3 Declarative semantics of logic programs

We now precisely define the semantics, that is, the meaning of a Prolog program. As already remarked in the previous chapter, each clause in a program is to be viewed as an assertion. For example the clauses of the program SUM

\[
\text{sum}(X, 0, X).
\]

\[
\text{sum}(X, s(Y), s(Z)) :- \text{sum}(X, Y, Z).
\]

assert:

1. For all objects \(X\): \(\text{sum}(X, 0, X)\) holds.
2. For all objects \(X, Y, Z\): \(\text{sum}(X, s(Y), s(Z))\) holds if \(\text{sum}(X, Y, Z)\) holds.

In order to fully understand the meaning of these assertions we need to answer the following questions:

(a) What do we mean by ‘objects’? In other words: what is the universe of discourse the variables are ranging over?

(b) What are the interpretations of the constants and functors?

(c) What is the meaning of the predicates (in our example the predicate \(\text{sum}\))?
A similar question is: Given a program \(P\) and a predicate \(p\) occurring in \(P\), for which terms \(t_1, \ldots, t_n\) without variables is the atomic formula

\[ p(t_1, \ldots, t_n) \]

true?

As we will see, each of these questions can be answered in two different reasonable ways, and, consequently, we will have two different declarative semantics of Prolog programs.

3.1 Least Herbrand interpretation

From a computer science perspective it is probably most natural to answer the questions raised above as follows:

(a) Since we cannot expect Prolog to know our intended universe of discourse (in the program SUM it could be the natural numbers, the integers, the real numbers,
e.t.c) the objects should be constructed from the material at hand, namely the **object language** of the program, that is, the set $\mathcal{L}$ of constants and functors occurring in the program. This means that our universe of discourse is the set of **closed terms**, that is, terms without variables. In logic programming jargon closed terms are also called **ground terms** and the set of ground terms is called the **Herbrand universe** \(^2\). For our program **SUM** the Herbrand Universe would be the set

$$HU_{\{0,s\}} = \{0, s(0), s(s(0)), s(s(s(0))), \ldots\}$$

which naturally corresponds to the set of natural numbers

$$\mathbb{N} = \{0, 1, 2, 3, \ldots\}$$

(b) In the Herbrand universe constants are interpreted by themselves and a functor $f$ is interpreted as a **term constructor**, which means $f$ applied to arguments (= ground terms) $t_1, \ldots, t_n$ simply is the term $f(t_1, \ldots, t_n)$.

(c) It is not enough to just require the predicates to satisfy the clauses. For example, in our program **SUM** we could define $\text{sum}(t_1, t_2, t_3)$ to be true for all ground terms $t_1, t_2, t_3$ and thus satisfy clauses 1 and 2, but this is certainly not intended. What is intended, is to let an atomic formula $\text{sum}(t_1, t_2, t_3)$ be true if its truth is **forced** by the clauses, that is, if it can be derived by certain **inference rules** from the clauses.

In order to describe these inference rules, we need the following notion: A **ground instance** of a clause is obtained by simultaneously substituting every variable in the clause by a ground term.

For example, a ground instance of the clause

$$\text{sum}(X, s(Y), s(Z)) :- \text{sum}(X, Y, Z)$$

could be obtained by the substitution

$$X/s(0), Y/0, Z/s(0)$$

yielding

$$\text{sum}(s(0), s(0), s(s(0))) :- \text{sum}(s(0), 0, s(0))$$

\(^2\)Jacques Herbrand (1908-1931), French mathematician
The inference process can now be described by the following two rules:

(i) If \( A \) is a ground instance of a fact, then \( A \) can be inferred.

(ii) If \( A : B_1, \ldots, B_n \) is a ground instance of a rule, and \( B_1, \ldots, B_n \) have already been inferred, then \( A \) can be inferred.

Via these rules the clauses of a program \( P \) can viewed as an inference engine producing more and more inferred atomic formulas.

The set of all atomic formulas inferrible in this way is called the **least Herbrand model** of \( P \) and is denoted \( \mathcal{M}(P) \).

Hence in the least Herbrand model a closed atomic formula \( p(t_1, \ldots, t_n) \) is true if it is inferrible from the program; otherwise it is false. The stipulation that exactly those closed atomic formulas are true which can be inferred is called the **closed world assumption**.

The inference of a closed atomic formula can be visualized by an **inference tree**. For example, the inference tree for \( \text{sum}(s(0),s(0),s(s(0))) \) would be

\[
\text{sum}(s(0),s(s(0)),s(s(s(0)))) \\
\quad | \quad (ii) \\
\quad | \\
\text{sum}(s(0),s(0),s(s(0))) \\
\quad | \quad (ii) \\
\quad | \\
\text{sum}(s(0),0,s(s(0)))
\]

(i)

**Exercise.** Draw an inference tree for \( \text{connection}(\text{amsterdam},\text{brisbane}) \) w.r.t. the program FLIGHTS. How does the inference tree relate to Prolog’s search tree for the query \( ?- \text{connection}(\text{amsterdam},\text{brisbane}) \)?
Let us summarize the essentials of the least Herbrand interpretation:

In the least Herbrand interpretation of a program $P$ the universe of discourse is the Herbrand universe which is the set of all ground terms.

A closed atomic formula $p(t_1, \ldots, t_n)$ is true in the Herbrand interpretation if and only if it is in the least Herbrand model, $M(P)$, that is, if it can be inferred using the rules (i) and (ii) above.

3.2 Logical interpretation

In the logical interpretation of a program the questions (a), (b) and (c) are answered in a radically different way: Since a logic program neither specifies the universe of discourse nor defines the meaning of the constants of functors, it is reasonable (from a logical point of view) to consider all possible interpretations of constants functors and predicates that happen to satisfy the clauses of the program. This means we allow as a model of $P$ an arbitrary (nonempty) set as universe of discourse and arbitrary interpretations of the constants, functors and predicates that satisfy the program $P$.

A formula $A$ ($A$ could be an atomic formula, but also a more complex formula) is said to be a logical consequence of a program $P$ if $A$ holds in all models of $P$, that is, $A$ is true w.r.t. any interpretation of the constants, functors and predicates that satisfies $P$.

We say a formula $A$ is true w.r.t. a program $P$ in the logical interpretation if it is true in all models of $P$.

Although it might seem that because of its generality the notion of logical consequence is difficult to analyze, there exists in fact a rather simple proof calculus for deriving exactly the logical consequences of a program. It would be beyond the scope of this course describe this calculus, or to define more precisely the notion of logical consequence. Instead we refer the interested student to any textbook in logic, for example the monograph


which gives a good and easy to read introduction into formal logic. Formal logic will also be discussed in detail in other courses, for example `Programming with Abstract Data Types', (CS-376).

Let us summarize the essentials of the logical interpretation of a program:
In the logical interpretation all models of a program $P$, that is, all possible interpretations of the constants, functors and predicates that satisfy $P$ are considered. A formula is true in the logical interpretation if and only if it holds in all models of $P$.

3.3 Relations between least Herbrand interpretation and logical interpretation

Although the least Herbrand interpretation and the logical interpretation of a logic program are very different they surprisingly coincide with respect to closed atomic formulas:

**Theorem.**

A closed atomic formula is true in the least Herbrand interpretation of a program $P$ if and only if it is a logical consequence of $P$.

The following example shows that the least Herbrand interpretation and the logical interpretation differ for atomic formulas with variables. This difference will actually help us to better understand the behavior of Prolog:

Recall the program **SUM** and consider the following atomic formulas:

(a) $\text{sum}(X, 0, X)$ to be read as “for all $X$: $\text{sum}(X, 0, X)$”, and

(b) $\text{sum}(0, X, X)$ to be read as “for all $X$: $\text{sum}(0, X, X)$”.

Clearly (a) and (b) are both true in the Herbrand interpretation of **SUM** (which essentially is the set of natural numbers with 0, successor operation and addition).

Let us find out how Prolog reacts to the corresponding queries.

Query (a):

?- $\text{sum}(X, 0, X)$.

$X = _G222$;

No

The answer means that Prolog has confirmed (by a single computation) that $\text{sum}(X, 0, X)$ holds for all $X$.

Query (b):
?- sum(0,X,X).
X = 0;
X = s(0);
X = s(s(0));
.
.
This means that Prolog doesn't confirm that sum(X,0,X) holds for all X. Prolog only finds that this holds for X = 0, and if we ask for a second solution it finds X = s(0) (by a second computation), and so on.

Did we discover a weakness in Prolog's inference mechanism?

No! Prolog's behavior is perfectly reasonable from the logical point of view: While the atomic formula sum(X,0,X) is true in the logical interpretation the atomic formula sum(0,X,X) is not.

That sum(0,X,X) is not a logical consequence of the program P can be shown by the following interpretation that satisfies the P, but not sum(X,0,X):

Let the universe of discourse be the set of all integers, {...,−2,−1,0,1,2,...}, interpret the constant 0 by the integer 0, interpret the functor s by the identity function (which maps any number to itself), and define the meaning of the predicate sum by

\[ \text{sum}(n,m,k) \Leftrightarrow n - m = k \quad (n,m,k \text{ integers}) \]

Under this interpretation the clauses of the program SUM correspond to the following assertions:

1. For all n: n − 0 = n  
   \((\text{sum}(X,0,X))\)

2. For all n,m,k: n − m = k if n − m = k
   \((\text{sum}(X,s(Y),s(Z)) :- \text{sum}(X,Y,Z))\)
   remember that s is interpreted as the identity

which are obviously true. However the atomic formula \(\text{sum}(0,X,X)\) corresponds to the false statement

For all n : 0 − n = n