formulæ and $\psi$ is any formula, then $\Phi \cup \{\psi\}$ or $\Phi \cup \{\neg \psi\}$ is consistent
  - Assume $\vdash \neg (\Phi \land \psi)$ and $\vdash \neg (\Phi \land \neg \psi)$. Then $\vdash (\Phi \rightarrow \neg \psi)$ and $\vdash (\Phi \rightarrow \neg \neg \psi)$. Therefore $\vdash \neg \Phi$, a contradiction.

• Let $\text{SF}(\varphi)$ be the set of all subformulæ of $\varphi$

• For any consistent $\varphi$, let $\varphi^\#$ be a **maximal consistent extension** of $\varphi$ (i.e., $\varphi \in \varphi^\#$ and for every $\psi \in \text{SF}(\varphi)$, either $\psi \in \varphi^\#$ or $\neg \psi \in \varphi^\#$. (Existence guaranteed by extension lemma)
Canonical models

- For a maximal consistent set $\varphi^\#$, the *canonical model* $\text{CM}(\varphi^\#)$ is defined by $I(p) = \text{true}$ iff $p \in \varphi^\#$.

- Truth lemma: For any $\psi \in \text{SF}(\varphi)$, $I(\psi) = \text{true}$ iff $\psi \in \varphi^\#$
  - Case $\psi = p$: by construction
  - Case $\psi = \bot$: $\Phi \cup \{\bot\}$ cannot be consistent
  - Case $\psi = (\psi_1 \rightarrow \psi_2)$: by induction hypothesis and derivation

- Therefore, if $\varphi$ is consistent, then for any maximal consistent set $\varphi^\#$, $\text{CM}(\varphi^\#) \models \varphi$
  - any consistent formula is satisfiable
  - any unsatisfiable formula is inconsistent
  - any valid formula is derivable
Example: Combinational Circuits

- **Multiplexer**

  - S selects whether $I_0$ or $I_1$ is output to $Y$
  - $Y = \text{if } S \text{ then } I_1 \text{ else } I_0 \text{ end}$
  - $(Y \leftrightarrow ((S \land I_1) \lor (\neg S \land I_0))))$

Pictures taken from: http://www.scs.ryerson.ca/~aabhari/cps213Chapter4.ppt
Boolean Specifications

- Evaluator (output is 1 if input matches a certain binary value)
- Encoder (output i is set if binary number i is on input lines)
- Majority function (output is 1 if half or more of the inputs are 1)
- Comparator (output is 1 if input0 > input1)
- Half-Adder, Full-Adder, ...
Software Example

- Code generator optimization
  - if \((p \text{ and } q)\) then if \((r)\) then \(x\) else \(y\) else if \((q \text{ or } r)\) then \(y\) else if \((p \text{ and } \text{not } r)\) then \(x\) else \(y\)

- Loop optimization
Verification of Boolean Functions

- **Latch-Up:** can a certain line go up?
  - does \((\varphi \rightarrow \neg L_0)\) hold?
  - is \((\varphi \land L_0)\) satisfiable?

- **Given \(\varphi, \psi\):** does \((\varphi \leftrightarrow \psi)\) hold?
  - usually reduced to SAT:
    - is \(((\varphi \land \neg \psi) \lor (\neg \varphi \land \psi))\) satisfiable?
  - efficient SAT-solver exist (annual competition)
  - partitioning techniques

- **any output depends only on some inputs**
  - find which ones
  - generate test patterns (BIST: built-in-self-test)
Propositional Resolution

- A clause is a disjunction of literals
- Resolution rule: Given two clauses
  \[ C_1 = (x_1 \lor \ldots \lor x_n \lor p) \quad \text{or} \quad (\neg p \rightarrow x_1 \lor \ldots \lor x_n) \]
  \[ C_2 = (y_1 \lor \ldots \lor y_m \lor \neg p) \quad \text{or} \quad (p \rightarrow y_1 \lor \ldots \lor y_n) \]
  infer
  \[ C_3 = (x_1 \lor \ldots \lor x_n \lor y_1 \lor \ldots \lor y_m) \]
- If the empty clause is derivable, the set of clauses is unsatifiable \((\bot) = (x \lor \bot)\)
SAT Solving

• DPLL solvers
  ▪ backtracking-based resolution: let $R$ be a set of clauses, $p$ a proposition
    $$R = ((R \land p) \lor (R \land \neg p))$$
  ▪ unit propagation: no backtracking necessary
  ▪ pure literal elimination: $p$ only positive or only negative

from wikipedia
Optimizing Boolean Functions

- Given $\varphi$; find $\psi$ such that $(\varphi \leftrightarrow \psi)$ holds and $\psi$ is „optimal“
  - much harder question
  - optimal wrt. speed / size / power /...
  - translation to normal form (e.g., OBDD)
Break!
Predicate Logic

- used to formalize mathematical reasoning
  - dates back to Frege (1879) „Begriffsschrift“
    - „Eine der arithmetischen nachgebildete Formelsprache des reinen Denkens“
  - individuals, predicates (sets of individuals), relations (sets of pairs), ...
  - quantification of statements (quantum = how much)
    - all, none, at least one, at most one, some, most, many, ...
    - need for variables to denote “arbitrary” objects
  - In contrast to propositional logic, first-order logic adds
    - structure to basic propositions
    - quantification on (infinite) domains
FOL: Syntax

- New syntactic elements
  - $\mathcal{R}$ is a set of *relation symbols*, where each $p \in \mathcal{R}$ has an arity $n \in \mathbb{N}_0$
  - $\mathcal{V}$ is a denumerable set of *(first-order or individual)* variables
  - An *atomic formula* is $p(x_1, \ldots, x_n)$, where $p \in \mathcal{R}$ is $n$-ary and $(x_1, \ldots, x_n) \in \mathcal{V}^n$.

- Syntax of first-order logic
  \[
  \text{FOL} ::= \mathcal{R}(\mathcal{V}^n) \mid \bot \mid (\text{FOL} \rightarrow \text{FOL}) \mid \exists \mathcal{V} \text{ FOL}
  \]
FOL: Syntax

- Abbreviations and parenthesis as in PL
  - Of course, $\forall x \ \varphi = \neg \exists x \ \neg \varphi$

- Propositions = 0-ary relations
  - Predicates = 1-ary relations
  - if all predicates are propositions, then FOL = PL

- Examples
  - $\exists x \forall x \exists x \ (p() \rightarrow \exists x (q() \rightarrow p()))$
  - $\exists x \forall x \exists y \ \neg p(x)$
  - $\exists x \exists y \ (p(x,y) \rightarrow p(y,x))$
  - $(\exists x \forall y \ p(x,y) \rightarrow \forall y \exists x \ p(x,y))$
Typed FOL

- Often, types/sorts are used to differentiate domains
- Signature $\Sigma = (\mathcal{D}, \mathcal{F}, \mathcal{R})$, where
  - $\mathcal{D}$ is a (finite) set of *domain names*
  - $\mathcal{F}$ is a set of *function symbols*, where each $f \in \mathcal{F}$ has an arity $n \in \mathbb{N}_0$ and a type $D \in \mathcal{D}^{n+1}$
    - 0-ary functions are called *constants*
  - $\mathcal{R}$ is a set of *relation symbols*, where each $p \in \mathcal{R}$ has an arity $n \in \mathbb{N}_0$ and a type $D \in \mathcal{D}^n$
    - unary relations are called *predicates*
    - propositions can be seen as 0-ary relations
- *Remark*: domains and types are for ease of use only (can be simulated in an untyped setting by additional predicates)
Terms and Formulas

• Let again $\mathcal{V}$ be a (denumerable) set of (first-order) variables, where each variable has a type $D \in \mathcal{D}$ (written as $x:D$) (for any type, there is an unlimited supply of variables of that type)

• The notions Term and Atomic Formula AtF are defined recursively:
  - each variable of type $D$ is a term of type $D$
  - if $f$ is an $n$-ary function symbol of type $(D_1, \ldots, D_n, D_{n+1})$ and $t_1, \ldots, t_n$ are terms of type $D_1, \ldots, D_n$, then $f(t_1, \ldots, t_n)$ is a term of type $D_{n+1}$
  - if $p$ is an $n$-ary relation symbol of type $(D_1, \ldots, D_n)$ and $t_1, \ldots, t_n$ are terms of type $D_1, \ldots, D_n$, then $p(t_1, \ldots, t_n)$ is an atomic formula

• Revised syntax of first-order logic
  $$\text{FOL ::= AtF} \mid \bot \mid (\text{FOL} \rightarrow \text{FOL}) \mid \exists \mathcal{V} : \mathcal{D} \text{ FOL}$$
Examples

• $\forall x: \text{Boy} \, \exists y: \text{Girl} \, \text{loves}(x,y)$
• $\forall x: \text{Human} \, \exists y: \text{Human} \, (\text{needs}(x,y) \land \text{loves}(y,x))$
• $\forall x,y: \text{Int} \, \text{equals}(\text{plus}(x,y), \text{plus}(y,x))$
• $\forall x: \text{Int} \, \neg \text{equals}((\text{zero}()), \text{succ}(x))$
• ...


(We give the typed semantics only)

First-Order Model

- Let a universe $U$ be some nonempty set, and let $\emptyset \neq D^U \subseteq U$ for every $D \in D$ be the domain of $D$
- Interpretation $I$: assignment $F \mapsto U^{n+1}$
  $R \mapsto U^n$
- Valuation $V$: assignment $\forall \mapsto U$
  interpretations and valuations must respect typing
- Model $M$: $(U, I, V)$
FOL: Semantics

- Given a model $M: (U,I,V)$, the value $t^M$ of term $t$ (of type $D$) can be defined inductively
  - if $t=x \in V$, then $t^M = V(x)$
  - if $t=f(t_1,\ldots,t_n)$, then $t^M = I(f)(t_1^M,\ldots,t_n^M)$

- Likewise, the validation relation $\models$ between model $M$ and formula $\varphi$
  - $M \models p(t_1,\ldots,t_n)$ if $(t_1^M,\ldots,t_n^M) \in I(p)$
  - $M \not\models \bot$; $M \models (\varphi \rightarrow \psi)$ if $M \models \varphi$ implies $M \models \psi$
  - $M \models \exists x \varphi$ if $M' \models \varphi$ for some $M'$ which differs at most in $V(x)$ from $M$

- Validity and satisfiability is defined as in the propositional case
Examples

- \( \models \forall x \varphi \rightarrow \exists x \varphi \)
- \( \models \forall x \varphi \land \forall x \psi \rightarrow \forall x (\varphi \land \psi) \)
- \( \models \exists x \varphi \lor \exists x \psi \rightarrow \exists x (\varphi \lor \psi) \)
- \( \models \forall x \forall y \varphi \rightarrow \forall y \forall x \varphi \)
- \( \models \forall x \varphi \rightarrow \varphi(x:=t) \)
- If \( \models \varphi \), then \( \models \forall x \varphi \)
FOL: Calculus

- A sound and complete axiom system for FOL:
  - all substitution instances of axioms of PL
  - modus ponens: \( \varphi, (\varphi \rightarrow \psi) \vdash \psi \)
  - \( \vdash (\varphi(x:=t) \rightarrow \exists x \varphi) \) instantiation
  - \( (\varphi \rightarrow \psi) \vdash (\exists x \varphi \rightarrow \psi) \) if \( x \) doesn't occur in \( \psi \) particularization

- Relaxation: particularization may be applied if there is no free occurrence of \( x \) in \( \psi \); i.e., \( x \) may occur in \( \psi \) inside the scope of a quantification
FOL: Completeness

• As in the propositional case, correctness is easy
  ($\vdash \phi \Rightarrow \models \phi$, “every derivable formula is valid”)

• Completeness ($\models \phi \Rightarrow \vdash \phi$, “every valid formula is derivable”)
  follows with a similar proof as previously:
  given a consistent formula, construct a model satisfying it
  $\sim \vdash \neg \phi \Rightarrow \sim \models \neg \phi$

• Extension lemma: If $\Phi$ is a finite consistent set of formulæ and $\psi$ is any formula, then $\Phi \cup \{\psi\}$ or $\Phi \cup \{\neg \psi\}$ is consistent

• Needs additionally: If $\Phi$ is any consistent set of formulæ and $\forall x \psi$ is a formula in $\Phi$, then $\Phi \cup \{\psi(t)\}$ is consistent for any term $t$

• From this, a canonical model can be constructed as before
Example

- Consider the formula

\[ \forall xyz ((p(x, y) \land p(y, z)) \rightarrow p(x, z)) \]
\[ \land \forall x \neg p(x, x) \land \forall x p(x, f(x)) \]

This formula is satisfiable only in infinite models