Chapter 2

Logic programming

This chapter briefly introduces the basic logic programming terminology that is used in this thesis. Complete definitions can be found in [34, 43, 48].

2.1 Syntax

A **program clause** (or clause) is a formula $A \leftarrow W$ for which A is an atom, the symbol $\leftarrow$ means an implication, and W is a conjunction of literals, i.e. positive or negative atoms. A is called **head** of the clause and W is called **body**. All variables in the head are universally quantified. A clause (or term) is **ground** if it contains no variable. The **closed clause** is a clause with no free occurrences of any variables, i.e. all variables are bound via quantifiers. We will use also the term **rule** for program clauses with unempty body. If W is absent we call the formula a **fact**. A **logic program**[1] is a finite set of program clauses. A **goal** is a clause of the form $\leftarrow W$ where W is a conjunction of literals. We will use very often :- instead of $\leftarrow$. Queries will be written $\leftarrow W$, or ?-W following syntax of Prolog. Thus

$$p(X, f(X)) :- r(X). \qquad (1)$$
$$p(X, g(Y)) :- p(X, Z). \qquad (2)$$
$$?\text{-}\ p(a, b), p(a, c). \qquad (3)$$

is a logic program in which (1),(2) are program clauses and (3) is a goal. The clause (1) is closed.

[1]If it is unambiguous, we will omit the adjective 'logic'.

The **first order language** given by an alphabet consists of the set of all well-formed formulas constructed from the symbols of the alphabet.

The **definition** of a predicate p appearing in a program P is the set of all program clauses in P which have p in their head. We will need the notion of completion of a program $Comp(P)$. Informally the completion of a program is obtained by replacing implication with equivalence in the rules, and adding the axioms of equality theory.

2.2 Semantics

An **pre-interpretation** J of a first order language L consists of the following:

1. A non-empty set D, called **domain**;
2. for each constant in L , the assignment of an element in D;
3. for each n-ary function symbol in L, the assignment of a mapping from D^n to D.

An **interpretation** I of a first order language L consists of

1. a pre-interpretation J with domain D, and
2. for each n-ary predicate symbol in L, the assignment of a mapping from D^n into $(true, false)$ (or, equivalently, a relation on D^n).

We assume that logical connectives as well as quantifiers have the ordinary semantics ([48] p. 13). Then a formula in L is given a truth value applying the definitions above. An interpretation I is a **model for a closed formula** F if F is true wrt I. Let S be a set of closed formulas and F be a closed formula of a language L. We say F is a **logical consequence** of S if, for every interpretation I of L, I is a model for S implies that I is a model for F. We will write $S \models F$. $F \models G$ iff every model of F is a model of G. I is a **model for the program** P iff I is a model for each clause in P. The **reduced clause** r **of a clause** c is a minimal subset of literals of c such that r and c has the same model.

Let L be a first order language. The **Herbrand universe** U_L for L is the set of all ground terms, which can be formed out of the constants and function symbols appearing in L. If L has no constants, we introduce one, to form ground terms. Let L be a first order language. The **Herbrand base** B_L for L is the set of all ground atoms which can be formed by using predicate symbols from L with ground terms from the Herbrand universe as arguments. An **Herbrand interpretation** for L is any interpretation on the domain of Herbrand universe U_L. Let L be a first order language and S a set of closed formulas of L. An **Herbrand model** for S is an Herbrand interpretation for L which is a model for S. For **normal programs**, i.e. programs with negative literals in the clause body, we refer to the model of the completion $Comp(P)$ of a program P.

2.3 Answer

A **substitution** θ is a finite set of the form $\{v_1/t_1, ..., v_n/t_n\}$, where $v_1, ..., v_n$ are distinct variables and each term t_i is distinct from v_i. Each element v_i/t_i is called **binding** for v_i. θ is called a **ground substitution** if the t_i are all ground terms. θ is called **variable-pure substitution** if the t_i are all variables. Let P be a normal program and G a goal $\leftarrow W$. An **answer** for $P \cup \{G\}$ is a substitution for variables in W. The answer does not necessarily contain a binding for every variable in G. A **correct answer** is an answer θ such that $\forall(W\theta)$ (all variables in $W\theta$ are universally quantified) is a logical consequence of $Comp(P)$.

2.4 Error diagnosis

A program LP **covers** a fact e if $LP \models e$, i.e. if in every possible interpretation I $LP \models_I e$. It is useful to split the set of predicate definitions LP into 2 parts. Let P be the definition of predicate of the same name and the same arity as the fact e, and $B = LP - P$. Than we can rewrite the above definition in the form $P \cup B \models e$. That notion of coverage is sometimes called **intensional coverage**. In machine learning there is often used the notion of **extensional coverage** of fact e [43]. In the latter all predicates in B are defined extensionally, i.e. by ground facts only.

An **intended interpretation** for a program P is a normal Herbrand interpretation [48] for the completion $Comp(P)$ of P. The aim of logic program synthesis is to find a program which has the intended interpretation as a model.

Let P be a program, G a goal $\leftarrow W$, and I an intended interpretation for P.

1. Program P is **correct wrt** I if I is a model for $Comp(P)$.
2. If θ is an answer for $P \cup \{G\}$ and $W\theta$ is not valid in I, then P is **inconsistent wrt** I.
3. If $P \cup \{G\}$ has a finitely failed SLDNF-tree[48] and W is satisfiable in I, then P is **incomplete wrt** I.

In the following text we will omit the clause "wrt I" if no ambiguity may arise. We say that an instance $A \leftarrow W$ of a program statement in P is an **incorrect statement instance** for P wrt I if A is unsatisfiable in I and W is valid in I. Then we can summarise that P is incorrect wrt I iff there is an uncovered atom for P wrt I or there is an incorrect statement instance for P wrt I.

2.5 Types and modes

We assume that every argument of a predicate have its **type** and an input/output **mode**. The type is a nonempty set of ground terms that the argument can take. The mode says whether the argument is input (its value must be known before an evaluation of the predicate) or output (the value is computed inside the predicate). Let us have $reverse/2$ predicate for reverting lists, e.g. ?-$reverse([k, a, b, a, t], X)$ returns $X = [t, a, b, a, k]$. For this predicate the type and mode definition can be written as $[+list, -list]$, i.e. the both arguments are lists, the mode of the first argument is input $(+)$ and the mode of the second one is output (-).