3 First-Order Logic

First-order logic

- is expressive:
  can be used to formalize mathematical concepts,
  can be used to encode Turing machines,
  but cannot axiomatize natural numbers or uncountable sets,
- has important decidable fragments,
- has interesting logical properties (model and proof theory).

First-order logic is also called (first-order) *predicate logic*.

3.1 Syntax

Syntax:

- non-logical symbols (domain-specific)
  $\Rightarrow$ terms, atomic formulas
- logical connectives (domain-independent)
  $\Rightarrow$ Boolean combinations, quantifiers

Signatures

A signature $\Sigma = (\Omega, \Pi)$ fixes an alphabet of non-logical symbols, where

- Ω is a set of *function symbols* f with *arity* $n \geq 0$, written $\text{arity}(f) = n$,
- Π is a set of *predicate symbols* P with arity $m \geq 0$, written $\text{arity}(P) = m$.

Function symbols are also called *operator symbols*.
If $n = 0$ then f is also called a *constant (symbol)*.
If $m = 0$ then P is also called a *propositional variable*.

We will usually use

b, c, d for constant symbols,

f, g, h for non-constant function symbols,

P, Q, R, S for predicate symbols.

Convention: We will usually write $f/n \in \Omega$ instead of $f \in \Omega$, $\text{arity}(f) = n$ (analogously for predicate symbols).

Refined concept for practical applications:
many-sorted signatures (corresponds to simple type systems in programming languages); no big change from a logical point of view.

Variables

Predicate logic admits the formulation of abstract, schematic assertions. (Object) variables are the technical tool for schematization.

We assume that X is a given countably infinite set of symbols which we use to denote *variables*.

Terms

Terms over Σ and X (Σ-terms) are formed according to these syntactic rules:

$$\begin{array}{llll} s,t,u,v & ::= & x & ,\ x \in X & \text{(variable)} \\ & \mid & f(s_1,...,s_n) & ,\ f/n \in \Omega & \text{(functional term)} \end{array}$$

By $\mathrm{T}_\Sigma(X)$ we denote the set of Σ-terms (over X). A term not containing any variable is called a *ground term*. By T_Σ we denote the set of Σ-ground terms.

Atoms

Atoms (also called atomic formulas) over Σ are formed according to this syntax:

$$\begin{array}{llll} A,B & ::= & P(s_1,\ldots,s_m) & ,\ P/m \in \Pi \quad \text{(non-equational atom)} \\ & \Big[\ \mid & (s \approx t) & \qquad\qquad\qquad \text{(equation)}\ \Big] \end{array}$$

Whenever we admit equations as atomic formulas we are in the realm of *first-order logic with equality*. Admitting equality does not really increase the expressiveness of first-order logic (see next chapter). But deductive systems where equality is treated specifically are much more efficient.

Literals

$$\begin{array}{llll} L & ::= & A & \text{(positive literal)} \\ & \mid & \neg A & \text{(negative literal)} \end{array}$$

Clauses

$$\begin{array}{llll} C,D & ::= & \bot & \text{(empty clause)} \\ & \mid & L_1 \vee \ldots \vee L_k,\ \ k \geq 1 & \text{(non-empty clause)} \end{array}$$

General First-Order Formulas

$F_\Sigma(X)$ is the set of *first-order formulas* over Σ defined as follows:

$$
\begin{array}{lcll}
F, G, H & ::= & \bot & \text{(falsum)} \\
 & \mid & \top & \text{(verum)} \\
 & \mid & A & \text{(atomic formula)} \\
 & \mid & \neg F & \text{(negation)} \\
 & \mid & (F \wedge G) & \text{(conjunction)} \\
 & \mid & (F \vee G) & \text{(disjunction)} \\
 & \mid & (F \rightarrow G) & \text{(implication)} \\
 & \mid & (F \leftrightarrow G) & \text{(equivalence)} \\
 & \mid & \forall x\, F & \text{(universal quantification)} \\
 & \mid & \exists x\, F & \text{(existential quantification)}
\end{array}
$$

Notational Conventions

We omit parentheses according to the conventions for propositional logic.

$\forall x_1, \dots, x_n\, F$ and $\exists x_1, \dots, x_n\, F$ abbreviate $\forall x_1 \dots \forall x_n\, F$ and $\exists x_1 \dots \exists x_n\, F$.

We use infix-, prefix-, postfix-, or mixfix-notation with the usual operator precedences.

Examples:

$$
\begin{array}{ccc}
s + t * u & \text{for} & +(s, *(t, u)) \\
s * u \leq t + v & \text{for} & \leq(*(s, u), +(t, v)) \\
-s & \text{for} & -(s) \\
s! & \text{for} & !(s) \\
|s| & \text{for} & |\_|(s) \\
0 & \text{for} & 0()
\end{array}
$$

Example: Peano Arithmetic

$$
\begin{array}{lcl}
\Sigma_{\mathrm{PA}} & = & (\Omega_{\mathrm{PA}},\ \Pi_{\mathrm{PA}}) \\
\Omega_{\mathrm{PA}} & = & \{0/0,\ +/2,\ */2,\ s/1\} \\
\Pi_{\mathrm{PA}} & = & \{</2\}
\end{array}
$$

Examples of formulas over this signature are:

$\forall x, y\, ((x < y \vee x \approx y) \leftrightarrow \exists z\, (x + z \approx y))$
$\exists x \forall y\, (x + y \approx y)$
$\forall x, y\, (x * s(y) \approx x * y + x)$
$\forall x, y\, (s(x) \approx s(y) \rightarrow x \approx y)$
$\forall x \exists y\, (x < y \wedge \neg \exists z (x < z \wedge z < y))$

Positions in Terms and Formulas

The set of positions is extended from propositional logic to first-order logic:

The *positions* of a term s (formula F):

$$\begin{aligned}
&\text{pos}(x) = \{\varepsilon\},\\
&\text{pos}(f(s_1,\dots,s_n)) = \{\varepsilon\} \cup \textstyle\bigcup_{i=1}^{n} \{\, ip \mid p \in \text{pos}(s_i)\,\},\\
&\text{pos}(P(t_1,\dots,t_n)) = \{\varepsilon\} \cup \textstyle\bigcup_{i=1}^{n} \{\, ip \mid p \in \text{pos}(t_i)\,\},\\
&\text{pos}(\forall x\, F) = \{\varepsilon\} \cup \{\, 1p \mid p \in \text{pos}(F)\,\},\\
&\text{pos}(\exists x\, F) = \{\varepsilon\} \cup \{\, 1p \mid p \in \text{pos}(F)\,\}.
\end{aligned}$$

The prefix order $\leq$, the subformula (subterm) operator, the formula (term) replacement operator and the size operator are extended accordingly. See the definitions in Sect. 2.

Variables

The *set of variables* occurring in a term t is denoted by $\text{var}(t)$ (and analogously for atoms, literals, clauses, and formulas).

Bound and Free Variables

In $\mathsf{Q}x\,F$, $\mathsf{Q} \in \{\exists, \forall\}$, we call F the *scope* of the quantifier $\mathsf{Q}x$. An *occurrence* of a variable x is called *bound*, if it is inside the scope of a quantifier $\mathsf{Q}x$. Any other occurrence of a variable is called *free*.

Formulas without free variables are called *closed formulas* (or *sentential forms*).

Formulas without variables are called *ground*.

Example:

$$\forall y \quad \overbrace{((\forall x \quad \overbrace{P(x)}^{\text{scope of } \forall x} \quad) \quad \rightarrow \quad R(x, y))}^{\text{scope of } \forall y}$$

The occurrence of y is bound, as is the first occurrence of x. The second occurrence of x is a free occurrence.

Substitutions

Substitution is a fundamental operation on terms and formulas that occurs in all inference systems for first-order logic.

Substitutions are mappings

$$\sigma : X \to \mathrm{T}_\Sigma(X)$$

such that the *domain* of σ, that is, the set

$$\mathrm{dom}(\sigma) = \{\, x \in X \mid \sigma(x) \neq x \,\},$$

is finite. The set of variables *introduced* by σ, that is, the set of variables occurring in one of the terms $\sigma(x)$, with $x \in \mathrm{dom}(\sigma)$, is denoted by $\mathrm{codom}(\sigma)$.

Substitutions are often written as $\{x_1 \mapsto s_1, \dots, x_n \mapsto s_n\}$, with x_i pairwise distinct, and then denote the mapping

$$\{x_1 \mapsto s_1, \dots, x_n \mapsto s_n\}(y) = \begin{cases} s_i, & \text{if } y = x_i \\ y, & \text{otherwise} \end{cases}$$

We also write $x\sigma$ for $\sigma(x)$.

The *modification* of a substitution σ at x is defined as follows:

$$\sigma[x \mapsto t](y) = \begin{cases} t, & \text{if } y = x \\ \sigma(y), & \text{otherwise} \end{cases}$$

Why Substitution is Complicated

We define the application of a substitution σ to a term t or formula F by structural induction over the syntactic structure of t or F by the equations below.

In the presence of quantification it is surprisingly complex: We must not only ensure that bound variables are not replaced by σ. We must also make sure that the (free) variables in the codomain of σ are not *captured* upon placing them into the scope of a quantifier $\mathsf{Q}y$. Hence the bound variable must be renamed into a “fresh”, that is, previously unused, variable z.

Application of a Substitution

"Homomorphic" extension of σ to terms and formulas:

$$
\begin{aligned}
f(s_1,\dots,s_n)\sigma &= f(s_1\sigma,\dots,s_n\sigma)\\
\bot\sigma &= \bot\\
\top\sigma &= \top\\
P(s_1,\dots,s_n)\sigma &= P(s_1\sigma,\dots,s_n\sigma)\\
(u \approx v)\sigma &= (u\sigma \approx v\sigma)\\
\neg F\sigma &= \neg(F\sigma)\\
(F \circ G)\sigma &= (F\sigma \,\circ\, G\sigma) \quad \text{for each binary connective } \circ\\
(\mathsf{Q}x\,F)\sigma &= \mathsf{Q}z\,(F\,\sigma[x \mapsto z]) \quad \text{with } z \text{ a fresh variable}
\end{aligned}
$$

If $s = t\sigma$ for some subsitution σ, we call the term s an *instance* of the term t, and we call t a *generalization* of s (analogously for formulas).

3.2 Semantics

To give semantics to a logical system means to define a notion of truth for the formulas. The concept of truth that we will now define for first-order logic goes back to Tarski.

As in the propositional case, we use a two-valued logic with truth values "true" and "false" denoted by 1 and 0, respectively.

Algebras

A Σ-*algebra* (also called Σ-interpretation or Σ-structure) is a triple

$$\mathcal{A} = (U_{\mathcal{A}},\ (f_{\mathcal{A}} : U_{\mathcal{A}}^n \to U_{\mathcal{A}})_{f/n\in\Omega},\ (P_{\mathcal{A}} \subseteq U_{\mathcal{A}}^m)_{P/m\in\Pi})$$

where $U_{\mathcal{A}} \neq \emptyset$ is a set, called the *universe* of $\mathcal{A}$.

By Σ-*Alg* we denote the class of all Σ-algebras.

Σ-algebras generalize the valuations from propositional logic.

Assignments

A variable has no intrinsic meaning. The meaning of a variable has to be defined externally (explicitly or implicitly in a given context) by an assignment.

A *(variable) assignment* (over a given Σ-algebra $\mathcal{A}$), is a function $\beta : X \to U_{\mathcal{A}}$.

Variable assignments are the semantic counterparts of substitutions.

Value of a Term in $\mathcal{A}$ with respect to β

By structural induction we define

$$\mathcal{A}(\beta) : \mathrm{T}_\Sigma(X) \to U_{\mathcal{A}}$$

as follows:

$$\begin{aligned}
\mathcal{A}(\beta)(x) &= \beta(x), & x \in X\\
\mathcal{A}(\beta)(f(s_1,\dots,s_n)) &= f_{\mathcal{A}}(\mathcal{A}(\beta)(s_1),\dots,\mathcal{A}(\beta)(s_n)), & f/n \in \Omega
\end{aligned}$$

In the scope of a quantifier we need to evaluate terms with respect to modified assignments. To that end, let $\beta[x \mapsto a] : X \to U_{\mathcal{A}}$, for $x \in X$ and $a \in U_{\mathcal{A}}$, denote the assignment

$$\beta[x \mapsto a](y) = \begin{cases} a & \text{if } x = y \\ \beta(y) & \text{otherwise} \end{cases}$$

Truth Value of a Formula in $\mathcal{A}$ with respect to β

$\mathcal{A}(\beta) : \mathrm{F}_\Sigma(X) \to \{0, 1\}$ is defined inductively as follows:

$$\begin{aligned}
\mathcal{A}(\beta)(\bot) &= 0\\
\mathcal{A}(\beta)(\top) &= 1\\
\mathcal{A}(\beta)(P(s_1,\dots,s_n)) &= \text{if } (\mathcal{A}(\beta)(s_1),\dots,\mathcal{A}(\beta)(s_n)) \in P_{\mathcal{A}} \text{ then } 1 \text{ else } 0\\
\mathcal{A}(\beta)(s \approx t) &= \text{if } \mathcal{A}(\beta)(s) = \mathcal{A}(\beta)(t) \text{ then } 1 \text{ else } 0\\
\mathcal{A}(\beta)(\neg F) &= 1 - \mathcal{A}(\beta)(F)\\
\mathcal{A}(\beta)(F \wedge G) &= \min(\mathcal{A}(\beta)(F), \mathcal{A}(\beta)(G))\\
\mathcal{A}(\beta)(F \vee G) &= \max(\mathcal{A}(\beta)(F), \mathcal{A}(\beta)(G))\\
\mathcal{A}(\beta)(F \to G) &= \max(1 - \mathcal{A}(\beta)(F), \mathcal{A}(\beta)(G))\\
\mathcal{A}(\beta)(F \leftrightarrow G) &= \text{if } \mathcal{A}(\beta)(F) = \mathcal{A}(\beta)(G) \text{ then } 1 \text{ else } 0\\
\mathcal{A}(\beta)(\forall x\, F) &= \min_{a \in U_{\mathcal{A}}} \{\mathcal{A}(\beta[x \mapsto a])(F)\}\\
\mathcal{A}(\beta)(\exists x\, F) &= \max_{a \in U_{\mathcal{A}}} \{\mathcal{A}(\beta[x \mapsto a])(F)\}
\end{aligned}$$

Example

The "Standard" interpretation for Peano arithmetic:

$$
\begin{array}{rcl}
U_{\mathbb{N}} & = & \{0, 1, 2, \ldots\} \\
0_{\mathbb{N}} & = & 0 \\
s_{\mathbb{N}} & : & n \mapsto n + 1 \\
+_{\mathbb{N}} & : & (n, m) \mapsto n + m \\
*_{\mathbb{N}} & : & (n, m) \mapsto n * m \\
<_{\mathbb{N}} & = & \{\, (n, m) \mid n \text{ less than } m \,\}
\end{array}
$$

Note that $\mathbb{N}$ is just one out of many possible Σ_{PA}-interpretations.

Values over $\mathbb{N}$ for sample terms and formulas:

Under the assignment $\beta : x \mapsto 1, y \mapsto 3$ we obtain

$$
\begin{array}{lcl}
\mathbb{N}(\beta)(s(x) + s(0)) & = & 3 \\
\mathbb{N}(\beta)(x + y \approx s(y)) & = & 1 \\
\mathbb{N}(\beta)(\forall x, y\, (x + y \approx y + x)) & = & 1 \\
\mathbb{N}(\beta)(\forall z\, (z < y)) & = & 0 \\
\mathbb{N}(\beta)(\forall x \exists y\, (x < y)) & = & 1
\end{array}
$$

Ground Terms and Closed Formulas

If t is a ground term, then $\mathcal{A}(\beta)(t)$ does not depend on β, that is, $\mathcal{A}(\beta)(t) = \mathcal{A}(\beta')(t)$ for every β and β'.

Analogously, if F is a closed formula, then $\mathcal{A}(\beta)(F)$ does not depend on β, that is, $\mathcal{A}(\beta)(F) = \mathcal{A}(\beta')(F)$ for every β and β'.

An element $a \in U_{\mathcal{A}}$ is called *term-generated*, if $a = \mathcal{A}(\beta)(t)$ for some ground term t.

In general, not every element of an algebra is term-generated.

3.3 Models, Validity, and Satisfiability

F is *true* in $\mathcal{A}$ under assignment β:

$$\mathcal{A}, \beta \models F \quad :\Leftrightarrow \quad \mathcal{A}(\beta)(F) = 1$$

F is *true* in $\mathcal{A}$ ($\mathcal{A}$ is a *model* of F; F is *valid* in $\mathcal{A}$):

$$\mathcal{A} \models F \quad :\Leftrightarrow \quad \mathcal{A}, \beta \models F \ \text{ for all } \beta \in X \to U_{\mathcal{A}}$$

F is *valid* (or is a *tautology*):

$$\models F \quad :\Leftrightarrow \quad \mathcal{A} \models F \ \text{ for all } \mathcal{A} \in \Sigma\text{-Alg}$$

F is called *satisfiable* if there exist $\mathcal{A}$ and β such that $\mathcal{A}, \beta \models F$. Otherwise F is called *unsatisfiable*.

Entailment and Equivalence

F *entails* (*implies*) G (or *G is a consequence of F*), written $F \models G$, if for all $\mathcal{A} \in \Sigma$-Alg and $\beta \in X \to U_{\mathcal{A}}$, we have

$$\mathcal{A}, \beta \models F \quad \Rightarrow \quad \mathcal{A}, \beta \models G$$

F and G are called *equivalent*, written $F \models\mid G$, if for all $\mathcal{A} \in \Sigma$-Alg and $\beta \in X \to U_{\mathcal{A}}$ we have

$$\mathcal{A}, \beta \models F \quad \Leftrightarrow \quad \mathcal{A}, \beta \models G$$

Proposition 3.1 *$F \models G$ if and only if $(F \to G)$ is valid*

Proof. ($\Rightarrow$) Suppose that $(F \to G)$ is not valid. Then there exist an algebra $\mathcal{A}$ and an assignment β such that $\mathcal{A}(\beta)(F \to G) = 0$, which means that $\mathcal{A}(\beta)(F) = 1$ and $\mathcal{A}(\beta)(G) = 0$, or in other words $\mathcal{A}, \beta \models F$ but not $\mathcal{A}, \beta \models G$. Consequently, $F \models G$ does not hold.

($\Leftarrow$) Suppose that $F \models G$ does not hold. Then there exist an algebra $\mathcal{A}$ and an assignment β such that $\mathcal{A}, \beta \models F$ but not $\mathcal{A}, \beta \models G$. Therefore $\mathcal{A}(\beta)(F) = 1$ and $\mathcal{A}(\beta)(G) = 0$, which implies $\mathcal{A}(\beta)(F \to G) = 0$, so $(F \to G)$ is not valid. □

Proposition 3.2 *$F \models\mid G$ if and only if $(F \leftrightarrow G)$ is valid.*

Extension to sets of formulas N as in propositional logic, e. g.:

$N \models F \quad :\Leftrightarrow \quad$ for all $\mathcal{A} \in \Sigma$-Alg and $\beta \in X \to U_{\mathcal{A}}$:
if $\mathcal{A}, \beta \models G$ for all $G \in N$, then $\mathcal{A}, \beta \models F$.

Validity vs. Unsatisfiability

Validity and unsatisfiability are just two sides of the same medal as explained by the following proposition.

Proposition 3.3 *Let F and G be formulas, let N be a set of formulas. Then*

(i) F *is valid if and only if* $\neg F$ *is unsatisfiable.*

(ii) $F \models G$ *if and only if* $F \wedge \neg G$ *is unsatisfiable.*

(iii) $N \models G$ *if and only if* $N \cup \{\neg G\}$ *is unsatisfiable.*

Hence in order to design a theorem prover (validity checker) it is sufficient to design a checker for unsatisfiability.

Substitution Lemma

Lemma 3.4 *Let $\mathcal{A}$ be a Σ-algebra, let β be an assignment, let σ be a substitution. Then for any Σ-term t*

$$\mathcal{A}(\beta)(t\sigma) = \mathcal{A}(\beta \circ \sigma)(t),$$

where $\beta \circ \sigma : X \to U_{\mathcal{A}}$ is the assignment $\beta \circ \sigma(x) = \mathcal{A}(\beta)(x\sigma)$.

Proof. We use induction over the structure of Σ-terms.

If $t = x$, then $\mathcal{A}(\beta \circ \sigma)(x) = \beta \circ \sigma(x) = \mathcal{A}(\beta)(x\sigma)$ by definition of $\beta \circ \sigma$.

If $t = f(t_1, \dots, t_n)$, then $\mathcal{A}(\beta \circ \sigma)(f(t_1, \dots, t_n)) = f_{\mathcal{A}}(\mathcal{A}(\beta \circ \sigma)(t_1), \dots, \mathcal{A}(\beta \circ \sigma)(t_n)) = f_{\mathcal{A}}(\mathcal{A}(\beta)(t_1\sigma), \dots, \mathcal{A}(\beta)(t_n\sigma)) = \mathcal{A}(\beta)(f(t_1\sigma, \dots, t_n\sigma)) = \mathcal{A}(\beta)(f(t_1, \dots, t_n)\sigma)$ by induction. ☐

Proposition 3.5 *Let $\mathcal{A}$ be a Σ-algebra, let β be an assignment, let σ be a substitution. Then for every Σ-formula F*

$$\mathcal{A}(\beta)(F\sigma) = \mathcal{A}(\beta \circ \sigma)(F)\,.$$

Corollary 3.6 $\mathcal{A}, \beta \models F\sigma \quad \Leftrightarrow \quad \mathcal{A}, \beta \circ \sigma \models F$

These theorems basically express that the syntactic concept of substitution corresponds to the semantic concept of an assignment.

Two Lemmas

Lemma 3.7 *Let $\mathcal{A}$ be a Σ-algebra. Let F be a Σ-formula with free variables $x_1, \ldots, x_n$. Then*

$$\mathcal{A} \models \forall x_1, \ldots, x_n\, F \quad \textit{if and only if} \quad \mathcal{A} \models F\,.$$

Proof. ($\Rightarrow$) Suppose that $\mathcal{A} \models \forall x_1, \ldots, x_n\, F$, that is, $\mathcal{A}(\beta)(\forall x_1, \ldots, x_n\, F) = 1$ for all assignments β. By definition, that means

$$\min_{a_1,\ldots,a_n \in U_\mathcal{A}} \{\mathcal{A}(\beta[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n])(F)\} = 1,$$

and therefore $\mathcal{A}(\beta[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n])(F) = 1$ for all $a_1, \ldots, a_n \in U_\mathcal{A}$.

Let γ be an arbitrary assigmnment. We have to show that $\mathcal{A}(\gamma)(F) = 1$. For every $i \in \{1, \ldots, n\}$ define $a_i = \gamma(x_i)$, then $\gamma = \gamma[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n]$, and therefore $\mathcal{A}(\gamma)(F) = \mathcal{A}(\gamma[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n])(F) = 1$.

($\Leftarrow$) Suppose that $\mathcal{A} \models F$, that is, $\mathcal{A}(\gamma)(F) = 1$ for all assignments γ.

Then in particular $\mathcal{A}(\beta[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n])(F) = 1$ for all $a_1, \ldots, a_n \in U_\mathcal{A}$ (take $\gamma = \beta[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n]$). Therefore

$$\mathcal{A}(\beta)(\forall x_1, \ldots, x_n\, F) = \min_{a_1,\ldots,a_n \in U_\mathcal{A}} \{\mathcal{A}(\beta[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n])(F)\} = 1.$$

□

Note that it is not possible to replace $\mathcal{A} \models \ldots$ by $\mathcal{A}, \beta \models \ldots$ in Lemma 3.7.

Lemma 3.8 *Let $\mathcal{A}$ be a Σ-algebra. Let F be a Σ-formula with free variables $x_1, \ldots, x_n$. Let σ be a substitution and let $y_1, \ldots, y_m$ be the free variables of $F\sigma$. Then*

$$\mathcal{A} \models \forall x_1, \ldots, x_n\, F \quad \textit{implies} \quad \mathcal{A} \models \forall y_1, \ldots, y_m\, F\sigma\,.$$

Proof. By the previous lemma, we have $\mathcal{A} \models \forall x_1, \ldots, x_n\, F$ if and only if $\mathcal{A} \models F$ and similarly $\mathcal{A} \models \forall y_1, \ldots, y_m\, F\sigma$ if and only if $\mathcal{A} \models F\sigma$. So it suffices to show that $\mathcal{A} \models F$ implies $\mathcal{A} \models F\sigma$. Suppose that $\mathcal{A} \models F$, that is, $\mathcal{A}(\beta)(F) = 1$ for all assignments β. Then for every assignment γ, we have by Prop. 3.5 $\mathcal{A}(\gamma)(F\sigma) = \mathcal{A}(\gamma \circ \sigma)(F) = 1$ (take $\beta = \gamma \circ \sigma$), and therefore $\mathcal{A} \models F\sigma$. □

3.4 Algorithmic Problems

Validity(F): $\models F$?

Satisfiability(F): F satisfiable?

Entailment(F,G): does F entail G?

Model($\mathcal{A}$,F): $\mathcal{A} \models F$?

Solve($\mathcal{A}$,F): find an assignment β such that $\mathcal{A}, \beta \models F$.

Solve(F): find a substitution σ such that $\models F\sigma$.

Abduce(F): find G with "certain properties" such that $G \models F$.

Theory of an Algebra

Let $\mathcal{A} \in \Sigma$-Alg. The *(first-order) theory* of $\mathcal{A}$ is defined as

$$\mathrm{Th}(\mathcal{A}) = \{\, G \in \mathrm{F}_\Sigma(X) \mid \mathcal{A} \models G \,\}$$

Problem of axiomatizability:

Given an algebra $\mathcal{A}$ (or a class of algebras) can one *axiomatize* Th($\mathcal{A}$), that is, can one write down a formula F (or a recursively enumerable set F of formulas) such that

$$\mathrm{Th}(\mathcal{A}) = \{\, G \mid F \models G \,\}?$$

Two Interesting Theories

Let $\Sigma_{\mathrm{Pres}} = (\{0/0, s/1, +/2\},\ \{<\})$ and $\mathbb{N}_+ = (\mathbb{N}, 0, s, +, <)$ its standard interpretation on the natural numbers. Th($\mathbb{N}_+$) is called *Presburger arithmetic* (M. Presburger, 1929). (There is no essential difference when one, instead of $\mathbb{N}$, considers the integer numbers $\mathbb{Z}$ as standard interpretation.)

Presburger arithmetic is decidable in 3EXPTIME (D. Oppen, JCSS, 16(3):323–332, 1978), and in 2EXPSPACE, using automata-theoretic methods (and there is a constant $c \geq 0$ such that $\mathrm{Th}(\mathbb{Z}_+) \notin \mathrm{NTIME}(2^{2^{cn}})$).

However, $\mathbb{N}_* = (\mathbb{N}, 0, s, +, *, <)$, the standard interpretation of $\Sigma_{\mathrm{PA}} = (\{0/0, s/1, +/2, */2\}, \{<\})$, has as theory the so-called *Peano arithmetic* which is undecidable and not even recursively enumerable.

(Non-)Computability Results

1. For most signatures Σ, validity is undecidable for Σ-formulas.
   (One can easily encode Turing machines in most signatures.)
2. Gödel's completeness theorem:
   For each signature Σ, the set of valid Σ-formulas is recursively enumerable.
   (We will prove this by giving complete deduction systems.)
3. Gödel's incompleteness theorem:
   For $\Sigma = \Sigma_{\mathrm{PA}}$ and $\mathbb{N}_* = (\mathbb{N}, 0, s, +, *, <)$, the theory $\mathrm{Th}(\mathbb{N}_*)$ is not recursively enumerable.

These complexity results motivate the study of subclasses of formulas (*fragments*) of first-order logic

Some Decidable Fragments

Some decidable fragments:

- *Monadic class*: no function symbols, all predicates unary; validity is NEXPTIME-complete.
- Variable-free formulas without equality: satisfiability is NP-complete. (why?)
- Variable-free Horn clauses (clauses with at most one positive atom): entailment is decidable in linear time.
- Finite model checking is decidable in exponential time and PSPACE-complete.

3.5 Normal Forms and Skolemization

Study of normal forms motivated by

- reduction of logical concepts,
- efficient data structures for theorem proving.

The main problem in first-order logic is the treatment of quantifiers. The subsequent normal form transformations are intended to eliminate many of them.

Prenex Normal Form (Traditional)

Prenex formulas have the form

$$\mathsf{Q}_1 x_1 \ldots \mathsf{Q}_n x_n \; F,$$

where F is quantifier-free and $\mathsf{Q}_i \in \{\forall, \exists\}$; we call $\mathsf{Q}_1 x_1 \ldots \mathsf{Q}_n x_n$ the *quantifier prefix* and F the *matrix* of the formula.

Computing prenex normal form by the reduction system $\Rightarrow_P$:

$$\begin{array}{rcl}
H[(F \leftrightarrow G)]_p & \Rightarrow_P & H[(F \rightarrow G) \wedge (G \rightarrow F)]_p \\
H[\neg \mathsf{Q}x\, F]_p & \Rightarrow_P & H[\overline{\mathsf{Q}}x\, \neg F]_p \\
H[((\mathsf{Q}x\, F) \;\circ\; G)]_p & \Rightarrow_P & H[\mathsf{Q}y\, (F\{x \mapsto y\} \;\circ\; G)]_p, \\
 & & \quad \circ \in \{\wedge, \vee\} \\
H[((\mathsf{Q}x\, F) \rightarrow G)]_p & \Rightarrow_P & H[\overline{\mathsf{Q}}y\, (F\{x \mapsto y\} \rightarrow G)]_p, \\
H[(F \;\circ\; (\mathsf{Q}x\, G))]_p & \Rightarrow_P & H[\mathsf{Q}y\, (F \;\circ\; G\{x \mapsto y\})]_p, \\
 & & \quad \circ \in \{\wedge, \vee, \rightarrow\}
\end{array}$$

Here y is always assumed to be some fresh variable and $\overline{\mathsf{Q}}$ denotes the quantifier *dual* to Q, i. e., $\overline{\forall} = \exists$ and $\overline{\exists} = \forall$.

Skolemization

Intuition: replacement of $\exists y$ by a concrete choice function computing y from all the arguments y depends on.

Transformation $\Rightarrow_S$
(to be applied outermost, *not* in subformulas):

$$\forall x_1, \ldots, x_n \exists y\, F \quad \Rightarrow_S \quad \forall x_1, \ldots, x_n\, F\{y \mapsto f(x_1, \ldots, x_n)\}$$

where f/n is a new function symbol (*Skolem function*).

Together: $F \Rightarrow_P^* \underbrace{G}_{\text{prenex}} \Rightarrow_S^* \underbrace{H}_{\text{prenex, no } \exists}$

Theorem 3.9 *Let F, G, and H as defined above and closed. Then*

(i) F and G are equivalent.

(ii) $H \models G$ but the converse is not true in general.

(iii) G satisfiable (w. r. t. Σ-Alg) $\Leftrightarrow$ H satisfiable (w. r. t. Σ'-Alg) where $\Sigma' = (\Omega \cup SKF, \Pi)$ if $\Sigma = (\Omega, \Pi)$.

The Complete Picture

$$
\begin{array}{llll}
F & \Rightarrow_P^* & \mathsf{Q}_1 y_1 \ldots \mathsf{Q}_n y_n \, G & (G \text{ quantifier-free}) \\
 & \Rightarrow_S^* & \forall x_1, \ldots, x_m \, H & (m \leq n,\ H \text{ quantifier-free}) \\
 & \Rightarrow_{CNF}^* & \underbrace{\underbrace{\forall x_1, \ldots, x_m}_{\text{leave out}} \ \bigwedge_{i=1}^{k} \underbrace{\bigvee_{j=1}^{n_i} L_{ij}}_{\text{clauses } C_i}}_{F'} &
\end{array}
$$

$N = \{C_1, \ldots, C_k\}$ is called the *clausal (normal) form* (CNF) of F.
Note: The variables in the clauses are implicitly universally quantified.

Theorem 3.10 *Let F be closed. Then $F' \models F$. (The converse is not true in general.)*

Theorem 3.11 *Let F be closed. Then F is satisfiable if and only if F' is satisfiable if and only if N is satisfiable*

Optimization

The normal form algorithm described so far leaves lots of room for optimization. Note that we only can preserve satisfiability anyway due to Skolemization.

- the size of the CNF is exponential when done naively; the transformations we introduced already for propositional logic avoid this exponential growth;
- we want to preserve the original formula structure;
- we want small arity of Skolem functions (see next section).

3.6 Getting Skolem Functions with Small Arity

A clause set that is better suited for automated theorem proving can be obtained using the following steps:

- eliminate trivial subformulas
- replace beneficial subformulas
- produce a negation normal form (NNF)
- apply miniscoping
- rename all variables
- Skolemize
- push quantifiers upward
- apply distributivity

We start with a closed formula.

Elimination of Trivial Subformulas

Eliminate subformulas $\top$ and $\bot$ essentially as in the propositional case modulo associativity/commutativity of $\wedge$, $\vee$:

$$
\begin{aligned}
H[(F \wedge \top)]_p &\Rightarrow_{\text{OCNF}} H[F]_p \\
H[(F \vee \bot)]_p &\Rightarrow_{\text{OCNF}} H[F]_p \\
H[(F \leftrightarrow \bot)]_p &\Rightarrow_{\text{OCNF}} H[\neg F]_p \\
H[(F \leftrightarrow \top)]_p &\Rightarrow_{\text{OCNF}} H[F]_p \\
H[(F \vee \top)]_p &\Rightarrow_{\text{OCNF}} H[\top]_p \\
H[(F \wedge \bot)]_p &\Rightarrow_{\text{OCNF}} H[\bot]_p \\
H[\neg\top]_p &\Rightarrow_{\text{OCNF}} H[\bot]_p \\
H[\neg\bot]_p &\Rightarrow_{\text{OCNF}} H[\top]_p \\
H[(F \rightarrow \bot)]_p &\Rightarrow_{\text{OCNF}} H[\neg F]_p \\
H[(F \rightarrow \top)]_p &\Rightarrow_{\text{OCNF}} H[\top]_p \\
H[(\bot \rightarrow F)]_p &\Rightarrow_{\text{OCNF}} H[\top]_p \\
H[(\top \rightarrow F)]_p &\Rightarrow_{\text{OCNF}} H[F]_p \\
H[\mathsf{Q}x\, \top]_p &\Rightarrow_{\text{OCNF}} H[\top]_p \\
H[\mathsf{Q}x\, \bot]_p &\Rightarrow_{\text{OCNF}} H[\bot]_p
\end{aligned}
$$

Replacement of Beneficial Subformulas

The functions ν and $\bar{\nu}$ that give us an overapproximation for the number of clauses generated by a formula are extended to quantified formulas by

$$\nu(\forall x\, F) = \nu(\exists x\, F) = \nu(F),$$
$$\bar{\nu}(\forall x\, F) = \bar{\nu}(\exists x\, F) = \bar{\nu}(F).$$

The other cases are defined as for propositional formulas.

Introduce top-down fresh predicates for beneficial subformulas:

$$H[F]_p \;\Rightarrow_{\mathrm{OCNF}}\; H[P(x_1, \dots, x_n)]_p \wedge \mathrm{def}(H, p, P, F)$$

if $\nu(H[F]_p) > \nu(H[P(\dots)]_p \wedge \mathrm{def}(H, p, P, F))$,

where $\{x_1, \dots, x_n\}$ are the free variables in F, P/n is a predicate new to $H[F]_p$, and $\mathrm{def}(H, p, P, F)$ is defined by

$$\forall x_1, \dots, x_n\, (P(x_1, \dots, x_n) \rightarrow F), \text{ if } \mathrm{pol}(H, p) = 1,$$
$$\forall x_1, \dots, x_n\, (F \rightarrow P(x_1, \dots, x_n)), \text{ if } \mathrm{pol}(H, p) = -1,$$
$$\forall x_1, \dots, x_n\, (P(x_1, \dots, x_n) \leftrightarrow F), \text{ if } \mathrm{pol}(H, p) = 0.$$

As in the propositional case, one can test $\nu(H[F]_p) > \nu(H[P]_p \wedge \mathrm{def}(H, p, P, F))$ in constant time without actually computing ν.

Negation Normal Form (NNF)

Apply the reduction system $\Rightarrow_{\mathrm{NNF}}$:

$$H[F \leftrightarrow G]_p \;\Rightarrow_{\mathrm{NNF}}\; H[(F \rightarrow G) \wedge (G \rightarrow F)]_p$$

if $\mathrm{pol}(H, p) = 1$ or $\mathrm{pol}(H, p) = 0$.

$$H[F \leftrightarrow G]_p \;\Rightarrow_{\mathrm{NNF}}\; H[(F \wedge G) \vee (\neg G \wedge \neg F)]_p$$

if $\mathrm{pol}(H, p) = -1$.

$$\begin{aligned}
H[F \rightarrow G]_p &\;\Rightarrow_{\mathrm{NNF}}\; H[\neg F \vee G]_p \\
H[\neg\neg F]_p &\;\Rightarrow_{\mathrm{NNF}}\; H[F]_p \\
H[\neg(F \vee G)]_p &\;\Rightarrow_{\mathrm{NNF}}\; H[\neg F \wedge \neg G]_p \\
H[\neg(F \wedge G)]_p &\;\Rightarrow_{\mathrm{NNF}}\; H[\neg F \vee \neg G]_p \\
H[\neg \mathsf{Q} x\; F]_p &\;\Rightarrow_{\mathrm{NNF}}\; H[\bar{\mathsf{Q}} x\; \neg F]_p
\end{aligned}$$

Miniscoping

Apply the reduction system $\Rightarrow_{\text{MS}}$ modulo associativity and commutativity of $\wedge$, $\vee$. For the rules below we assume that x occurs freely in F, F', but x does not occur freely in G:

$$\begin{aligned}
H[\mathsf{Q}x\,(F \wedge G)]_p &\Rightarrow_{\text{MS}} H[(\mathsf{Q}x\,F) \wedge G]_p \\
H[\mathsf{Q}x\,(F \vee G)]_p &\Rightarrow_{\text{MS}} H[(\mathsf{Q}x\,F) \vee G]_p \\
H[\forall x\,(F \wedge F')]_p &\Rightarrow_{\text{MS}} H[(\forall x\,F) \wedge (\forall x\,F')]_p \\
H[\exists x\,(F \vee F')]_p &\Rightarrow_{\text{MS}} H[(\exists x\,F) \vee (\exists x\,F')]_p \\
H[\mathsf{Q}x\,G]_p &\Rightarrow_{\text{MS}} H[G]_p
\end{aligned}$$

Variable Renaming

Rename all variables in H such that there are no two different positions p, q with $H|_p = \mathsf{Q}x\,F$ and $H|_q = \mathsf{Q}'x\,G$.

Standard Skolemization

Apply the reduction system:

$$H[\exists x\,F]_p \Rightarrow_{\text{SK}} H[F\{x \mapsto f(y_1, \ldots, y_n)\}]_p$$

where p has minimal length,
$\{y_1, \ldots, y_n\}$ are the free variables in $\exists x\,F$,
and f/n is a new function symbol to H.

Final Steps

Apply the reduction system modulo commutativity of $\wedge$, $\vee$ to push $\forall$ upward:

$$\begin{aligned}
H[(\forall x\,F) \wedge G]_p &\Rightarrow_{\text{OCNF}} H[\forall x\,(F \wedge G)]_p \\
H[(\forall x\,F) \vee G]_p &\Rightarrow_{\text{OCNF}} H[\forall x\,(F \vee G)]_p
\end{aligned}$$

Note that variable renaming ensures that x does not occur in G.

Apply the reduction system modulo commutativity of $\wedge$, $\vee$ to push disjunctions downward:

$$H[(F \wedge F') \vee G]_p \Rightarrow_{\text{CNF}} H[(F \vee G) \wedge (F' \vee G)]_p$$

3.7 Herbrand Interpretations

From now on we shall consider FOL without equality. We assume that Ω contains at least one constant symbol.

An *Herbrand interpretation* (over Σ) is a Σ-algebra $\mathcal{A}$ such that

- $U_{\mathcal{A}} = \mathrm{T}_\Sigma$ (= the set of ground terms over Σ)
- $f_{\mathcal{A}} : (s_1, \ldots, s_n) \mapsto f(s_1, \ldots, s_n),\ f/n \in \Omega$

In other words, *values are fixed* to be ground terms and *functions are fixed* to be the *term constructors*. Only predicate symbols $P/m \in \Pi$ may be freely interpreted as relations $P_{\mathcal{A}} \subseteq \mathrm{T}_\Sigma^m$.

Proposition 3.12 *Every set of ground atoms I uniquely determines an Herbrand interpretation $\mathcal{A}$ via*

$$(s_1, \ldots, s_n) \in P_{\mathcal{A}} \quad \text{if and only if} \quad P(s_1, \ldots, s_n) \in I$$

Thus we shall identify Herbrand interpretations (over Σ) with sets of Σ-ground atoms.

Existence of Herbrand Models

An Herbrand interpretation I is called an *Herbrand model* of F, if $I \models F$.

The importance of Herbrand models lies in the following theorem, which we will prove later in this lecture:

Let N be a set of (universally quantified) Σ-clauses. Then the following properties are equivalent:

(1) N has a model.
(2) N has an Herbrand model (over Σ).
(3) $G_\Sigma(N)$ has an Herbrand model (over Σ).

where $G_\Sigma(N) = \{\, C\sigma \text{ ground clause} \mid (\forall \vec{x}\, C) \in N,\ \sigma : X \to \mathrm{T}_\Sigma \,\}$ is the set of *ground instances* of N.