CAP 5602
Summer, 2011

Lesson 1: The Prolog Syntax

The topics

1. data objects in prolog: numbers, atoms, variables, structures, lists
2. clauses: facts, rules and predicates
3: programs

1. Data Objects

We will start the description of the prolog program bottom up, starting with the smllest units and working our way up to the program.

An **atom** is

a. a sequence of one or more letters (upper or lower case), numeral, and underscores that start with a lower case letter, or

b. any sequence of characters enclosed in single quotes, or

c. strings of one or more special characters form a list that includes + - * / > < = & # @ :

Examples of atoms:

john
month_of_May
'Bill'
+++

A **variable** is a sequence of one or more characters, numbers, and underscores that starts with an upper case letter or an underscore.

Examples of variables:

Author
_15A

All versions of Prolog allow the use of integers. They are represented as non-empty sequences of numerals from 0 to 9, optionally preceded by a + or - sign.

Examples of integers:

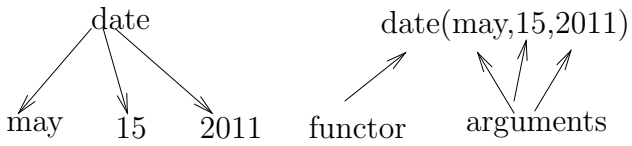

Figure 1: The tree representation of structures

56
023
+6
-24

SWI prolog allows the use decimal numbers, but you must specify the 0 before the fraction. You may also use the exponential notation.

Examples of decimal numbers:

5.5
+0.5
2E-3
-3e2

The **structures** allow us to form data structures. They have the syntax
$functor(t_1, \ldots, t_n)$
where $n \geq 1$.

The functor is an atom, and $t_1, \ldots, t_n$, are the **arguments** of the structure. The arguments can also be structures.

Examples of structures

date(may, 15, 2011)
address(23, 'Caca Street', 'Slumville')
person('Fat cat', address(100, 'Mansion Avenue', 'Nob Hill'), income(200000000))
point(2,-5.5)

The structures can be represented as trees, as shown in Figure 1.

The **lists** are a particular type of structures that will be discussed in a later lesson.

2. Clauses

The clauses can be either **facts** or **rules**. A fact is either a structure or an atom, followed by a period.

Examples of facts

likes('John', 'Mary').

rains.

likes(X,prolog) .

Notice that the period does not have to follow immediately the last character of the structure.

A rule has the form

head :- $c_1, c_2, \ldots, c_n$.

where $n \geq 1$, and head, c_1, ..., c_n are either structures or atoms. We call the head of the rule, and c_1, ..., c_n the body of the rule. For those of us who think semantically, head is the conclusion, :- is if, c_1, ..., c_n are the conditions and the commas are logical ands. So, we can read the above rule as

$(c_1 \wedge c_2 \wedge \ldots \wedge c_n) \longrightarrow head$

Examples of rules

animal(X) :- cat(X).

mother(X,Y) :- parent(X,Y), female(X).

Semantically, the first rule states that every cat is an animal, and the second one that X is the mother of Y, if X is a parent of Y, and X is a female.

Besides being case sensitive, prolog allows overloading. For example, the clauses shown can be used in the same program.

rains.

rains(date(may,14,2011), miami).

rains(often).

Here, rains can be an atom, or a functor with 1 or 2 arguments. To differentiate among them, we use the notation rains/0 , rains/1, rains/2 where the number that follows the / sign is the **arity** i.e. the number of arguments.

We refer to rains/0 , rains/1, rains/2 as **predicates**.

Many times it is useful to think of facts as rules that have heads but no bodies. Then, we say that a predicate is defined by all rules that have that

predicate as the head.

Example

The clauses below define the predicate path/2 in a directed graph.

```
path(X,X).
path(X,Y):- arc(X,Z), path(Z,Y).
```

3. Prolog programs

A prolog program consists of clauses, blank lines, and clauses.

/* insert a comment, that can take several lines */

or

% some comment

that can also be used as an end comment. The SWI compiler uses the later syntax.

The blank lines are used for readability and, just like the comments are ignored by the compiler.

For clarity we write each clause on a different line and we use blank lines to separate the clauses that define a given predicate.

Example of a program

```
% the arc clauses
arc(a,b).
arc(b,c).
arc(c,a).
arc(b,d).

% the first path predicate
path(X,X).
path(X,Y) :- arc(X,Z), path(Z,Y).

% the second path predicate
path2(X,X).
path2(X,Y) :- path2(X,Z), path(Z,Y).
```