CS257: Introduction to Automated Reasoning

First-order Logic: Semantics

Stanford University

C∃NTVUR

Outline

- Semantics of First-order logic (MI 2.2)
- PCNF (CC 2.5) and Clausal Form
- First-order Resolution

* Some of the slides today are contributed by Clark Barrett and Giles Reger.

Semantics

The syntax of a first-order language is defined w.r.t. a **signature**, $\Sigma := \langle \Sigma^S, \Sigma^F \rangle$, where:

- Σ^S is a set of **sorts**
- Σ^F is a set of **function symbols**

Example Given $\Sigma_N = \langle \Sigma^S := \{\mathrm{Nat}\}, \Sigma^F := \{0, S, +, \times, <\} \rangle$, $< 0Sx$ is a Σ-formula.

In propositional logic, the truth of a formula was determined by a **variable assignment** over the propositional symbols.

In first-order logic, the truth of a Σ-formula depends on:

1. what collection of things each sort σ refers to
2. what the function symbols denote
3. what is the value of each free variable

Semantics

The truth of a Σ-formula is determined by an **interpretation** $\mathcal{I}$ of Σ consisting of:

1. For each sort $\sigma \in \Sigma^S$, a nonempty set called the **domain** of σ, written $dom(\sigma)$
2. A mapping from each n-ary function symbol f in Σ^F of sort $sort(f) = \langle \sigma_1, \ldots, \sigma_n, \sigma_{n+1} \rangle$ to $f^{\mathcal{I}}$, an n-ary function from $dom(\sigma_1) \times \cdots \times dom(\sigma_n)$ to $dom(\sigma_{n+1})$
3. A mapping from each variable v of sort σ to its interpretation $v^{\mathcal{I}}$, an element of $dom(\sigma)$

Note 1: We always assume $dom(\mathrm{Bool}) = \{\mathtt{T}, \mathtt{F}\}$
Note 2: We always assume $\bot^{\mathcal{I}} = \mathtt{F}$, $\top^{\mathcal{I}} = \mathtt{T}$.
Note 3: We always define the equality symbol $=_\sigma$ as $f(x, y) = \mathtt{T}$ iff $x = y$.

Note 4: (1) and (2) is called a **structure** or a **model**.

Semantics: Example

Consider a signature $\Sigma = \langle \Sigma^S, \Sigma^F \rangle$ for a fragment of set theory:
$\Sigma^S = \{E, S\}$, $\Sigma^F = \{\varnothing, \epsilon\}$, $sort(\varnothing) = S$ and $sort(\epsilon) = \langle E, S, \text{Bool} \rangle$
We have a set of variables $\{v_i^e\}$ of sort E, and a set of variables $\{v_i^s\}$ of sort S

Consider the following interpretation $\mathcal{I}$ for this signature:

1. $dom(E) = dom(S) = \mathcal{N}$, the set of natural numbers
2. $\epsilon^{\mathcal{I}} = <$, and $\varnothing^{\mathcal{I}} = 0$
3. $v_i^{e\mathcal{I}} = i$ and $v_i^{s\mathcal{I}} = 0$, for $i = 0, 1, \ldots$

What do these Σ-formulas mean in this interpretation? Are the formulas true in the interpretation?

- $\epsilon\, v_1^e v_2^s$
- $\exists\, v_1^e\, \forall\, v_1^s\, \neg \epsilon\, v_1^s\;\, v_1^e$

Semantics

An interpretations $\mathcal{I}$ is analogous to a variable assignment in propositional logic.

We now define how to determine the truth value of a Σ-formula given $\mathcal{I}$, which is analogous to determining the truth value of a formula given a variable assignment in propositional logic.

The first step is to give meaning to well-sorted terms based on $\mathcal{I}$

We define an **evaluation function** e from well-sorted terms and interpretations to $(\bigcup_{\sigma \in \Sigma^S} dom(\sigma))$:

- For each variable v, $e(v, \mathcal{I}) = v^{\mathcal{I}}$
- For each constant f, $e(f, \mathcal{I}) = f^{\mathcal{I}}$
- For a well-sorted term $t := ft_1 \dots t_n$, $e(f\ t_1, \dots, t_n, \mathcal{I}) = f^{\mathcal{I}}(e(t_1, \mathcal{I}), \dots, e(t_n, \mathcal{I}))$

Semantics

Given e, we define a function $\overline{e}$ from Σ-formulas and interpretations to $\{1, 0\}$:

- For each atomic formula α, $\overline{e}(\alpha, \mathcal{I}) = 1$ iff $e(\alpha, \mathcal{I}) = \mathtt{T}$
- $\overline{e}(\neg\alpha, \mathcal{I}) = 1 - \overline{e}(\alpha, \mathcal{I})$
- $\overline{e}(\alpha \to \beta, \mathcal{I}) = \max(1 - \overline{e}(\alpha, \mathcal{I}), \overline{e}(\beta, \mathcal{I}))$
- $\overline{e}(\forall\, v\, \alpha, \mathcal{I}) = 1$ iff $\overline{e}(\alpha, \mathcal{I}(v \,|\, d)) = 1$ for every $d \in \mathit{dom}(\sigma)$ where $\sigma = \mathit{sort}(v)$.

$\mathcal{I}(v \mid d)$ signifies the interpretation that is the same as $\mathcal{I}$ everywhere except that it maps variable v to d.

The following are the same:

- $e(\alpha, \mathcal{I}) = 1$
- $\mathcal{I} \vDash \alpha$
- α is true in $\mathcal{I}$
- $\mathcal{I}$ **satisfies** α

Logical implication, validity

Let Γ be a set of Σ-formulas. We write $\mathcal{I} \models \Gamma$ to signify that $\mathcal{I} \models \alpha$ for every $\alpha \in \Gamma$.

If Γ is a set of Σ-formulas and α is a Σ-formula, then Γ **logically implies** α, written $\Gamma \models \alpha$, iff for every interpretation $\mathcal{I}$ of Σ, if $\mathcal{I} \models \Gamma$ then $\mathcal{I} \models \alpha$.

We write $\beta \models \alpha$ as an abbreviation for $\{\beta\} \models \alpha$.

β and α are **logically equivalent** (written $\beta \models\!\!\!\!\models\!\!\dashv \alpha$) iff $\beta \models \alpha$ and $\alpha \models \beta$.

A Σ-formula α is **valid**, written $\models \alpha$ iff $\mathcal{I} \models \alpha$ for every interpretation $\mathcal{I}$.

Suppose that $\Sigma^S = \{\sigma\}$, $\Sigma^F = \{P, Q\}$, $sort(P) = \langle \sigma, \text{Bool} \rangle$, $sort(Q) = \langle \sigma, \sigma, \text{Bool} \rangle$, and all variables have sort σ. Do the following statements hold?

1. $\forall\, v_1\, Pv_1 \models Pv_2$
2. $Pv_1 \models \forall\, v_1\, Pv_1$
3. $\forall\, v_1\, Pv_1 \models \exists\, v_2\, Pv_2$
4. $\exists\, v_2\, \forall\, v_1\, Qv_1 v_2 \models \forall\, v_1\, \exists\, v_2\, Qv_1 v_2$
5. $\forall\, v_1\, \exists\, v_2\, Qv_1 v_2 \models \exists\, v_2\, \forall\, v_1\, Qv_1 v_2$
6. $\models \exists\, v_1\, (Pv_1 \rightarrow \forall\, v_2\, Pv_2)$

Exercise

The truth of a Σ-formula is determined by an **interpretation** $\mathcal{I}$ of Σ consisting of:

1. For each sort $\sigma \in \Sigma^S$, a nonempty set called the **domain** of σ, written $dom(\sigma)$
2. A mapping from each n-ary function symbol f in Σ^F of sort $sort(f) = \langle \sigma_1, \ldots, \sigma_n, \sigma_{n+1} \rangle$ to $f^{\mathcal{I}}$, an n-ary function from $dom(\sigma_1) \times \cdots \times dom(\sigma_n)$ to $dom(\sigma_{n+1})$
3. A mapping from each variable v of sort σ to its interpretation $v^{\mathcal{I}}$, an element of $dom(\sigma)$

Consider the signature where
$\Sigma^S = \{\sigma\}, \Sigma^F = \{Q, =_\sigma\}$,
$sort(x) = sort(y) = \sigma, sort(Q) = \langle \sigma, \sigma, \text{Bool} \rangle$.

For each of the following Σ-formulas, describe an interpretation that satisfies it.

1. $\forall\, x\, \forall\, y\ =_\sigma x\ y$
2. $\forall\, x\, \forall\, y\ Qxy$
3. $\forall\, x\, \exists\, y\ Qxy$

Submit one of your interpretations to

https://pollev.com/andreww095

Exercise

The truth of a Σ-formula is determined by an **interpretation** $\mathcal{I}$ of Σ consisting of:

1. For each sort $\sigma \in \Sigma^S$, a nonempty set called the **domain** of σ, written $dom(\sigma)$
2. A mapping from each n-ary function symbol f in Σ^F of sort $sort(f) = \langle \sigma_1, \ldots, \sigma_n, \sigma_{n+1} \rangle$ to $f^{\mathcal{I}}$, an n-ary function from $dom(\sigma_1) \times \cdots \times dom(\sigma_n)$ to $dom(\sigma_{n+1})$
3. A mapping from each variable v of sort σ to its interpretation $v^{\mathcal{I}}$, an element of $dom(\sigma)$

Consider the signature where

$\Sigma^S = \{\sigma\}, \Sigma^F = \{Q, =_\sigma\}$,

$sort(x) = sort(y) = \sigma, sort(Q) = \langle \sigma, \sigma, \mathrm{Bool} \rangle$.

For each of the following Σ-formulas, describe an interpretation that satisfies it.

1. $\forall\, x\, \forall\, y\ =_\sigma x\ y$ — $dom(\sigma)$ has one element
2. $\forall\, x\, \forall\, y\ Qxy$ — $Q^{\mathcal{I}}(a, b) = \mathtt{T}$ for $(a, b) \in dom(\sigma)^2$
3. $\forall\, x\, \exists\, y\ Qxy$ — for each $a \in dom(\sigma)$, there is $b \in dom(\sigma)$ with $Q^{\mathcal{I}}(a, b) = \mathtt{T}$

Invariance of truth values

Theorem: Given a signature Σ, suppose two Σ-intepretations, $\mathcal{I}_1$ and $\mathcal{I}_2$ have the same structure and agree at all variables (if any) which occur free in the **wff** α. Then $\mathcal{I}_1 \models \alpha$ iff $\mathcal{I}_2 \models \alpha$.

Proof: We call the evaluation functions for $\mathcal{I}_1$ and $\mathcal{I}_2$, e_1 and e_2, respectively. The proof is by induction on well-formed formulas α.

1. If α is an atomic formula, then all variables in α occur free. Thus $\mathcal{I}_1$ and $\mathcal{I}_2$ agree on all variables in α. It follows that $e_1(t) = e_2(t)$ for each term t in α (technically we should prove this by induction too). The result follows.
2. If α is $(\neg\alpha)$ or $(\alpha \rightarrow \beta)$, the result is immediate from the inductive hypothesis.
3. Suppose $\alpha = \forall\, v\, \beta$.The variables free in α are the same as those free in β except for v. For any d in $dom(sort(v))$, $\mathcal{I}_1(v|d)$ and $\mathcal{I}_2(v|d)$ agree at all variables free in β. The result follows from the inductive hypothesis.

As a corollary of this theorem, we have that for sentences, satisfaction is independent of the variable assignment.

Notational conventions for formulas

From now on, in order to improve readability, we allow ourselves to use the infix notation for logical operators and functions that are typically written using infix.

We may also add a period immediately after a quantifier and its variable for clarity.

Example $\forall\, x.\, \forall\, y.\, x = y$ instead of $\forall\, x\, \forall\, y\ = x\ y$

We can also omit parentheses by defining precedence:

- Precedence for propositional logic still applies
- Quantifiers has the highest precedence after $\neg$.

Example $\neg\forall x.\, Px \wedge Qx$ reads $(\neg(\forall x.\, Px))) \wedge Qx)$

Finally, we will allow the use of parentheses following function symbols.

Example $\forall x.\, p(r(x)) \wedge q(x)$ instead of $\forall x.\, p\, r\, x \wedge q\, x$

Prenex Normal Form (PNF)

We now define some useful syntactic restrictions to first-order logic.

A formula is in **prenex normal form** (PNF) iff it is of the form

$$Q_1 x_1. \cdots Q_n x_n. \alpha,$$

where each Q_i is a quantifier and α is a **quantifier-free formula.**

We say the formula is in **prenex conjunctive normal form** (PCNF) iff in addition α is in conjunctive normal form (when replacing every atomic formula with a propositional variable).

Example: the following formula is in PCNF:

$$\forall y. \exists z. ((p(f(y)) \vee q(z)) \wedge (\neg q(z) \vee q(y)))$$

Clausal Form

We say a first-order logic formula is in **Clausal Form**, iff

1. it is in PCNF;
2. it is closed (i.e., does not contain free variables);
3. it only contains universal quantifiers.

Example: Are the following formulas in Clausal Form?

- $\forall y. \exists z. (p(f(y)) \land \neg q(y, z))$
- $\forall y. \forall z. (p(f(y)) \land \neg q(x, z))$
- $\forall y. \forall z. (p(f(y)) \land \neg q(y, z))$

Clausal Form: transformation

Skolem's Theorem: Any sentence can be transformed into an equi-satisfiable formula in clausal form.

The high level transformation strategy is as follows:

$$\text{Sentence} \Rightarrow \text{PNF} \Rightarrow \text{PCNF} \Rightarrow \text{Clausal Form}$$

We use the following formula as a running example:

$$(\forall x.(p(x) \to q(x))) \to (\forall x.p(x) \to \forall x.q(x))$$

I: Transforming into PNF

Any sentence can be transformed into a logically equivalent formula in PNF in 4 steps.

$$(\forall x.(p(x) \to q(x))) \to (\forall x.p(x) \to \forall x.q(x))$$

Step 1: Rename the bounded variables s.t. 1) the bounded variables are disjoint from free variables; 2) no variable appears in two quantifiers.

$$(\forall x.(p(x) \to q(x))) \to (\forall y.p(y) \to \forall z.q(z))$$

I: Transforming into PNF

$$(\forall x.(p(x) \to q(x))) \to (\forall y.p(y) \to \forall z.q(z))$$

Step 2: Eliminate all occurrences of $\to$ and $\leftrightarrow$ using the following logical equivalence:

- $\alpha_1 \leftrightarrow \alpha_2 \models\!\!\!\!\!\dashv (\alpha_1 \to \alpha_2) \land (\alpha_2 \to \alpha_1)$
- $\alpha_1 \to \alpha_2 \models\!\!\!\!\!\dashv \neg\alpha_1 \lor \alpha_2$

$$(\forall x.(\neg p(x) \lor q(x))) \to (\forall y.p(y) \to \forall z.q(z))$$
$$(\forall x.(\neg p(x) \lor q(x))) \to (\neg\forall y.p(y) \lor \forall z.q(z))$$
$$\neg(\forall x.(\neg p(x) \lor q(x))) \lor (\neg\forall y.p(y) \lor \forall z.q(z))$$

I: Transforming into PNF

$$(\neg\forall x.(\neg p(x) \vee q(x))) \vee (\neg\forall y.p(y)) \vee \forall z.q(z)$$

Step 3: Collapse double negations and move all negations inward until they apply only to atomic formulas using:

- $\neg\neg\alpha \models\Dashv \alpha$
- De Morgan's Laws
- $\neg\forall x.\alpha \models\Dashv \exists x.\neg\alpha$
- $\neg\exists x.\alpha \models\Dashv \forall x.\neg\alpha$

$$(\exists x.\neg(\neg p(x) \vee q(x))) \vee (\exists y.\neg p(y)) \vee \forall z.q(z)$$
$$(\exists x.(p(x) \wedge \neg q(x))) \vee (\exists y.\neg p(y)) \vee \forall z.q(z)$$

I: Transforming into PNF

$$(\exists x.(p(x) \land \neg q(x))) \lor (\exists y.\neg p(y)) \lor \forall z.q(z)$$

Step 4: Move all quantifiers to the front of the formula using:

- $\alpha \circ Qx.\beta \vDash\Dashv Qx.(\alpha \circ \beta)$ if x does not occur in α
- $(Qx.\alpha) \circ \beta \vDash\Dashv Qx.(\alpha \circ \beta)$ if x does not occur in β

Note: $\circ \in \{\land, \lor\}$

The equivalence says if a formula α's truth value does not depend on x then one is allowed to quantify over x.

$$\forall z.\exists x.\exists y.((p(x) \land \neg q(x)) \lor \neg p(y) \lor q(z))$$

II: Transforming into PCNF

Transformation from PNF to an logically equivalent PCNF is straightforward. We can treat each atomic formula as a propositional symbol and apply the distributive laws from propositional logic.

$$\forall z.\exists x.\exists y.((p(x) \land \neg q(x)) \lor \neg p(y) \lor q(z))$$

becomes

$$\forall z.\exists x.\exists y.((p(x) \lor \neg p(y) \lor q(z)) \land (\neg q(x) \lor \neg p(y) \lor q(z))$$

This formula contains existential quantifiers and is therefore not in clausal form.

We say a first-order logic formula is in **Clausal Form**, iff

1. it is in PCNF;
2. it is closed (i.e., does not contain free variables);
3. it only contains universal quantifiers.

III: Transforming into Clausal Form (Skolemization)

$$\forall z.\exists x.\exists y.((p(x) \vee \neg p(y) \vee q(z)) \wedge (\neg q(x) \vee \neg p(y) \vee q(z))$$

For every existential quantifier $\exists x$ in the PCNF, let $y_1, \ldots, y_n$ be the universally quantified variables preceding $\exists x$.

We introduce a new function symbol f_x to our signature Σ with arity n and $sort(f_x) = \langle sort(y_1), \ldots sort(y_n), sort(x) \rangle$. Delete $\exists x$ and replace any occurrence of x by $f_x(y_1, \ldots, y_n)$.

For our running example, introduce unary functions f_x and f_y for $\exists x$ and $\exists y$, respectively.

$$\forall z.((p(f_x(z)) \vee \neg p(f_y(z)) \vee q(z)) \wedge (\neg q(f_x(z)) \vee \neg p(f_y(z)) \vee q(z)),$$

These functions are called **Skolem functions** and the process of replacing existential quantifiers by functions is called **Skolemization**.

Note: Technically, the resulting formula is no longer a Σ-formula, but a Σ'-formula, where $\Sigma'^S = \Sigma^S$ and $\Sigma'^F = \Sigma^F \cup \{f\}$

Clausal Form

Skolem's Theorem: Any sentence can be transformed into an equi-satisfiable formula in clausal form.

The transformation procedure we just described serves as a proof of Skolem's Theorem (modulo proofs that the steps are satisfiability-preserving).

For details about the proof, see Chapter 9.2 of "Mathematical Logic for Computer Science (3rd Edition)" by Mordechai Ben-Ari.

Clausal Form

As with propositional logic, we can write a formula in clause form unambiguously as a set of clauses, e.g.:

$$\forall z.((p(f(z)) \vee \neg p(g(z)) \vee q(z)) \wedge (\neg q(f(z)) \vee \neg p(g(z)) \vee q(z)),$$

can be written as

$$\Delta := \{\{p(f(z)), \neg p(g(z)), q(z)\}, \{\neg q(f(z)), \neg p(g(z)), q(z)\}\}$$

We could lift the propositional resolution to the first-order logic:

$$\frac{l \in C_1 \quad \neg l \in C_2 \quad C_1, C_2 \in \Delta}{\Delta \cup \{(C_1 - \{l\}) \cup (C_2 - \{\neg l\})\}} \ (\textbf{prop. resolution})$$

where l is a literal (i.e., an atomic formula or its negation).

Example Consider $\Delta := \{\{p(f(z)), q(z)\}, \{\neg p(f(z)), \neg p(g(z))\}\}$

$\{q(z), \neg p(g(z))\}$ is a resolvant of the two clauses. Δ and $\Delta \cup \{\{q(z), \neg p(g(z))\}\}$ are equivalent.

First-order resolution

$$\frac{l \in C_1 \quad \neg l \in C_2 \quad C_1, C_2 \in \Delta}{\Delta \cup \{(C_1 - \{l\}) \cup (C_2 - \{\neg l\})\}} \text{ (prop. resolution)}$$

where l is a **literal** (i.e., an atomic formula or its negation).

Now consider $\Delta := \{\{\neg P\,z, Q\,z\}, \{P\,a\}, \{\neg Q\,a\}\}$, where z is a universally quantified variable, and a is a constant.

Is $P\,z$ equal to $P\,a$?

So we can instantiate the literals to make them equal and then perform resolution

First-order resolution: Unification

A substitution θ is a map from variables to well-sorted terms (of matching sorts)

Note: we assume the term does not contain any variables

We write $t\theta$ for the literal we get by replacing variables in t according to θ

Example: Let $\theta := \{z \mapsto a\}$, then $(p(g(z, z)))\theta = p(g(a, a))$

We use $\{l_1, \ldots, l_n\}\theta$ to represent $\{l_1\theta, \ldots, l_n\theta\}$

A substitution θ is a **unifier** of two terms s and t if $t\theta = s\theta$

Can there be more than one unifier of two terms?

Can there be no unifier of two terms?

First-order resolution

Now we can write first-order resolution as

$$\frac{l_1 \in C_1 \quad \neg l_2 \in C_2 \quad C_1, C_2 \in \Delta \quad \theta \text{ is a unifier of } l_1, l_2}{\Delta \cup \{(C_1 - \{l\}) \cup (C_2 - \{\neg l\})\}\theta} \text{ (\textbf{First-order resolution})}$$

Example: $\{\neg P\, z, Q\, z\}, \{P\, a\}, \{\neg Q\, a\}$

First-order resolution

Now we can write first-order resolution as

$$\frac{l_1 \in C_1 \quad \neg l_2 \in C_2 \quad C_1, C_2 \in \Delta \quad \theta \text{ is a unifier of } l_1, l_2}{\Delta \cup \{(C_1 - \{l\}) \cup (C_2 - \{\neg l\})\}\theta} \text{ (**First-order resolution**)}$$

Example
$$\frac{\{\neg P\, z, Q\, z\}, \{P\, a\}, \{\neg Q\, a\}}{\{\neg P\, z, Q\, z\}, \{P\, a\}, \{\neg Q\, a\}, \{Q\, a\}} \ (\theta := \{z \mapsto a\} \text{ unifies } P\, z \text{ and } P\, a)$$

First-order resolution

Now we can write first-order resolution as

$$\frac{l_1 \in C_1 \quad \neg l_2 \in C_2 \quad C_1, C_2 \in \Delta \quad \theta \text{ is a unifier of } l_1, l_2}{\Delta \cup \{(C_1 - \{l\}) \cup (C_2 - \{\neg l\})\}\theta} \text{ (**First-order resolution**)}$$

Example

$$\dfrac{\dfrac{\{\neg P\, z, Q\, z\}, \{P\, a\}, \{\neg Q\, a\}}{\{\neg P\, z, Q\, z\}, \{P\, a\}, \{\neg Q\, a\}, \{Q\, a\}} \; (\theta := \{z \mapsto a\} \text{ unifies } P\, z \text{ and } \{P\, a\})}{\{\neg P\, z, Q\, z\}, \{P\, a\}, \{\neg Q\, a\}, \{Q\, a\}, \{\}} \; (\text{Resolve } \{\neg Q\, a\}, \{Q\, a\})$$

Therefore, $\alpha := \forall z.((\neg P\, z \vee Q\, z) \wedge P\, a \wedge \neg Q\, a)$ is unsatisfiable.

What do we know about $\neg\alpha$?

This suggests a strategy to prove the validity of a Σ-formula α:

1. Negate the formula;
2. Transform into Clausal Form;
3. Apply first-order resolution until an empty clause is derived (might not terminate!)