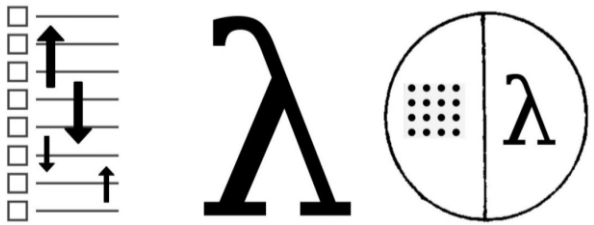
Fundusze Europejskie
Wiedza Edukacja Rozwój

Politechnika Wrocławska

Unia Europejska
Europejski Fundusz Społeczny

„ZPR PWr – Zintegrowany Program Rozwoju Politechniki Wrocławskiej”



Programming paradigms

L15: Logical programming

by Michał Szczepanik

Programming paradigms

Imperative in which the programmer instructs the machine how to change its state:

- procedural which groups instructions into procedures,
- object-oriented which groups instructions together with the part of the state they operate on.

Declarative in which the programmer merely declares properties of the desired result, but not how to compute it

- functional in which the desired result is declared as the value of a series of function applications,
- **logic in which the desired result is declared as the answer to a question about a system of facts and rules,**
- mathematical in which the desired result is declared as the solution of an optimization problem

Logical programming

Logic programming is a programming paradigm which is based on formal logic. Any program written in a logic programming language is a set of sentences in logical form, expressing facts and rules about some problem domain.

Rules are written as logical clauses with a head and a body:

"H is true if B1, B2, and B3 are true."

Facts are expressed similar to rules, but without a body:

"H is true."

Languages

Some logic programming languages, such as Datalog and ASP (Answer Set Programming), are purely declarative.

They allow for statements about what the program should accomplish, with no explicit step-by-step instructions on how to do so.

Others, such as Prolog, are a combination of declarative and imperative. They may also include procedural statements:

"To solve H, solve B1, B2, and B3."

Prolog

Prolog was first conceived by Alain Colmerauer and his group in Marseille, France, in the 1970.

The first Prolog system was developed in 1972 by Alain Colmerauer with Philippe Roussel

This is one of first and also well known logical programming language.

There are many implementations of Prolog:

https://en.wikipedia.org/wiki/Comparison_of_Prolog_implementations

Syntax and Basic Fields

In prolog some facts are declared. These facts constitute the Knowledge Base of the system. programmer can query against the Knowledge Base.

Output is get as affirmative if our query is already in the knowledge Base or it is implied by Knowledge Base, otherwise we get output as negative.

Knowledge Base can be considered similar to database, against which we can query.

Prolog facts are expressed in definite pattern. Facts contain entities and their relation.

Simple Facts

In Prolog we can make some statements by using facts. Facts either consist of a particular item or a relation between items. For example we can represent the fact that it is sunny by writing the program:

sunny.

query:

?- sunny.

?- is the Prolog prompt.

To this query, Prolog will answer **yes**. sunny is true because (from above) Prolog matches it in its database of facts.

Syntax and Basic Fields

Entities are written within the parenthesis separated by comma (,).
Their relation is expressed at the start and outside the parenthesis.
Every fact/rule ends with a dot (.).

Format (Facts with Arguments):

relation(entity1, entity2, ...k'th entity).

Example

```
friends(ann, tom).
singer(garu).
odd_number(5).
```

These facts can be interpreted as:

friends(ann, tom).	->	ann and tom are friends.
singer(garu).	->	garu is a singer.
odd_number(5).	->	5 is an odd number.

Running queries

Query 1 : ?- singer(garu).

Output : Yes.

Explanation : As our knowledge base contains the above fact, so output was 'Yes', otherwise it would have been 'No'.

Query 2 : ?- odd_number(7).

Output : No.

Explanation : As our knowledge base does not contain the above fact, so output was 'No'.

Variables and Unification

How do we say something like "What does cat eat"?

In database there is fact like:

eats(cat,mause).

For standard query:

?- eats(cat,what).

The answer will be **NO**

Variables are distinguished by starting with a capital letter.

?- eats(cat,What).

What = mause

y

Rules

Consider the following

'All men are mortal’:

This can be expressed as the following rule

```
mortal(X) :-
        human(X).
```

Rules

```
mortal(X) :-
        human(X).
human(socrates).

?- mortal(socrates).
```

Answer: Yes

Search and Bactracking

```
eats(tom,pears).
eats(tom,t_bone_steak).
eats(tom,apples).

?- eats(tom,FoodItem).
FoodItem = pears
FoodItem = t_bone_steak
FoodItem = apples
```

Bactracking Rules

```
hold_party(X):-birthday(X),
happy(X).
birthday(tom).birthday(fred).
birthday(helen).
happy(mary).
happy(jane).
happy(helen)

?- hold_party(Who).
X = helen
```

Recursion

As is commonly the case in many programming tasks, we often wish to repeatedly perform some operation either over a whole data-structure, or until a certain point is reached.

This is done in Prolog is by recursion. This simply means a program calls itself typically until some final point is reached.

Frequently in Prolog what this means is that we have a first fact that acts as some stopping condition followed up by some rule(s) that performs some operation before reinvoking itself.

Recursion

```
on_route(rome).
on_route(Place):-
        move(Place,Method,NewPlace),
        on_route(NewPlace).

move(home,taxi,halifax).
move(halifax,train,gatwick).
move(gatwick,plane,rome).
```

List

Prolog also has a special facility to split the first part of the list (called the head) away from the rest of the list (known as the tail). We can place a special symbol | (pronounced 'bar') in the list to distinguish between the first item in the list and the remaining list. For example, consider the following.

[first,second,third] =

[A|B]where *A = first* and *B=[second,third]*

Key Features

- Unification:
  The basic idea is, can the given terms be made to represent the same structure.
- Backtracking:
  When a task fails, prolog traces backwards and tries to satisfy previous task.
- Recursion:
  Recursion is the basis for any search in program.

Pros & Cons

Advantages :

- Easy to build database. Doesn’t need a lot of programming effort.
- Pattern matching is easy. Search is recursion based.
- It has built in list handling. Makes it easier to play with any algorithm involving lists.

Disadvantages :

- LISP (another logic programming language) dominates over prolog with respect to I/O features.
- Sometimes input and output is not easy.

Thank you for your attention.