Supplements to Luger, 6th ed.

Propositional Calculus

The *symbols* will be

1. *propositional symbols*: $p_1, p_2, \ldots$, denoted generically by P, Q, R, S, etc.
2. *truth symbols*: **true**, **false**
3. *connectives*: $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\equiv$
4. *parentheses*: (and)

The *propositions* (or *statements*) are defined by:

1. Propositional symbols and truth symbols are propositions.
2. If P and Q are propositions, then so are $(\neg P)$, $(P \vee Q)$, $(P \wedge Q)$, $(P \rightarrow Q)$, and $(P \equiv Q)$.
3. Nothing is a proposition except as required by the above.

Parentheses are dropped when the intended grouping is clear. It is normally assumed that $\neg$ has priority over $\vee$ and $\wedge$, and that these have priority over $\rightarrow$ and $\equiv$.

An *interpretation* (sometimes called a *truth value assignment*) for the propositional calculus is a mapping $v : \text{propositions} \rightarrow \{\mathrm{T}, \mathrm{F}\}$ defined by:

1. $v(p_i) \in \{\mathrm{T}, \mathrm{F}\}$, for all i
2. $v(\textbf{true}) = \mathrm{T}$ and $v(\textbf{false}) = \mathrm{F}$
3. $v(\neg P) = \mathrm{T}$ iff $v(P) = \mathrm{F}$
4. $v(P \vee Q) = \mathrm{T}$ iff $v(P) = \mathrm{T}$ or $v(Q) = \mathrm{T}$
5. $v(P \wedge Q) = \mathrm{T}$ iff $v(P) = \mathrm{T}$ and $v(Q) = \mathrm{T}$
6. $v(P \rightarrow Q) = \mathrm{T}$ iff $v(P) = \mathrm{F}$ or $v(Q) = \mathrm{T}$
7. $v(P \equiv Q) = \mathrm{T}$ iff $v(P) = v(Q)$

Predicate Calculus

Luger adapts the syntax of Prolog. We begin with some *alpahabet characters* consisting of:

1. the English alphabet letters, both uppercase and lowercase,
2. the numerals $0, 1, \ldots, 9$, and
3. the underscore character, _.

A *symbol expression* is any string of alphabet characters that begin with an English letter. Then the *symbols* are:

1. *logical connectives*: $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\equiv$,

2. *punctuation*: left and right parentheses and comma,
3. *quantifiers*: $\forall$ and $\exists$,
4. *constant symbols*: symbol expressions beginning with a lowercase letter,
5. *variable symbols*: symbol expressions beginning with an uppercase letter,
6. *function symbols*: symbol expressions beginning with a lowercase letter (distinguished from constant symbols by context), each having an *arity* indicating the number of arguments,
7. *predicate symbols*: same as function symbols, also distinguished by context,
8. *truth symbols*: **true** and **false**.

The *terms* are defined by:

1. Constant symbols and variable symbols are terms.
2. If f is an n-ary function symbol and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.
3. Nothing is a term except as required by the above.

A term is *closed* if it does not contain variable symbols. The *sentences* (also called (*first-order*) *formulas*) are defined by:

1. If p is an n-ary predicate symbol and $t_1, \ldots, t_n$ are terms, then $p(t_1, \ldots, t_n)$ is a sentence, known as an *atomic* sentence. The truth symbols **true** and **false** also are atomic sentences.
2. If s_1 and s_2 are sentences, then so are $(\neg s_1)$, $(s_1 \wedge s_2)$, $(s_1 \vee s_2)$, $(s_1 \rightarrow s_2)$, and $(s_1 \equiv s_2)$.
3. If s is a sentence, then so are $\forall X s$ and $\exists X s$.
4. Nothing is a sentence except as required by the above.

A sentence is *closed* if either it does not contain variable symbols or all its variable symbols are within the scopes of quantifiers.

Different *first-order languages* are defined by different selections of constant symbols, function symbols, and predicate symbols. An *interpretation* I for a first-order language L consists of:

1. A nonempty set D_I serving as the *domain*, the elements of which are called *individuals*.
2. For each constant symbol a in L, assignment of a unique individual a_I in D_I. In this case, let $I(a)$ denote a_I.
3. For each variable symbol V in L, assignment of a subset V_I of D_I (to serve as the range for possible values of V).
4. For each n-ary function symbol f in L, assignment of a function $f_I : D_I^n \rightarrow D_I$.
5. For each n-ary predicate symbol p in L, assigment of a predicate (relation) p_I on D_I^n.

Given an interpretation I for a first-order language L, a *term valuation* is a mapping I : closed_terms $\rightarrow D_I$, having the properties:

1. If t is a constant symbol a, $I(t) = a_I$.
2. If t is a function expression $f(t_1, \ldots, t_n)$, then $I(t) = f_I(I(t_1), \ldots, I(t_n))$.

Given an interpretation I for a first-order language L, a *truth valuation* is a mapping v : closed_sentences $\rightarrow \{\mathrm{T}, \mathrm{F}\}$ defined by:

1. $v(\textbf{true}) = \mathrm{T}$ and $v(\textbf{false}) = \mathrm{F}$.
2. For s atomic, having the form $p(t_1, \ldots, t_n)$, $v(s) = \mathrm{T}$ iff p_I holds for $(I(t_1), \ldots, I(t_n))$.
3. For s of the form $\neg s'$, $v(s) = \mathrm{T}$ iff $v(s') = \mathrm{F}$.
4. For s of the form $s_1 \wedge s_2$, $v(s) = \mathrm{T}$ iff $v(s_1) = \mathrm{T}$ and $v(s_2) = \mathrm{T}$.
5. For s of the form $s_1 \vee s_2$, $v(s) = \mathrm{T}$ iff $v(s_1) = \mathrm{T}$ or $v(s_2) = \mathrm{T}$.
6. For s of the form $s_1 \rightarrow s_2$, $v(s) = \mathrm{T}$ iff $v(s_1) = \mathrm{F}$ or $v(s_2) = \mathrm{T}$.
7. For s of the form $s_1 \equiv s_2$, $v(s) = \mathrm{T}$ iff $v(s_1) = v(s_2)$.
8. For s of the form $\forall X s'$, $v(s) = \mathrm{T}$ iff $v(s')$ is true in D_I for every interpretation of X as an individual in D_I. (For simplicity, some details are blurred here.)
9. For s of the form $\exists X s'$, $v(s) = \mathrm{T}$ iff $v(s')$ is true in D_I for some interpretation of X as an individual in D_I. (For simplicity, some details are blurred here.)

A *most general unifier* (mgu) for a set of expressions **E** is any unifier **g** such that, if **s** is a unifier for **E**, then there exists a substitution **s**′ such that **s** = **gs**′. E.g., {fred/X,fred/Y} = {Z/X, Z/Y}{fred/Z}.