Literals

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

Clauses

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

General First-Order Formulas

$\mathrm{F}_\Sigma(X)$ 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.

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

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

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

$+, *, <, \leq$ infix; $* \;>_p\; + \;>_p\; < \;>_p\; \leq$

Examples of formulas over this signature are:

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

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):

$\mathrm{pos}(x) = \{\varepsilon\}$,
$\mathrm{pos}(f(s_1, \dots, s_n)) = \{\varepsilon\} \cup \bigcup_{i=1}^{n} \{\, ip \mid p \in \mathrm{pos}(s_i)\, \}$,

$\mathrm{pos}(P(t_1, \dots, t_n)) = \{\varepsilon\} \cup \bigcup_{i=1}^{n} \{\, ip \mid p \in \mathrm{pos}(t_i)\, \}$,

$\mathrm{pos}(\forall x F) = \{\varepsilon\} \cup \{\, 1p \mid p \in \mathrm{pos}(F)\, \}$,
$\mathrm{pos}(\exists x F) = \{\varepsilon\} \cup \{\, 1p \mid p \in \mathrm{pos}(F)\, \}$.

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.

Bound and Free Variables

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

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

Formulas without variables are called *ground*.

Example:

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

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

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

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

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

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

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 depicted on the next page.

In the presence of quantification it is surprisingly complex: We need to make sure that the (free) variables in the codomain of σ are not *captured* upon placing them into the scope of a quantifier Qy, 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:

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

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.

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*, also called a *valuation* (over a given Σ-algebra $\mathcal{A}$), is a map $\beta : X \to U_{\mathcal{A}}$.

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:

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

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:

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

Example

The "Standard" Interpretation for Peano Arithmetic:

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

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

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

Ground Terms and Closed Formulas

If t is a ground term, then $\mathcal{A}(\beta)(t)$ does not depend on β:

$$\mathcal{A}(\beta)(t) = \mathcal{A}(\beta')(t)$$

for every β and β'.

Analogously, if F is a closed formula, then $\mathcal{A}(\beta)(F)$ does not depend on β:

$$\mathcal{A}(\beta)(F) = \mathcal{A}(\beta')(F)$$

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

Substitution Lemma

The following propositions, to be proved by structural induction, hold for all Σ-algebras $\mathcal{A}$, assignments β, and substitutions σ.

Lemma 3.1 *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 $\beta \circ \sigma(x) = \mathcal{A}(\beta)(x\sigma)$.

Proposition 3.2 *For any Σ-formula F, $\mathcal{A}(\beta)(F\sigma) = \mathcal{A}(\beta \circ \sigma)(F)$.*

Corollary 3.3 $\mathcal{A}, \beta \models F\sigma \quad \Leftrightarrow \quad \mathcal{A}, \beta \circ \sigma \models F$

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

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 $\beta \in X \to U_{\mathcal{A}}$, whenever $\mathcal{A}, \beta \models F$, then $\mathcal{A}, \beta \models G$.

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 3.4 *F entails G iff $(F \to G)$ is valid*

Proposition 3.5 *F and G are equivalent iff $(F \leftrightarrow G)$ is valid.*

Extension to sets of formulas N in the "natural way", e. g., $N \models F$

$:\Leftrightarrow$ 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 3.6 *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.