Constraint Programming: Environment and Programming Methods

Université de Nantes

Objectives

- some debugging tools
- some profiling tools
- code optimization
- some programming methods

Debugging

Debugging (1)

constraint programming = two levels

- host language
  ➔ usual debugging tools
- level of the constraint store and solver
  ➔ debugger ?

differents debuggings :

- *debugging for correctness* : some solutions are missing ; answers are not the expected ones
- *debugging for efficiency* : the computation time is too large

Debugging (2)

Problems encountered :

- renamed variables ($X \longrightarrow \_123$)
- constraints decoupled in *primitives* constraints

available tools :

- graphic trace of propagation in the store
- graphic visualization of the search tree

debugging tools in CP : still at the level of research
➔ use of the Prolog trace. . .

Profiling

- efficiency debugging
- few tools

GNU Prolog : no specific tool

ECL^{i}PS^e :

- profiling
- trace (with jump, leap, . . .)
- special debugger for CHR (to spy matching, firing, delaying of rules)

Programming environment

Programming methodology

Solving a problem in CP

1. drawing up of a *model* (*mathematical* representation of the problem)
2. Programming of the model
3. Refinement of the model and of the program : increase efficiency

in the following, we only consider the program

Evaluating efficiency

efficiency criteria : size of the search tree

disadvantage : does not take into account the costs of operations

where to act :

- ordering of clauses defining a predicate
- ordering of goals (ordering for introducing constraints in the *store*)
- enumeration strategies : selection of the variable to instanciate, choice of the value to use
- choice of constraints (primitive constraints or global constraints)
- choice of the local consistency which is enforced
- adding redundant constraints

Where to act (1)

- ordering of clauses
  - ordering by decreasing success probability
- ordering of goals
  - constraints likely to lead to a failure first ; other constraints at the end
  - goals that lead to the most branches at the end

Where to act (2)

- enumeration strategies
  - *first-fail* : enumeration with the smallest domain variable
  - selection of value : heuristics depending on the problem
- choice of constraints
  - factorizing arithmetic expressions (less auxiliary variables are added when decoupling in primitive constraints)
- choice of consistency
  - constraints creating *holes* : (hyper-)arc consistancy
  - arithmetic constraints : bound consistency

Redundant constraints (1)

a *redundant constraint* is a constraint that can be removed without changing the solutions
interest : redundant constraints selected to prune the search tree earlier

- **redundant constraint w.r.t. solutions** : constraint satisfied by all the solutions of the program, that leads quicker to a failure, or earlier to a solution
- **redundant constraint w.r.t. the *store*** : constraint implied by the constraint *store* (explicitly added to alleviate solver incompleteness)

Redundant constraints (2)

Example :

```
1 % computation of ∑_{i=1}^{n} i
2 sum(0,0).
3 sum(N,S+N):-
4     N #>= 1,   S #>= 0, sum(N-1,S).
```