In this assignment, you will write a simple **parser** for the “object language” being a simple logic programming language (similar e.g., to a subset of Prolog).

The **parser** itself has to be written using the **OCaml** version of **Yacc**.

Below is an abstract definition of the logic programming language’s syntax.

You should use your learning of Prolog (Assignments 1 and 2) to write a **concrete grammatical specification** for the logic programming language corresponding to the abstract formulation below. The objective of the exercise is for you to define the **grammar** as well as the **lexical tokens** for your surface language. You will be evaluated on your design of the surface syntax as well as the implementation of the parser.

You may e.g., deviate from the standard Prolog syntax if you so wish (in particular, you may want to use the if-then-else construction or a generalisation of it).

You will need to represent an **abstract syntax** of the logic programming language as a data type in OCaml. Abstract syntax trees (terms in this data type) will be the output of your parser.

The abstract structure of a Prolog program is the following:

- A *program* is a sequence of *clauses*.
- A *clause* can either be a *fact* or a *rule*.
- A fact has a *head* but no *body*. A rule has a *head* and a *body*.
- The *head* is a single *atomic formula*.
- A *body* is a sequence of *atomic formulas*.
- An *atomic formula* is a *k*-ary *predicate symbol* followed by *k terms*.
- A *term* is either a *variable*, a *constant*, or a *k*-ary *function symbol* with *k* sub*term*s.
- A *goal* is a sequence of *atomic formulas*.