Constraint Programming

January 2020

ADEO 2

Anissa Lamani

GNU Prolog solver

GNU Prolog solver

- www.gprolog.org
- Prolog is a logic based programming language.
- Declarative language: you specify the problem to be solved rather than how to solve it.
- Extension of a source file: pl
- Execution: consult (filename).

GNU Prolog solver

- Variables
  - Once a value is assigned it cannot be overwritten !
  - Variable names start with an upper-case letter.

Example: X=2, X=X+1, write(X).

X=2, Y is X+1, write(Y).

GNU Prolog solver

- Facts
  - A fact asserts some property of an item, or relation between two or more items.
  - Syntax: always begin with a lower case.

GNU Prolog solver

- Facts

  animal(dog).
  animal(cat).
  animal(horse).
  person(thomas).
  person(lucie).

- Requests

  Is Thomas a person?

  person(thomas).

GNU Prolog solver

- Facts

```
animal(dog).
animal(cat).
animal(horse).
person(thomas).
person(lucie).
```

- Requests

What are all the animals?

```
animal(X).
```

GNU Prolog solver

- Facts

```
animal(dog).
animal(cat).
animal(horse).
person(thomas).
person(lucie).
```

- Requests

Is there an animal?

```
animal(_).
```

GNU Prolog solver

- Facts

```
animal(dog).
animal(cat).
animal(horse).
person(thomas).
person(lucie).
```

- Requests

All the statements?

```
listing.
```

GNU Prolog solver

- Facts
  - A fact asserts some property of an item, or relation between two or more items.

Example:

bigger(elephant, horse).

bigger(horse, donkey).

bigger(donkey, monkey).

bigger(elephant, monkey).

GNU Prolog solver

- Facts
  - A fact asserts some property of an item, or relation between two or more items.
- Rules
  - Allow us to infer that a property or relationship holds based on preconditions.

GNU Prolog solver

- Rules
  - Allow us to infer that a property or relationship holds based on preconditions.

```
is_bigger(X,Y) :- bigger (X,Y).
is_bigger(X,Y) :- bigger(X,Z), bigger(Z,Y).
```

if and

```
is_bigger(elephant, monkey).

is_bigger(X, monkey).
```

GNU Prolog solver

- Exercise
  - Statements
    - Ben and Anna are married.
    - Ben is John’s father.
  - Rule
    - If M is married to F and F is E’s father then M is E’s mother.

Who is John’s mother?

GNU Prolog solver

- Example
  - Statements
    - Ben and Anna are married. married(ben, anna).
    - Ben is John’s father. father(ben, john).
  - Rule
    - If M is married to F and F is E’s father then M is E’s mother.

mother(M, E) :- married(F, M), father(F, E).

GNU Prolog solver

- Facts
  - A fact asserts some property of an object, or relation between two or more objects.
- Rules
  - Allow us to infer that a property or relationship holds based on preconditions.

Both facts and rules are predicates definitions.

GNU Prolog solver

- Lists

  [1, 2, 3, 4]

  Empty list: []

  [X|L]: X is the head of the list and L is the tail of the list.

  [1, 2, 3, 4] = [1|[2, 3, 4]]
  = [1|2|[3, 4]]
  = [1|2|3|[4]]
  = [1|2|3|4|[]]

GNU Prolog solver

- Lists

Check whether a list is in ascending order.

```
asc([]).
asc([A]).
asc([A|[B| T]]):- A<=B, asc([B|T]).
```

GNU Prolog solver

- Lists

  member(X,L): returns true is X is part of the list.

  last(L,X): returns true is X is the last element of the list.

  Reverse (L,X): reverses L and stores the result in X.

GNU Prolog solver

- Constraint programming

  - Finite domains FD (variables that can only take values in their domains).

  - By default: 0 to fd_max _integer (fd_max _integer is the greatest value that an FD variable can take).

  - Fd_domain/3: fd_domain(+list_of_variables_FD, integer, integer)

    Example: fd_domain([X1,X2,X3], 1, 4).

    Equivalent to: fd_domain(X1,1,4), fd_domain(X2,1,4), fd_domain(X3,1,4).

GNU Prolog solver

- Constraint programming

  - Fd_domain/2: fd_domain(+list_of_variables_FD, list of integers).

    Example: fd_domain(X1, [1, 4,10]).

GNU Prolog solver

- Constraint programming

fd_all_different(List_of_variables)

Example: fd_all_different([X,Y,Z])

GNU Prolog solver

- Constraint programming

  Arithmetics constraints

  # =, # \ =, #< , #>, #=<, #> =

  # = #, # \ = #, #< #, #> #, #=< #, #> = #

  Example:

  fd_domain (X, 1, 8),fd_domain (Y, 2, 7),X# = 2 ∗ Y .

  fd_domain (X, 1, 8),fd_domain (Y, 2, 7),X# = #2 ∗ Y .

GNU Prolog solver

- Constraint programming

fd_vector _max/ 1.

X# = < 512
Domain: 0..512.
X # \ = 10,
New domain 0..9 : 11..127.

GNU Prolog solver

- Constraint programming

  Boolean operators

  # < = >, # \< = >, #= >, #\ , # \/, # /\

  Example:

  (X#=Y) #<=> (Y#=<3)

GNU Prolog solver

- CSP resolution by GNU Prolog

Fd_labeling(list_of_fd_Variable).

GNU Prolog solver

- Constraint programming

fd_minimize: returns the minimum value allowed for a variable.

Example: X+Y #= 10, Y#<3 , fd_minimize(fd_labeling([X,Y]), X).

Each time fd_labeling([X,Y]) gives a solution X=n, the search is started again with a new constraint X#<n.

fd_maximize

GNU Prolog solver

- Constraint programming

fd_minimize: returns the minimum value allowed for a variable.