

Programming Languages and Programming Methodologies

Constraint Logic Programming

2008 - 2009
Prof. Gerda Janssens

PLPM 2008-2009

1

CLP

PLPM 2008-2009

2

A CLP system is typically defined by

- A domain of computation (reals, integers, booleans,)
- The type of allowed constraints
e.g., arithmetic constraints
 $3 * x + 7 * y = 4, x < 7$
- Constraint solving algorithm: Gaussian elimination, simplex algorithm
- Note: what about normal Prolog?
(Prolog terms with unification)

PLPM 2008-2009

3

Advantages of CLP approach

- Programs are more flexible and more expressive
- May save coding.
- Solvers are typically more efficient
- Also search space reduction
(thus again better efficiency)
- BUT the complexity of the solvers can affect the performance

PLPM 2008-2009

4

Basic CLP approach

- Prolog uses **generate + test**
- With refinements for using a coroutine-style or interleaving producer-consumer predicates
- CLP does first put the **constraints** and then the **generate**

PLPM 2008-2009

5

Example Search Space Reduction: complete space for Prolog!!!

```
solution(X,Y,Z) :- p(X), p(Y), p(Z), test(X,Y,Z).
```

```
p(14). p(15). p(16). p(7). p(3). p(11).  
test(X,Y,Z) :- Y is X + 1, Z is Y + 1 .
```

```
?- solution(X,Y,Z).
```

```
?- p(X), p(Y), p(Z), test(X,Y,Z).
```

```
% for X: 6 possible values, also for Y and Z
```

```
% thus 63 possibilities
```

```
% only 1 success branch in the search space
```

```
X = 14, Y = 15 and Z = 16!!!
```

PLPM 2008-2009

6

Search Space Reduction with CLP

```
solution(X,Y,Z) :- test(X,Y,Z), p(X), p(Y), p(Z).  
p(14). p(15). p(16). p(7). p(3). p(11).  
test(X,Y,Z) :- { Y = X + 1, Z = Y + 1 } .  
?- solution(X,Y,Z).  
?- test(X,Y,Z), p(X), p(Y), p(Z).  
% execution of test/3 puts 2 constraints in the  
% constraint store (CS): { Y = X + 1, Z = Y + 1 }  
?- p(X),p(Y),p(Z).  
% using p(14) adds the x = 14 constraint to the CS  
% { X = 14, Y = X + 1, Z = Y + 1 } or {X=14,Y=15,Z=16}  
?- p(15),p(16). ...  
... success  
% on backtracking 5 more values for x with also smaller  
% search spaces
```

PLPM 2008-2009

7

CLP(R)

- Is a language based on Prolog (still unification and backtracking)
- Depth-first left-to-right execution
- Able to solve sets of
 - Linear equations (=)
 - Inequations (<, =<, >, >=)
 - Disequations (=≠)
- Non-linear equations are delayed (waiting to become linear!)

PLPM 2008-2009

8

CLP(R) w.r.t Prolog

- is/2 is subsumed by =/2
- Constrain + generate

Some small exmples

- | | |
|-------------------------------|--------------------------------|
| 1. ?- {A + B = 10 }. | 1. B = 10.0 - A |
| 2. ?- {A + B = 10}, A=B. | 2. A = 5.0, B = 5.0 |
| 3. ?- {A + B = 10, A=B}. | 3. A = 5.0, B = 5.0 |
| 4. ?- {A + B = 10}, A = 5. | 4. TYPE ERROR, real nb expect. |
| 5. ?- {A + B = 10, A =\=B}. | 5. B = 10.0 -A, A =\= 5.0 |
| 6. ?- {A + B =< 10, A=B}. | 6. B = A, A =< 5.0 |
| 7. ?- {A + B =< 10, A =\= B}. | 7. A + B =< 10, A -B =\= 0.0 |
| 8. ?- {X = Y * Z}. | 8. X = Y * Z = 0.0 |
| 9. ?- {X = Y *Z, Y = 3} | 9. Y = 3.0, Z = 0.33...3 *X |

Relation with linear programming

- Maximize profit producing goods
- A1 amount of good1, A2 amount of good2
- Profit for good1: 10, for good2 : 15
- Maximize $10 \cdot A1 + 15 \cdot A2$
- $A1 + 2 \cdot A2 \leq 1500$ resource1
- $A1 + A2 \leq 1200$ resource2
- $A2 \leq 500$ resource 3

Using CLP

```
?- { Profit = 10*A1 + 15 * A2,  
    A1 + 2*A2 =< 1500,      % resource1  
    A1+A2 =< 1200,        % resource2  
    A2 =< 500,  
    A1 >= 0, A2 >= 0  
    },  
    maximize(Profit).
```

```
Profit = 13500.0  
A1 = 900.0  
A2 = 300.0
```

Practical things

- Use SICStus Prolog

Put in the beginning of the file the directive:

```
:- use_module(library(clpr)).
```

- You can use SICStus Prolog in the Pclabo's of the university
- Also these of Department Computerscience <http://www.cs.kuleuven.be/computerlab>

exercises

- Write a predicate that computes factorial
- Write the predicate sumlist.

CLP(FD)

- Finite domain solver
- Associates with each variable a finite subset of Z (the set of integers)
- E.g. variable D has domain $\{-123, -10..4, 10\}$
 - $?- D \text{ in } \{-123\} \vee (-10..4) \vee \{10\}$
- Arithmetic operations on the variables (+, -, *, /)
- Establish linear relationships among arithmetic expressions ($\# =$, $\# <$, $\# = <$)
- To narrow the domains of the variables

Practicalities

```
%SICStus
:- use_module(library(clpfd)).

% swi
:- use_module(library('clp(bounds)')).
:- use_module(library(clpfd)). %more recent versions
% less advanced!!!
```

```
?- D in {-123} \vee (-10..4) \vee {10}. % sicstus
?- D in -123 \vee (-10..4) \vee 10. %swi
?- X #= A + B, A in 1..3, B in 3..7.
```

Examples(1)

```
?- X #= A + B, A in 1..3, B in 3..7.  
A+B #= X, A in 1..3, B in 3..7, X in 4..10  
% min/max values are added...
```

```
?- X #= A - B, A in 1..3, B in 3..7.  
..., X in -6..0  
% -6 = min of A - max of B  
% 0 = max of A - min of B
```

```
?- X #= A - B, A in 1..3, B in 3..7, X #>= 0 .  
% more constraints : narrow domains  
X =0, A =3 , B =3
```

PLPM 2008-2009

17

Examples(2)

```
?- X *X + Y * Y #= Z * Z , L = [X,Y,Z],  
domain(L,1,1000).
```

```
X in 1..1000 , Y in 1..1000, Z in 1..1000
```

```
?- X *X + Y * Y #= Z * Z , L = [X,Y,Z],  
domain(L,1,1000), labeling([], L).  
% swi label(L).
```

```
X=4, Y=3, Z=5;  
X=8, Y=6, Z=10;  
X=12, Y=5, Z=13;
```

PLPM 2008-2009

18

Examples(3)

```
?- domain([X,Y,Z],1,4), X #< Y, Y #< Z.  
X in 1..2, Y in 2..3, Z in 3..4
```

```
?- domain([X,Y,Z],1,4), X #< Y, Y #< Z,  
labeling([], [X,Y,Z]).
```

```
1 2 3 ;  
1 2 4 ;  
1 3 4 ;  
2 3 4 ; ...
```

PLPM 2008-2009

19

Interval/bounds consistency

- Constraint solver maintains INTERVAL CONSISTENCY
- The finite domain is represented as an INTERVAL: [Lowerbound .. Upperbound]
- A constraint c is interval consistent if for every variable in the constraint c each of its 2 domain bounds participates in a solution to c .

PLPM 2008-2009

20

Interval consistency(2)

- A constraint c is interval consistent if for every variable in the constraint c each of its 2 domain bounds participates in a solution to c .

- Examples

```
?- domain([X,Y,Z],1,4), X #< Y, Y #< Z.  
X in 1..2, Y in 2..3, Z in 3..4  
?- X in 3..10, Y in 0..4, 3*X - 5 *Y #= 4.  
X in 3..8, Y in 1..4  
?- X in 3..10, Y in 0..4, 3*X - 5 *Y #= 4,  
  labeling([], [X,Y]).  
X = 3, Y = 1 ; X = 8, Y = 4
```

PLPM 2008-2009

21

Interval consistency(3)

```
?- domain([X,Y,Z],1,2), X #\=Y, Y #\= Z, X #\= Z.  
X in 1..2, Y in 1..2, Z in 1..2
```

```
?- domain([X,Y,Z],1,2), all_different([X,Y,Z]).  
% bounds/interval consistent for EACH constraint  
  (separately!!!)
```

```
?- L = [X,Y,Z], domain(L,1,2), all_different(L),  
  labeling([],L).
```

No!!!

PLPM 2008-2009

22

Global constraint: all_distinct(L)

- Checks domain consistency for all the variables in L

```
?- domain([X,Y,Z],1,2), all_distinct([X,Y,Z]).  
No  
?- X + Y #= Z, X =1, Z = 6, Y in 1..10, Y #\= 5.  
No  
% for #\= : also domain consistency is maintained if  
  both arguments are variables or constants
```

PLPM 2008-2009

23

Shoes problem

- Harriet, upon returning from the mall, is happily describing her four shoes purchases to her friend Aurora. Aurora just loves the four different kinds of shoes that Harriet bought (ecru espadrilles, fuchsia flats, purple pumps, and suede sandals), but Harriet can't recall at which different store (Foot Farm, Heels in a Handcart, The Shoe Palace, or Tootsies) she got each pair. Can you help these two figure out the order in which Harriet bought each pair of shoes, and where she bought each?

1. Harriet bought fuchsia flats at Heels in a Handcart.
2. The store she visited just after buying her purple pumps was not Tootsies.
3. The Foot Farm was Harriet's second stop.
4. Two stops after leaving The Shoe Palace, Harriet bought her suede sandals.

PLPM 2008-2009

24

Expressing a disjunction in CLP

```
nextto(X,Y) :- X #= Y + 1.  
nextto(X,Y) :- Y #= X + 1.
```

```
%disjoint tasks in CLP(FD): S starttime, D duration  
disjoint(S1,D1,S2,D2) :- ...  
disjoint(S1,D1,S2,D2) :- ...
```

Natural way to implement a disjunction: 2
Prolog clauses
Interaction with CLP approach

Expressing a disjunction in CLP(2)

Interaction with CLP approach :
declaration of variables and their domains
constrain the variables
generate by labeling (aka search)
Search?
labeling step
possible interaction with the constraints!!

Prolog disjunction

```
nextto(X,Y) :- X #= Y + 1.  
nextto(X,Y) :- Y #= X + 1.
```

```
?- nextto(No,B1), ... nextto(H,D),...
```

In execution trace

```
?- nextto(No,B1), ... nextto(H,D).  
   c11 |           | c12  
?- nextto(H,D)           ?- nextto(H,D)  
   c11 |   | c12   c11 |   \ c12  
   label1 label2   label3 label4 !!!!  
ok?????
```

REIFICATION of constraints

- Avoid backtracking
- Give a name to a constraint and talk about it by using this name
- Note **reification** comes from latin: **res + facere** meaning thing + make
- The satisfaction of a constraint is associated with a boolean variable (value 0 or 1).

Reification (2)

- $X \#= 1$ or $X \#= 2$
- B1 corresponds to (is the name for) " $X \#= 1$ "
B2 corresponds to " $X \#= 2$ "
- Now we can express that
either B1 or B2 has to be true
namely by $B1 + B2 \#>= 1$
- In SICStus we can say: $B1 \#<=> X \#= 1$.
- Thus $nextto(X,Y)$ becomes??

Nextto/2

```
nextto(X,Y) :-  
    X \#= Y + 1 \#<=> B1,  
    Y \#= X + 1 \#<=> B2,  
    B1 + B2 \#= 1 .
```

Non overlapping tasks: do it!!! (S1,D1) (S2,D2)

AKA propositional constraints: do no longer make the
booleans explicit!!! (see manual of SICStus)

```
C1 #/\ C2 % C1 \#<=> B1, C2 \#<=> B2, B1 + B2 \#= 2  
C1 #\| C2 % C1 \#<=> B1, C2 \#<=> B2, B1 + B2 \#>= 1
```

Example : exactly!!!