Constraint Programming: ECL^{i}PS^e

Eric MONFROY

IRIN, Université de Nantes

Objectives

- first steps in ECL^{i}PS^e
- generic and basic framework for constraint programming
- some constraint predicates and solvers
- illustration with examples

Generic framework

A very useful command of ECL^{i}PS^e

help

- `help text.` : list all matching predicates
- `help(text).` : list all matching predicates
- `help(predicate/arity)` : help on `predicate` of arity `ar`
- `help(module:predicate/arity)` : help on `predicate` of arity `ar` from the module/library `module`

Basic framework

a constraint program :

1. variable and domain declaration
2. setting constraints
3. search

in ECL^{i}PS^e :

```
1 :-lib(fd).                    % load fd constraints/solver
2 problem(Vars):-
3   domainVars(Vars),           % associate domains to variables
4   setConstraints(Vars),       % setting constraints
5   search(Vars).               % often labeling
```

Domain declaration

finite domains :

1. as sets : `X :: [1,3,6]`
2. as interval (but used as sets) : `X :: [1..300]`
3. as sets of atoms : `X :: [blue, red, green]`
4. for a list of variables : `[X,Y,Z] :: [0,1]`

Domain declaration : example

```
1 send(L):-
2     L=[S,E,N,D,M,O,R,Y],
3    [E,N,D,O,R,Y] :: [0..9],
4    [S,M] :: [1..9],

1 [eclipse 17]: send1(L).
2
3 L = [S{[1..9]}, E{[0..9]}, N{[0..9]}, D{[0..9]},
4    M{[1..9]}, O{[0..9]}, R{[0..9]}, Y{[0..9]}]
5 Yes (0.00s cpu)
```

FD constraints

finite domain constraints :

1. equality constraints : `?X #= ?Y`
2. disequality constraint : `?X #\= ?Y` or `?X ## ?Y`
3. greater than constraint : `?X #> ?Y`
4. less than constraint : `?X #< ?Y`
5. greater than or equal constraint : `?X #>= ?Y`
6. less than or equal constraint : `?X #<= ?Y`
7. all different constraint : `alldiff(ListVars)`

FD constraints : example (ctd)

program :

```
1  send(L):-
2    L=[S,E,N,D,M,O,R,Y],
3
4    % domain declarations
5    [E,N,D,O,R,Y] :: [0..9],
6    [S,M] :: [1..9],
7
8    % constraints
9     1000*S + 100*E + 10*N + D
10   +1000*M + 100*O + 10*R + E
11   #= 10000*M + 1000*O + 100*N + 10*E + Y,
12   alldifferent(L).
```

FD constraints : example (ctd)

answer :

```
 1 [eclipse 25]: send2(L).
 2 L = [9, E{[4..7]}, N{[5..8]}, D{[2..8]}, 1, 0, R{[2..8]}, Y{[2..8]}]
 3
 4 There are 11 delayed goals. Do you want to see them? (y/n)
 5 Delayed goals:
 6         E{[4..7]} #~ N{[5..8]}
 7         E{[4..7]} #~ D{[2..8]}
 8         E{[4..7]} #~ R{[2..8]}
 9         E{[4..7]} #~ Y{[2..8]}
10         N{[5..8]} #~ D{[2..8]}
11         N{[5..8]} #~ R{[2..8]}
12         N{[5..8]} #~ Y{[2..8]}
13         D{[2..8]} #~ R{[2..8]}
14         D{[2..8]} #~ Y{[2..8]}
15         0 - Y{[2..8]} + 91 * E{[4..7]} - 90 * N{[5..8]}
16           + 10 * R{[2..8]} + D{[2..8]}#=0
17         R{[2..8]} #~ Y{[2..8]}
18 Yes (0.00s cpu)
```

Search : labeling

- labeling = enumeration
- labeling : instantiates all variables in a list to values in their domain (get next value by backtracking)
- defined as :

```
1        labeling([]).
2        labeling([Var|Rest]) :-
3            indomain(Var),
4            labeling(Rest).
```

Search : labeling (2)

labeling can be improved

strategies to select the variable :

- deleteff(Var,List,Rest) : to select the variable with smallest domain
- deleteffc(Var,List,Rest) : to select the variable with smallest domain and most constrained

strategies to select the value :

- mindom : to select the minimum value
- maxdom : to select the maximum value

Labeling : example (ctd)

program :

```
 1 send(L):-
 2   L=[S,E,N,D,M,O,R,Y],
 3
 4   % domain declarations
 5   [E,N,D,O,R,Y] :: [0..9],
 6   [S,M] :: [1..9],
 7
 8   % constraints
 9    1000*S + 100*E + 10*N + D
10   +1000*M + 100*O + 10*R + E
11   #= 10000*M + 1000*O + 100*N + 10*E + Y,
12   alldifferent(L),
13
14   % enumeration
15   labeling(L).
```

Labeling : example (ctd)

answer :

```
1 [eclipse 44]: send3(L).
2
3 L = [9, 5, 6, 7, 1, 0, 8, 2]
4 More (0.00s cpu) ? ;
5
6 No (0.00s cpu)
```