Computer-Aided Verification

CS745/ECE745

Dr. Borzoo Bonakdarpour

University of Waterloo

(Fall 2013)

Predicate Logic and Theorem Proving

(Some Slides Adapted from Nancy Day's Lectures)

Agenda

- Predicate Logic Sytax and Semantics
- Extension of Sequent Calculus for FOL
- Resolution
- Definability and Compactness

First-order Logic Syntax and Semantics

Motivation

There are some kinds of human reasoning that we cannot do in propositional logic. For example:

Every person likes ice cream.
Billy is a person.
Therefore, Billy likes ice cream.

In propositional logic, the best we can do is $(A \wedge B) \Rightarrow C$, which is not a tautology.

Motivation

We need to be able to refer to *objects* . We want to symbolize both a *claim* and the object about which the claim is made. We also need to refer to *relations* between objects, as in “Waterloo is west of Toronto”. If we can refer to objects, we also want to be able to capture the meaning of *every* and *some* of .

The *predicates* and *quantifiers* of predicate logic allow us to capture these concepts.

Examples

All grass is green.

$\forall y \bullet Grass(y) \Rightarrow Green(y)$

Something is rotten in the state of Denmark.

$\exists x \bullet Rotten(x) \wedge Denmark(x)$

We can also have n-ary predicates. Example:

Every even number is divisible by two.

$\forall x \bullet Even(x) \Rightarrow Div(x, 2)$

The "$\bullet$" is pronounced "such that". For the moment, we are not dealing with types.

Quantifiers

Universal quantification $\forall$ corresponds to finite or infinite conjunction of the application of the predicate to all elements of the domain.

Existential quantification $\exists$ corresponds to finite or infinite disjunction of the application of the predicate to all elements of the domain.

Relationship between $\forall$ and $\exists$:

$\forall x \bullet P(x)$ is the same as $\neg\exists x \bullet \neg P(x)$
$\exists x \bullet P(x)$ is the same as $\neg\forall x \bullet \neg P(x)$

Functions

Consider how to formalize:

Mary's father likes music

One possible way: $\exists x \bullet Father(x, Mary) \wedge Likes(x, Music)$

which means: Mary has at least one father and he likes music. We'd like to capture the idea that Mary only has one father. We use functions to capture a single object that can be in relation to another object.

Example: $Likes(father(Mary), Music)$

We can also have n-ary functions.

Predicate Logic: Syntax

The *syntax* of predicate logic consists of:

1. constants
2. variables $x, y, \cdots$
3. functions
4. predicates
5. logical connectives
6. quantifiers $(\forall, \exists)$
7. punctuation: $\bullet$, ( )

Predicate Logic: Syntax

Definition. *Terms* are defined as follows:

1. Every constant is a term.
2. Every variable is a term.
3. If $t_1, t_2, t_3, \cdots t_n$ are terms then $f(t_1, t_2, t_3, \cdots t_n)$ is a term, where f is an n-ary function.
4. Nothing else is a term.

Predicate Logic: Syntax

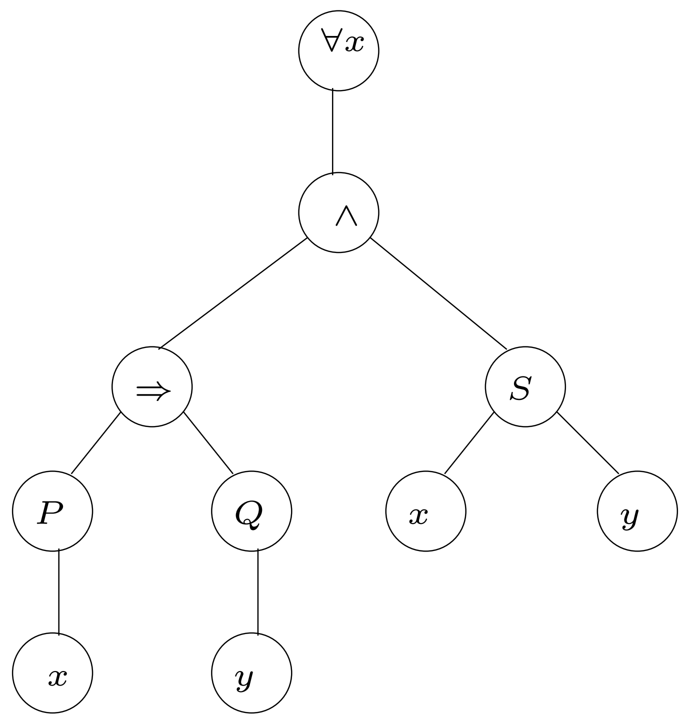
Definition. Well-formed formulas are defined as follows:

1. $P(t_1, t_2, t_3, \cdots t_n)$ is a wff, where t_i is a term, and P is an n-ary predicate. These are called *atomic formulas*.
2. If A and B are wffs, then so are $(\neg A)$, $(A \wedge B)$, $(A \vee B)$, $(A \Rightarrow B)$, and $(A \Leftrightarrow B)$.
3. If A is a wff, so is $\forall x \bullet A$.
4. If A is a wff, so is $\exists x \bullet A$.
5. Nothing else is a wff.

We often omit the brackets using the same precedence rules as propositional logic for the logical connectives.

Scope and Binding of Variables

Consider a parse tree for $\forall x \bullet (P(x) \Rightarrow Q(x)) \wedge S(x, y)$:



Scope and Binding of Variables

Variables occur both in nodes next to quantifiers and as leaf nodes in the parse tree.

A variable x is *bound* if starting at the leaf of x, we walk up the tree and run into a node with a quantifier and x.

A variable x is *free* if starting at the leaf of x, we walk up the tree and do not run into a node with a quantifier and x.

The *scope* of a variable x is the subtree starting at the node with the variable and its quantifier (where it is bound) minus any subtrees with $\forall x$ or $\exists x$ at their root.

Example: $\forall x \bullet (\forall x \bullet (P(x) \wedge Q(x))) \Rightarrow (\neg P(x) \vee Q(y))$

A wff is *closed* if it contains no free occurrences of any variable.

Substitution

Variables are place holders. Given a variable x, a term t and a formula P, we define $P[t/x]$ to be the formula obtained by replacing all *free* occurrences of variable x in P with t.

We have to watch out for *variable capture* in substitution.

Given a term t, a variable x and a formula A, we say that "t is free for x in A" if no free x leaf in A occurs in the scope of $\forall y$ or $\exists y$ for any free variable y occurring in t.

Example:

A is $\forall y \bullet P(x) \wedge Q(y)$
t is $f(y)$
t is NOT free for x in A.

Whenever we use $P[t/x]$, t and x must both be free for x in P.

Predicate Logic: Semantics

Recall that a semantics is a mapping between two worlds. A *model* for predicate logic consists of:

1. a non-empty *domain* of objects: D
2. a mapping I, called an *interpretation* that associates the terms of the syntax with objects in a domain

It's important that D be non-empty, otherwise some tautologies would not hold such as $(\forall x.A(x)) \Rightarrow (\exists x.A(x))$.

Interpretations

An *interpretation* assigns:

1. a fixed element $c' \in D$ to each constant c of the syntax
2. an n-ary function $f' : D^n \to D$ to each n-ary function, f, of the syntax
3. an n-ary relation $R' \subseteq D^n$ to each n-ary predicate, R, of the syntax

Example of a Model

Let's say our syntax has the constant c, the function f (unary), and two predicates P, and Q (both binary). In our model, choose the domain to be the natural numbers.

- $I(c)$ is 0
- $I(f)$ is suc, the successor function
- $I(P)$ is $<$
- $I(Q)$ is $=$

Example of a Model

What is the meaning of $P(c, f(c))$ in this model?

$$\begin{aligned} I(P(c, f(c))) &= I(c) < I(f(c)) \\ &= 0 < \mathsf{suc}(I(c)) \\ &= 0 < \mathsf{suc}(0) \\ &= 0 < 1 \end{aligned}$$

which is true.

Valuations

Definition. A valuation v, in an interpretation I, is a function from the terms to the domain D such that:

1. $v(c) = I(c)$
2. $v(f(t1, \cdots t_n)) = f(v(t_1, \cdots t_n))$
3. $v(x) \in D$, i.e., each variable is mapped onto some element in D

Example of a Valuation

- D is the set of natural numbers
- g is the function +
- h is the function suc
- c (constant) is 3
- y (variable) is 1

$$\begin{aligned} v(g(h(c), y)) &= v(h(c)) + v(y) \\ &= \text{suc}(v(c)) + 1 \\ &= \text{suc}(3) + 1 \\ &= 5 \end{aligned}$$

Predicate Logic: Satisfiability

Given a model m, with domain D and interpretation I, and a valuation v,

1. If A is an atomic wff, $P(t_1 \cdots t_n)$, m and v satisfy A iff $P'(v(t_1) \cdots v(t_n))$
2. If A has form $\neg B$, then m and v satisfy A iff m and v do not satisfy B.
3. If A has the form $B \wedge C$, then m and v satisfy A iff m and v satisfy B and m and v satisfy C , etc. for the other connectives.

Predicate Logic: Satisfiability

5. If A has the form $\forall x \bullet B$, then m and v satisfy A iff v satisfies B for all elements of D, i.e., for all $v(x) \in D$.
6. If A has the form $\exists x \bullet B$, then v satisfies A iff v satisfies B for some element of D, i.e., there is some $v(x) \in D$ for which B is satisfied.

Notice that if the formula is closed, then the valuation depends only on the model.

Notational convenience: while we have defined valuations for terms only, we will extend the use of v to be for wff also, mapping relations to their counterparts on the domain, and the logical connectives as we did in Boolean valuations.

Validity (Tautologies)

Definition. A predicate logic formula is *satisfiable* if *there exists* a model and *there exists* a valuation that satisfies the formula (i.e., in which the formula returns **T**).

Definition. A predicate logic formula is logically *valid* (tautology) if it is true in every model. It must be satisfied by every valuation in every model.

Definition. A wff, A, of predicate logic is a *contradiction* if it is false in every model. It must be false in every valuation in every model.

Semantic Entailment

Semantic entailment has the same meaning as it did for propositional logic.

$$\phi_1, \phi_2, \phi_3 \models \psi$$

means that for all v if $v(\phi_1) = \mathbf{T}$ and $v(\phi_2) = \mathbf{T}$, and $v(\phi_3) = \mathbf{T}$, then $v(\psi) = \mathbf{T}$, which is equivalent to saying

$$(\phi_1 \wedge \phi_2 \wedge \phi_3) \Rightarrow \psi$$

is a tautology, i.e.,

$$\boxed{\phi_1, \phi_2, \phi_3 \models \psi \;\equiv\; (\phi_1 \wedge \phi_2 \wedge \phi_3) \Rightarrow \psi}$$

Closed Formulas

Recall: A wff is *closed* if it contains no free occurrences of any variable.

We will mostly restrict ourselves to closed formulas. For formulas with free variables, close the formula by universally quantifying over all its free variables.

Sat., Tautologies, Contradictions

A closed predicate logic formula, is satisfiable if *there is* a model I in which the formula returns **T**.

A closed predicate logic formula, A, is a *tautology* if it is **T** in every model.

$$\models A$$

A closed predicate logic formula is a *contradiction* if it is **F** in every model.

Question. What is the complexity of checking the satisfiability of a predicate logic formula?.

Counterexamples

How can we show a formula is not a tautology?

Provide a counterexample. A *counterexample* for a closed formula is a model in which the formula does not have the truth value **T**.

Sequent Calculus in FOL

Sequent Calculus Rules for FOL

Rules for universal quantifier:

$$\frac{\Gamma, F(a) \hookrightarrow \Delta}{\Gamma, \forall x. F(x) \hookrightarrow \Delta}$$

$$\frac{\Gamma \hookrightarrow \Delta, F(a)}{\Gamma \hookrightarrow \Delta, \forall x. F(x)}$$

Here, the variable a is free in F and $F(x)$ is obtained from $F(a)$ by replacing all free occurrences of a by x.

Sequent Calculus Rules for FOL

Rules for existential quantifier:

$$\frac{\Gamma \hookrightarrow \Delta, F(a)}{\Gamma \hookrightarrow \Delta, \exists x.F(x)}$$

$$\frac{\Gamma, F(a)\Delta}{\Gamma, \exists x.F(x) \hookrightarrow \Delta}$$

Here, the variable a is free in F and $F(x)$ is obtained from $F(a)$ by replacing all free occurrences of a by x.

Sequent Calculus Rules for FOL

Example. Show that $\neg\exists.A(x) \Rightarrow \forall x.\neg A(x)$

Step1: $A(a) \hookrightarrow A(a)$
Step2: $A(a) \hookrightarrow \exists x.A(x)$
Step3: $\neg\exists x.A(x) \hookrightarrow \neg A(a)$
Step4: $\neg\exists x.A(x) \hookrightarrow \forall x.\neg A(x)$
Step4: $\hookrightarrow \neg\exists x.A(x) \Rightarrow \forall x.\neg A(x)$

Proof by Resolution

Literals and Clauses

A *literal* is a propositional variable or the negation of a propositional variable.

Two literals are said to be *complements* (or *conjugate*), if one is the negation of the other (e.g., p and $\neg p$)

A formula of the form $C_i = p_1 \vee p_2 \vee \cdots \vee p_n$, where each p_i is a literal is called a *clause*.

CNF

A formula in the *conjunctive normal form* (CNF) is a conjunction of clauses

For example, these formulas are in CNF:

$$(p \vee q) \wedge (\neg q \vee r \vee \neg m) \wedge (m \vee \neg n)$$

$$p \wedge q$$

It is possible to convert any formula into an equivalent formula in CNF.

CNF

The CNF equivalent of the following formulas:

$$(p \wedge q) \vee r$$

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

are these:

$$(p \vee r) \wedge (q \vee r)$$

$$\neg p \wedge \neg q$$

Resolution Rule

$$\frac{p_1 \vee \cdots \vee p_i \vee \ldots p_n, \quad q_1 \vee \cdots \vee q_j \vee \ldots q_m}{p_1 \vee \cdots \vee p_{i-1} \vee p_{i+1} \vee \ldots p_n \vee q_1 \cdots \vee q_{j-1} \vee q_{j+1} \vee \cdots \vee q_m}$$

where $p_1 \ldots p_n, q_1 \ldots q_m$ are propositions and p_i and q_j are complements.

The clause produced by the resolution rule is called the *resolvent* of the two input clauses.

The upper side of the rull is in CNF and may have multiple clauses.

Proof by Resolution

The resolution rule can be used to develop a finite-step proof for propositional logic:

1- Transform the CNF formual into a set S of caulses. For example, for formula:

$$(p \vee q \vee r) \wedge (\neg r \vee \neg p \vee m) \wedge q$$

we have:

$$S = \{\{p, q, r\}, \{\neg r, \neg p, m\}, \{q\}\}$$

Proof by Resolution

2- The resolution rule is applied to all possible pairs of clauses that contain complementary literals. After each application of the resolution rule, the resulting sentence is simplified by

- Removing repeated literals.
- If the sentence contains complementary literals, it is removed (as a validity).
- If not, and if it is not yet present in the clause set S, then it is added to S, and is considered for further resolution inferences.

Proof by Resolution

Example:

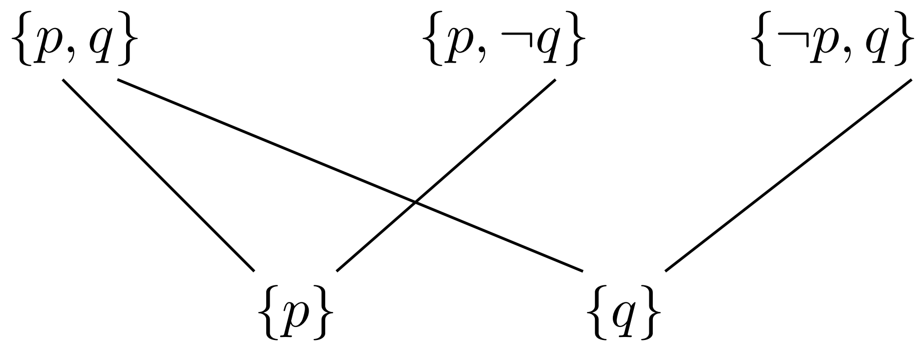
$$\frac{p \vee q, \quad \neg p \vee r}{q \vee r}$$

This is equal to (different syntax):

$$\frac{\{p, q\}, \{\neg p, r\}}{\{q, r\}}$$

Proof by Resolution

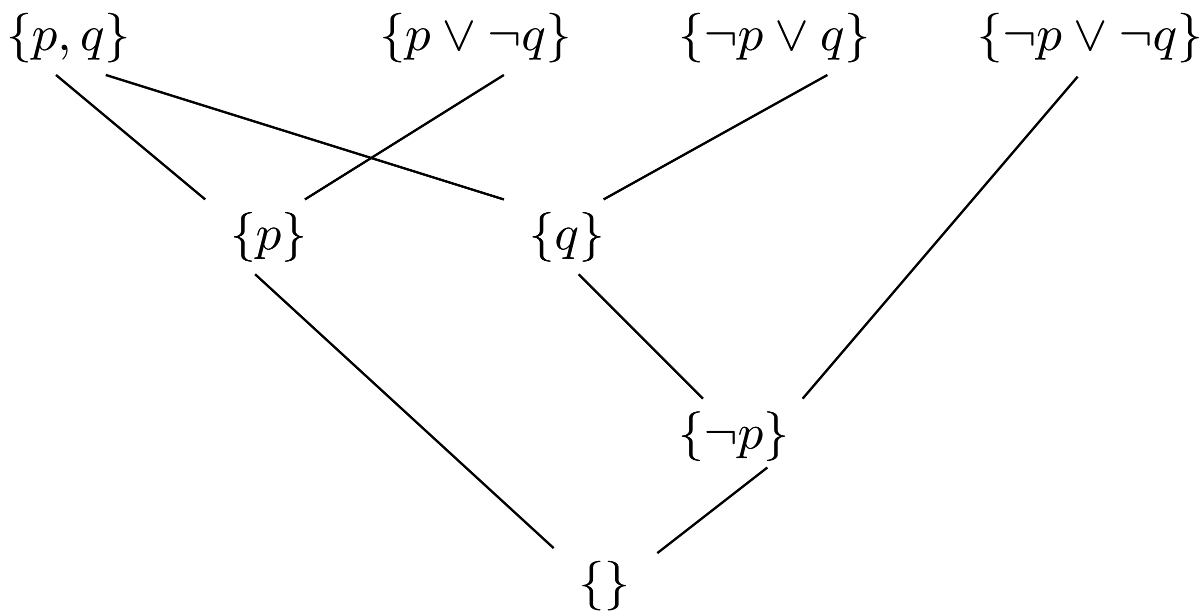
Example (in directed acyclic graph):



3- If the empty clause cannot be derived, and the resolution rule cannot be applied to derive any more new clauses, then the original formula is satisfiable.

Proof by Resolution

4- If after applying a resolution rule the empty clause is derived, the original formula is unsatisfiable (i.e., a contradiction).
Example:

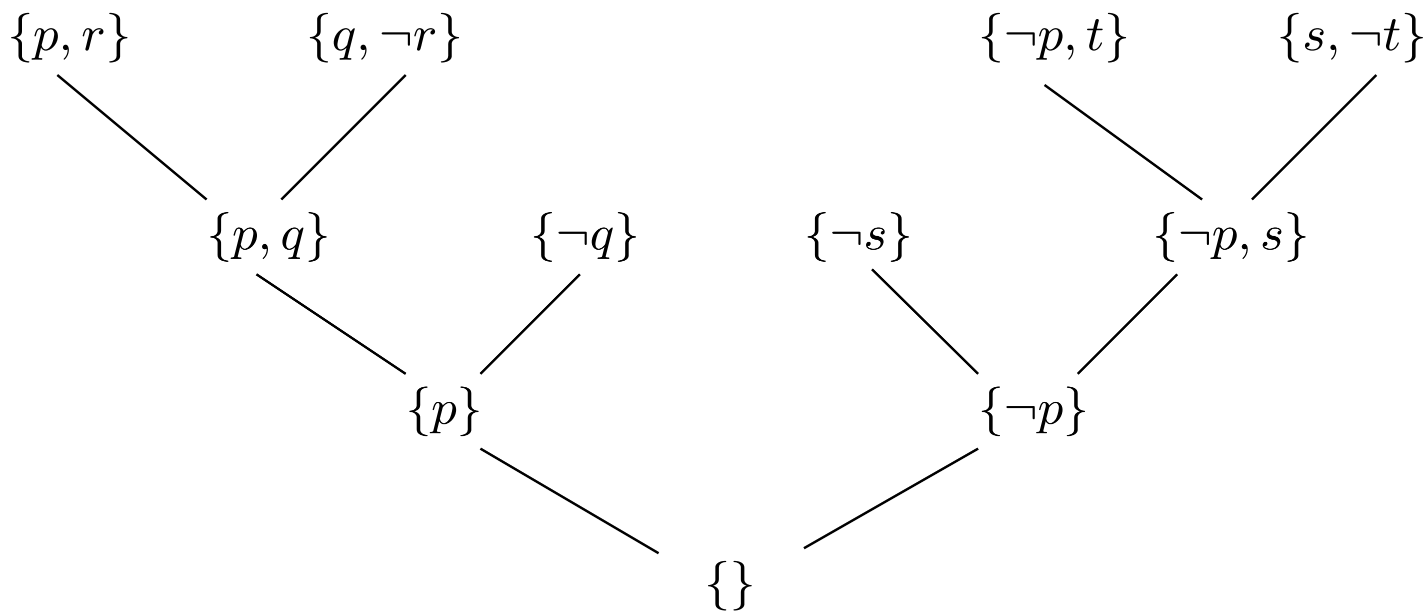


Example

$$S = (p \vee r) \wedge (r \Rightarrow q) \wedge \neg q \wedge (p \Rightarrow t) \wedge \neg s \wedge (t \Rightarrow s)$$

$$S = (p \vee r) \wedge (\neg r \vee q) \wedge \neg q \wedge (\neg p \vee t) \wedge \neg s \wedge (\neg t \vee s)$$

$$S = \{\{p, r\}, \{\neg r, q\}, \{\neg q\}, \{\neg p, t\}, \{\neg s\}, \{\neg t, s\}\}$$



Soundness and Completeness

Resolution for propositional logic is sound and complete.

Prenex Normal Form

A first-order formula is in *prenex normal form* (PNF), if it is written as a string of quantifiers followed by a quantifier-free part.

Every first-order formula has an equivalent formula in PNF. For example, formula

$$\forall x((\exists y A(y)) \vee ((\exists z B(z)) \to C(x)))$$

has the following PNF:

$$\forall x \exists y \forall z (A(y) \vee (B(z) \to C(x)))$$

Conversion to PNF

The rules for *conjunction* and *disjunction* say that

$(\forall x \phi) \land \psi$ is equivalent to $\forall x(\phi \land \psi)$
$(\forall x \phi) \lor \psi$ is equivalent to $\forall x(\phi \lor \psi)$

and

$(\exists x \phi) \land \psi$ is equivalent to $\exists x(\phi \land \psi)$
$(\exists x \phi) \lor \psi$ is equivalent to $\exists x(\phi \lor \psi)$

Conversion to PNF

The rules for *negation* say that

$\neg \exists x \phi$ is equivalent to $\forall x \neg \phi$

and

$\neg \forall x \phi$ is equivalent to $\exists x \neg \phi$

Conversion to PNF

The rules for *removing quantifiers* from the antecedent are:

$(\forall x \phi) \rightarrow \psi$ is equivalent to $\exists x (\phi \rightarrow \psi)$
$(\exists x \phi) \rightarrow \psi$ is equivalent to $\forall x (\phi \rightarrow \psi)$

The rules for *removing quantifiers* from the consequent are:

$\phi \rightarrow (\exists x \psi)$ is equivalent to $\exists x (\phi \rightarrow \psi)$
$\phi \rightarrow (\forall x \psi)$ is equivalent to $\forall x (\phi \rightarrow \psi)$

Example

Suppose that ϕ, ψ, and ρ are quantifier-free formulas and no two of these formulas share any free variable. The formula

$$(\phi \vee \exists x \psi) \rightarrow \forall z \rho$$

can be transformed into PNF as follows:

$$(\exists x(\phi \vee \psi)) \rightarrow \forall z \rho$$
$$\forall x((\phi \vee \psi) \rightarrow \forall z \rho)$$
$$\forall x(\forall z((\phi \vee \psi) \rightarrow \rho))$$
$$\forall x \forall z((\phi \vee \psi) \rightarrow \rho)$$

NNF

A formula is in *negation normal form* if negation occurs only immediately above propositions, and $\{\neg, \vee, \wedge\}$ are the only allowed Boolean connectives.

It is possible to convert any first-order formula to an equivalent formula in NNF. For exmple:

$\neg(\forall x.G)$ is $\exists x.\neg G$
$\neg(\exists x.G)$ is $\forall x.\neg G$
$\neg\neg G$ is G
$\neg(G_1 \wedge G_2)$ is $(\neg G_1) \vee (\neg G_2)$
$\neg(G_1 \vee G_2)$ is $(\neg G_1) \wedge (\neg G_2)$

SNF: Skolemization

Reduction to *Skolem normal form* is a method for removing existential quantifiers from first-order formulas

A first-order formula is in SNF, if it is in conjunctive PNF with only universal first-order quantifiers.

Important note: Skolemization only preserves *satisfiability*.

Skolemization

Skolemization is performed by replacing every existentially quantified variable y with a term $f(x_1, \ldots, x_n)$ where function f does not occur anywhere else in the formula.

If the formula is in PNF, $x_1, \ldots, x_n$ are the variables that are universally quantified where quantifiers precede that of y. The function f is called a *Skolem function*.

Skolemization

In general,

$\forall x_1 \dots x_k \exists y. \varphi(x_1 \dots x_k, y)$ is

$$\forall x_1 \dots x_k. \varphi(x_1 \dots x_k, f(x_1 \dots x_k))$$

For example, the formula

$$\forall x \exists y \forall z. P(x, y, z)$$

is not in SNF. Skolemization results in

$$\forall x \forall z. P(x, f(x), z)$$

Ground Clauses

A sentence A is in *clause form* iff it is a conjunction of (prenex) sentences of the form $\forall x_1 \ldots \forall x_m . C$, where C is a disjunction of literals, and the sets of bound variables $\{x_1, \ldots, x_m\}$ are disjoint for any two distinct clauses.

Each sentence $\forall x_1 \ldots \forall x_m . C$ is called a *clause*.

If a clause in A has no quantifiers and does not contain any variables, we say that it is a *ground clause*.

Ground Clauses

Lemma. For every sentence A, a sentence B in clause form such that A is valid iff B is unsatisfiable can be constructed.

Example

Let
$A = \neg\exists y.\forall z.(P(z,y) \Leftrightarrow \neg\exists x.(P(z,x) \wedge P(x,z)))$.

First, we negate A and eliminate $\Leftrightarrow$:

$$\exists y.\forall z.[(\neg P(z,y) \vee \neg\exists x.(P(z,x) \wedge P(x,z)))\wedge$$

$$(\exists x.(P(z,x) \wedge P(x,z)) \vee P(z,y))]$$

Example

Next, we put in this formula in NNF:

$$\exists y.\forall z.[(\neg P(z,y) \vee \forall x.(\neg P(z,x) \vee \neg P(x,z)))\wedge$$

$$(\exists x.(P(z,x) \wedge P(x,z)) \vee P(z,y))]$$

Next, we Skolemize:

$$\forall z.[(\neg P(z,a) \vee \forall x.(\neg P(z,x) \vee \neg P(x,z)))\wedge$$

$$((P(z,f(z)) \wedge P(f(z),z)) \vee P(z,a))]$$

Example

We now put in prenex form:

$$\forall z.\forall x.[(\neg P(z,a) \lor (\neg P(z,x) \lor \neg P(x,z)))\land$$
$$((P(z,f(z)) \land P(f(z),z)) \lor P(z,a))]$$

We put in CNF by distributing $\land$ over $\lor$:

$$\forall z.\forall x.[(\neg P(z,a) \lor (\neg P(z,x) \lor \neg P(x,z)))\land$$
$$(P(z,f(z)) \lor P(z,a)) \land (P(f(z),z)) \lor P(z,a))]$$

Example

Omitting universal quantifiers, we have the following three clauses:

$$C_1 = (\neg P(z_1, a) \vee (\neg P(z_1, x) \vee \neg P(x, z_1))$$

$$C_2 = (P(z_2, f(z_2)) \vee P(z_2, a))$$

$$C_3 = (P(f(z_3), z_3)) \vee P(z_3, a))]$$

Ground Resolution

Suppose, we want to prove (for the previous example) that $B = \neg A$ is unsatisfiable.

The *ground resolution* method is the resolution method applied to sets of ground clauses.

Ground Resolution

For example,

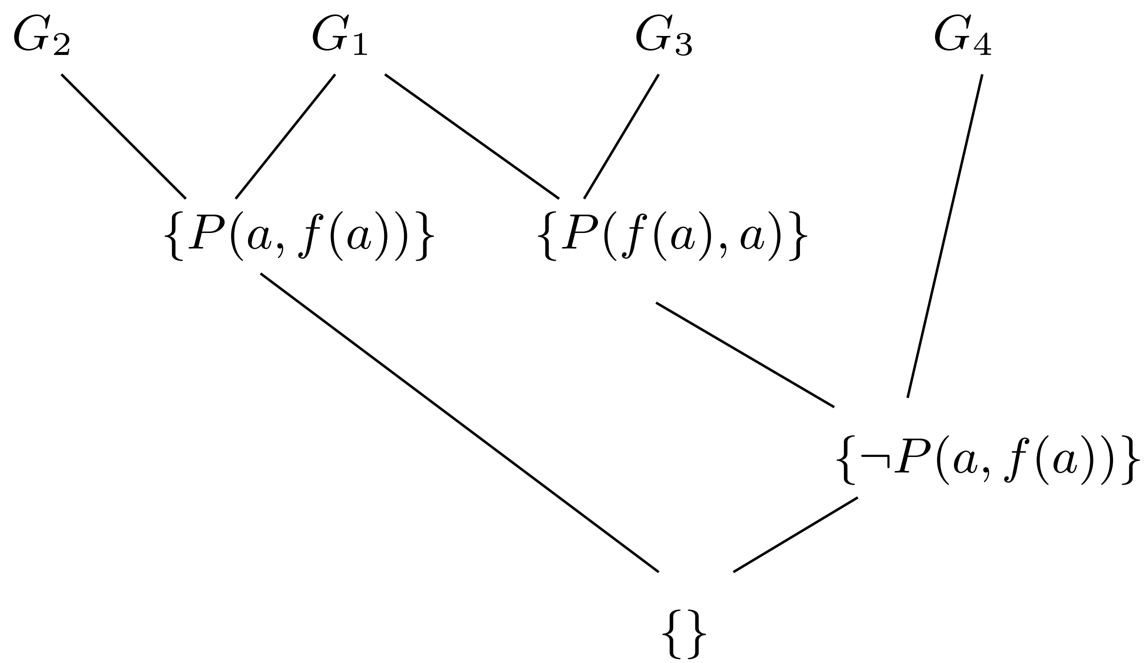
$G_1 = (\neg P(a,a))$
(from C_1, substituting a for x and z_1)

$G_2 = (P(a, f(a)) \vee P(a,a))$
(from C_2, substituting a for z_2)

$G_3 = (P(f(a), a)) \vee P(a,a))$
(from C_3, substituting a for z_3)

$G_4 = (\neg P(f(a), a) \vee \neg P(a, f(a)))$
(from C_1, substituting $f(a)$ for z_1 and a for x)

Example



Unification

To generalize ground resolution to arbitrary clauses, one is allowed to apply substitutions to the parent clauses.

For example, to obtain $\{P(a, f(a))\}$ from
$C_1 = (\neg P(z_1, a) \vee \neg P(z_1, x) \vee \neg P(x, z_1))$ and
$C_2 = (P(z_2, f(z_2)) \vee P(z_2, a))$,
first we substitute a for z_1, a for x, and a for z_2, obtaining

$$G_1 = (\neg P(a, a)) \text{ and } G_2 = (P(a, f(a)) \vee P(a, a))$$

and then we resolve on the literal $P(a, a)$.

Unification

Note that the two sets of literals $\{P(z_1, a), P(z_1, x), P(x, z_1)\}$ and $\{P(z_2, a)\}$ obtained by dropping the negation sign in C_1 have been *unified* by the substitution $(a/x, a/z1, a/z2)$.

Given two terms t and t' that do not share any variables, a substitution θ is called a *unifier* iff

$$\theta(t) = \theta(t')$$

Example

1. Let $t_1 = f(x, g(y))$ and $t_2 = f(g(u), g(z))$. The substitution $(g(u)/x, y/z)$ is a most general unifier yielding the most common instance $f(g(u), g(y))$.
2. However, $t_1 = f(x, g(y))$ and $t_2 = f(g(u), h(z))$ are not unifiable since this requires $g = h$.
3. Let $t_1 = f(x, g(x), x)$ and $t_2 = f(g(u), g(g(z)), z)$. To unify these two, we must have $x = g(u) = z$. But we also need $g(x) = g(g(z))$, that is, $x = g(z)$. This implies $z = g(z)$.

General Resolution

1. Find two clauses containing the same predicate, where it is negated in one clause but not in the other.
2. Perform a unification on the two predicates. (If the unification fails, you made a bad choice of predicates. Go back to the previous step and try again.)
3. If any unbound variables which were bound in the unified predicates also occur in other predicates in the two clauses, replace them with their bound values (terms) there as well.
4. Discard the unified predicates, and combine the remaining ones from the two clauses into a new clause, also joined by the $\vee$ operator.

Example

For clauses

$$A = (\neg P(z,a) \vee (\neg P(z,x) \vee \neg P(x,z))$$
$$B = (P(z,f(z)) \vee P(z,a))$$

We choose subsets $A' = A$ and $B' = \{P(z,a)\}$ and and unifier $(a/z, a/x)$, we obtain resolvent

$$C = \{P(a,f(a)\}$$

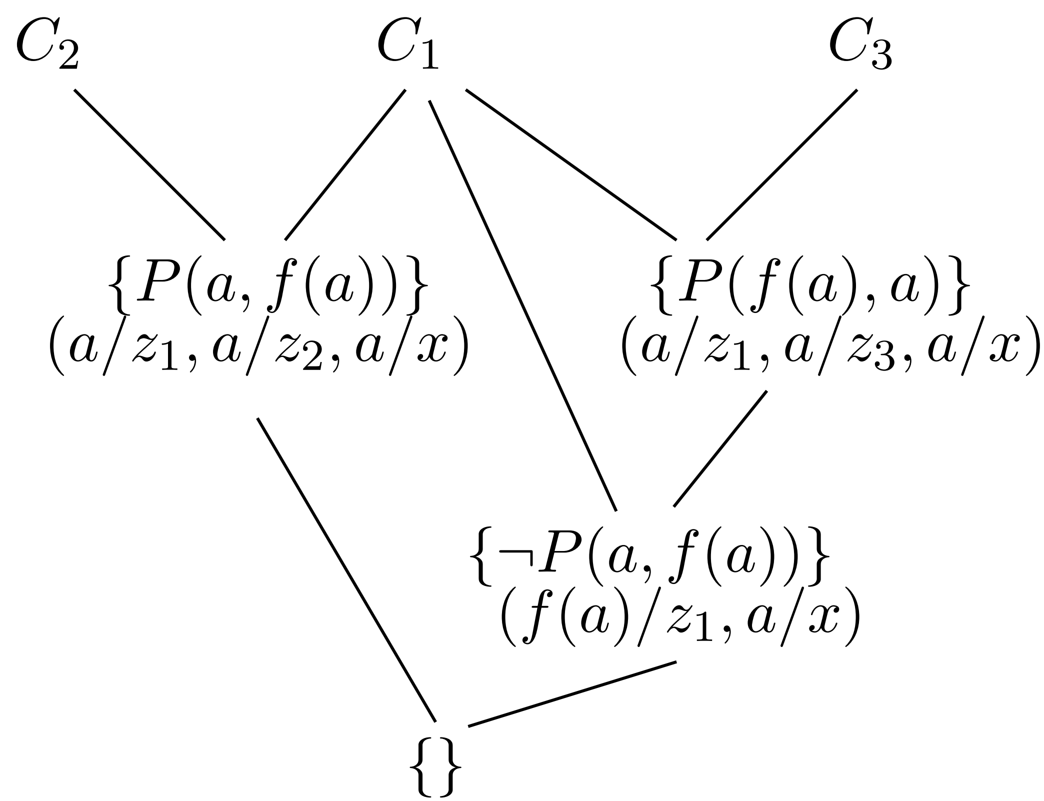
Example

$$C_1 = (\neg P(z_1, a) \vee (\neg P(z_1, x) \vee \neg P(x, z_1))$$

$$C_2 = (P(z_2, f(z_2)) \vee P(z_2, a))$$

$$C_3 = (P(f(z_3), z_3)) \vee P(z_3, a))]$$

Example



Definability and Compactness

Definability

Let $I = (D, (.)^I)$ be a first-order interpretation and φ a first-order formula. A set S of k-tuples over D, $S \subseteq D^k$, is *defined* by the formula φ if

$$S = \{(\theta(x_1), \ldots, \theta(x_k)) \mid I, \theta \models \varphi\}$$

A set S is *definable* in first-order logic if it is defined by some first-order formula φ.

Definability

Let Σ be a set of first-order sentences and $\mathcal{K}$ a set of interpretations. We say that Σ defines $\mathcal{K}$ if

$$I \in \mathcal{K} \text{ if and only if } I \models \Sigma.$$

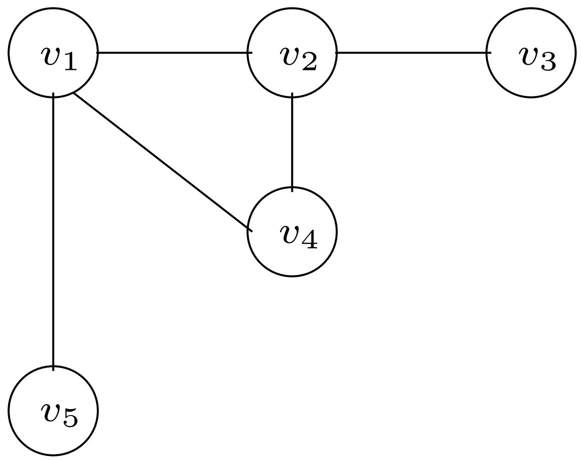
A set $\mathcal{K}$ is *(strongly) definable* if it is defined by a (finite) set of first-order formulas Σ.

Compactness in FOL

Theorem. Let Σ be a set of first-order formulas. Σ is satisfiable iff every finite subset of Σ is satisfiable.

Graphs

An undirected *graph* is a tuple (V, E), where V is a set of vertices and E is a set of edges. An edge is a pair (v_1, v_2), where $v_1, v_2 \in V$.



$$V = \{v_1, v_2, v_3, v_4, v_5\}$$
$$E = \{(v_1, v_2), (v_2, v_3), (v_2, v_4), (v_1, v_4), (v_1, v_5)\}$$

Graphs in FOL

If $(v_1, v_2) \in E$, we say that v_1 is *adjacent* to v_2.

Adjacency in a graph can be expressed by a binary relation. Thus, relation $E(v_1, v_2)$ is interpreted as "v_1 is adjacent to v_2". A graph is any model of the following 2 axioms:

1. $\forall x.\forall y.E(x, y) \Rightarrow E(y, x)$ ("if x is adjacent to y, then y is adjacent to x")
2. $\forall x.\neg E(x, x)$ ("no x is adjacent to itself")

Graphs in FOL

We can express many properties of a graph in the language of first-order logic.

For instance, the property “G contains a triangle” is the following formula:

$$\exists x.\exists y.\exists z.(E(x,y) \wedge E(y,z) \wedge E(z,x))$$

Example

Define first-order formulas for :

- A graph has *girth* of size 4
- A graph is 3-colorable

Graph Connectivity in FOL

We cannot express graph *connectivity* in FOL (i.e., graph connectivity is not definable in FOL).

Proof.

- Let predicate C express "G is a connected graph". We add constants s and t vertices.
- For any k, let L_k be the proposition "there is no path of length k between s and t". For example,

$$L_3 = \neg\exists x.\exists y.(E(s,x) \wedge E(x,y) \wedge E(y,t))$$

Graph Connectivity in FOL

- Now consider the set of propositions

$$\Sigma = \{\text{axiom}(1), \text{axiom}(2), C, L_1, L_2, \ldots\}$$

- Σ is finitely satisfiable: there do exist connected graphs with s and t, that are connected by an arbitrarily long path. This is because any finite subset $F \subset \Sigma$ must have bounded k's, such a graph satisifes F.

Graph Connectivity in FOL

- By the compactness theorem, Σ is satisfiable; i.e., there exists some model G of all propositions Σ, which is a graph that cannot be connected by a path of length k, for any k, for all k.
- This is clearly wrong. In a connected graph, any 2 nodes are connected by a path of finite length!

What does first-order mean?

We can only quantify over variables.

In higher-order logics, we can quantify over functions, and predicates. For example, in second-order logic, we can express the induction principle:

$$\forall P \bullet (P(0) \wedge (\forall n \bullet P(n) \Rightarrow P(n+1))) \Rightarrow (\forall n \bullet P(n))$$

Propositional logic can also be thought of as zero-order.