

CONSTRAINT LOGIC PROGRAMMING- I



CPT S 355

FIBONACCI IN PROLOG



- **fib(N, F) :-**
- **N=0, F=1 ;**
- **N=1, F=1 ;**
- **N>1,**
- **N1 is N-1, fib(N1,F1),**
- **N2 is N-2, fib(N2,F2),**
- **F is F1 + F2.**

FIBONACCI IN PROLOG



- Intended use:
 - **?- fib(6,F).**
 - **F=13**

- A question in the opposite direction:
 - **?- fib(N, 13).**
 - **Error**

- Goal $N > 1$ is executed with N uninstantiated

FIBONACCI IN CLP(R)



- **fib(N, F) :-**
- **{ N = 0, F = 1 } ;**
- **{ N = 1, F = 1 } ;**
- **{ N > 1, F = F1 + F2, N1 = N - 1, N2 = N - 2}**
- **,**
- **fib(N1, F1),**
- **fib(N2, F2).**

FIBONACCI IN CLP(R)



- This can be executed in the opposite direction:
 - **?- fib(N, 13).**
 - **N = 6**

- However, still gets into trouble when asked an unsatisfiable question:
 -
 - **?- fib(N, 4).**

FIBONACCI IN CLP(R)



- ?- fib(N, 4).
- The program keeps trying to find two Fibonacci numbers F_1 and F_2 such that $F_1 + F_2 = 4$. It keeps generating larger and larger solutions for F_1 and F_2 , all the time hoping that eventually their sum will be equal 4. It does not realise that once their sum has exceeded 4, it will only be increasing and so can never become equal 4. Finally this hopeless search ends in a stack overflow.

FIBONACCI: EXTRA CONSTRAINTS



- Fix this problem by adding constraints
- Easy to see: for all N : $F(N) \geq N$
- Therefore variables N_1 , F_1 , N_2 and F_2 must always satisfy the constraints:
 - $F_1 \geq N_1$, $F_2 \geq N_2$.

FIBONACCI: EXTRA CONSTRAINTS



- **fib(N, F) :-**
- **.....**
- **;**
- **{ N > 1, F = F1+F2, N1 = N-1, N2 = N-2,**
- **F1 >= N1, F2 >= N2}, % Extra constraints**
- **fib(N1, F1),**
- **fib(N2, F2).**

FIBONACCI: EXTRA CONSTRAINTS



- ?- **fib**(N, 4).
- **no**
- The recursive calls of **fib** expand the expression for F in the condition **F = 4**:
 - **4 = F = F1 + F2 =**
 - **F1' + F2' + F2 =**
 - **F1'' + F2'' + F2' + F2**

FIBONACCI: EXTRA CONSTRAINTS



- The recursive calls of **fib** expand the expression for F in the condition $F = 4$:
 - $4 = F = F_1 + F_2 =$
 - $F_1' + F_2' + F_2 =$
 - $F_1'' + F_2'' + F_2' + F_2$
- Additional constraints that make the above unsatisfiable:
 - $F_1' \geq N_1' > 1, F_2'' \geq N_2'' > 1,$
 - $F_2' \geq N_2' > 1, F_2 \geq N_2 > 1$

FIBONACCI: EXTRA CONSTRAINTS



- Each time this expression is expanded, new constraints are added to the previous constraints. At the time that the four-term sum expression is obtained, the constraint solver finds out that the accumulated constraints are a contradiction that can never be satisfied.

CONSTRAINT LOGIC PROGRAMMING



- Pure Prolog: limited constraint satisfaction language; all constraints are just equalities between terms
- To extend Prolog to a "real" CLP language: add other types of constraints in addition to matching
- Constraint Logic Programming (CLP) = Constraint programming + LP

Constraint Programming



Constraint Programming and Languages is driven by a number of goals

- Declarative Nature
 - Ideally, programs should specify the constraints to be solved, not the algorithms used to solve them
- Expressivity
 - Constraint Languages should be able to easily specify the variables, domains and constraints (e.g. conditional, global, etc...);
- Efficiency
 - Solutions should be found as efficiently as possible, i.e. With the minimum possible use of resources (time and space).

These goals are partially conflicting goals and have led to the various developments in this research and development area.

APPLICATIONS OF CLP



- Scheduling
- Logistics
- Resource management in production, transportation, placement
- Simulation

Typical applications involve assigning resources to activities

- machines to jobs,
- people to rosters,
- crew to trains or planes,
- doctors and nurses to duties and wards.

Introduction to OZ



- Can be used for constraint-based problem solving.
- two basic techniques of constraint programming:
 - constraint propagation
 - constraint distribution
- Current Version: 3.0 (Available in Mozart System).
[Older Version:2.0]

Tools for Constraint Programming

16

- Constraint Logic Programming
 - CLP(R)
 - Oz
 - CHIP
 - ECLIPSE
 - GNU Prolog (CLP-FD)
 - SICStus
- Constraint Programming
 - CHIP++
 - ILOG
 - GE Code

Download & Installation



- Next Class:

<http://www.mozart-oz.org/documentation/fdt/>

- Download:

<http://www.mozart-oz.org/download/view.cgi>

Screenshot



```
Oz Programming Interface (emacs@EME206-01)
Buffers Files Tools Edit Search Mule Compile Help
Root = sol(s:S e:E n:N d:D m:M o:O r:R y:Y)      # 1
Root ::: 0#9                                     # 2
{FD.distinct Root}                               # 3
S \=: 0                                           # 4
M \=: 0                                           # 5
          1000*S + 100*E + 10*N + D
+         1000*M + 100*O + 10*R + E
=: 10000*M + 1000*O + 100*N + 10*E + Y
{FD.distribute ff Root}
end

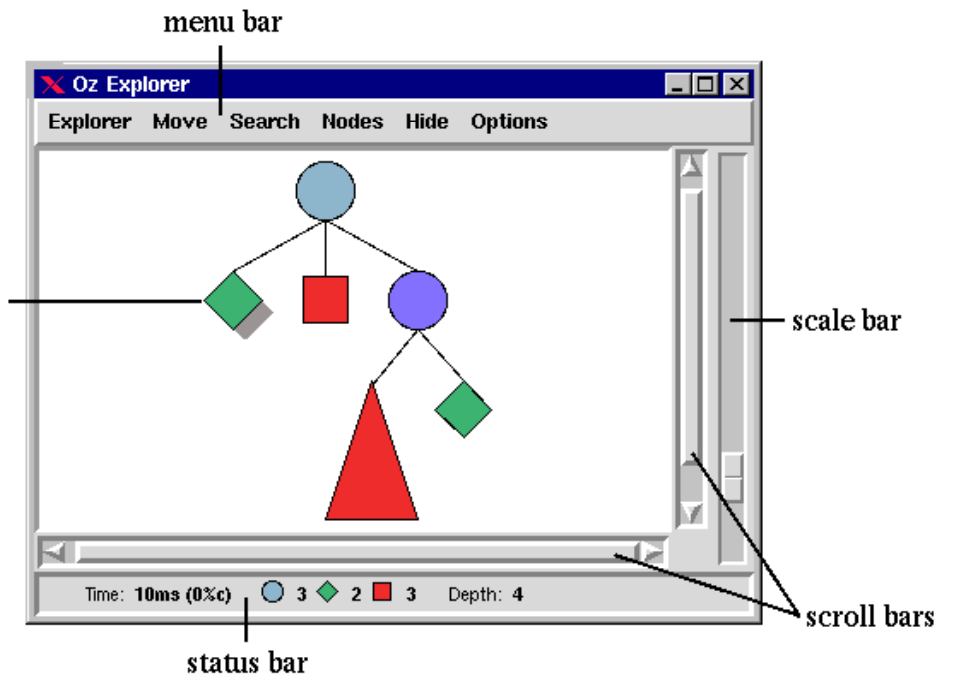
{Browse {SearchAll Money}}
```

```
%%
--\** Oz.oz (Oz)--L24--22%-----
cd c:/
ozc -c "Oz.oz"
Mozart Compiler 1.3.2 (20060615) playing Oz 3

%%% feeding file Oz.oz

***** binding analysis error *****
***
*** variable FD not introduced
***
--\** *compilation* (Compilation:exit [1])--L10--Top-----
Mozart Compiler 1.3.2 (20060615) playing Oz 3

--\** *Oz Compiler* (Compilation:open)--L3--All-----
```



REFERENCES



- “Constraint Logic Programming: A Survey”, by Joxan Jaffar and Michael J. Maher, *Journal of Logic Programming* 1994: 19, 20: 503-581
- “Constraint Logic Programming - An Informal Introduction”, by Thom Frühwirth, Alexander Herold, Volker Küchenhoff, Thierry le Provost, Pierre Lim, Eric Monfroy, Mark Wallace, technical report ECRC-93-5
- “Constraint Logic Programming”, by Dick Fountain, *Byte Magazine* February 1995, © Mc Graw Hill
- “Algorithms for Constraint-Satisfaction Problems: A Survey”, by Vipin Kumar, *AI magazine*, Vol.13, N°1, Spring 1992: 32-44
- Journées Francophones de Programmation en Logique, LaBRI, Université Bordeaux I, 25-27 mai 1994