Constraint Programming: Theory and Practice

Topics:

Introduction and Motivation

Prof. Dr. Slim Abdennadher

Constraint Programming: Much Quoted Sentence

> **Constraint Programming** represents one of the closest approaches computer science has yet made to the Holy Grail of programming: **the user states the problem, the computer solves it**.

Eugene C. Freuder, Inaugural issue of the *Constraints Journal*, 1997.

Constraint programming has been identified by ACM as one of the strategic directions for computing research.

Is this possible?

- Let us try to solve the following Problem: Given a natural number N, the sum of the squares of 4 or fewer numbers is equal to N.
- Mathematical statement of the problem:

$$\forall N \exists S_1, S_2, S_3, S_4 \; N = S_1^2 + S_2^2 + S_3^2 + S_4^2$$

- Encoding the problem in a Constraint Language:

```
:- use_module(library(clpfd)).
solve(N,Sol) :-
      L = [S1,S2,S3,S4],
      N #= S1*S1 + S2*S2 + S3*S3 + S4*S4,
      labeling([],L).
```

- It works perfectly!

Constraint Programming in a Nutshell

Given a **description** of the problem in natural language, Constraint Programmers (CP)

- first **model** the problem as a **constraint problem** including **variables**, their **domains** and **constraints**
- then **resolve** this by means of **constraint solvers**

What are Constraints?

- **Variable:** A place holder for values

$$X, Y, Z, L$$

- **Function Symbol:** Mapping values to values

$$+, -, \times$$

- **Relation Symbol:** Relation between values

$$=, \neq, \leq$$

- **Primitive Constraint:** Constraint relation with arguments

$$X \leq 8$$

$$2X + Y = 10$$

What are Constraints?

- **Constraint:** a conjunction of primitive constraints

$$X \leq 5 \wedge X = Y \wedge 2X + Y = 10$$

Constraint Satisfaction

- Given a constraint C, two questions:
  - **Satisfaction:** Does it have a solution?
  - **Solution:** Give me a solution if it has one?
- **Constraint Solver:** answers the satisfaction problem

Historical Remarks

- **60s:** Early use of constraints in graphics software, e.g. SKETCHPAD.
- **60s, 70s:** Constraint networks in artificial intelligence (CSP).
- **70s:** Logic programming (Prolog).
- **80s:** Constraint logic programming (CLP).
  - First steps in CLP by Jaffar and Lassez ca. 1987
  - Finite Domain CSP studied by Mackworth ca. 1992
  - First Practical implementations by Colmerauer et. al (Prolog III) c.a. 1990
  - CHIP commercialized by COSYTEC 1990
  - ILOG SOLVER introduced 1994
  - First global constraint (all-different) c.a. 1994

⋮

Constraint Logic Programming

- **Constraint Satisfaction Problems (CSP)**: artificial intelligence (1970s)
- **Constraint Logic Programming (CLP)**: developed in the mid-1980's
  - two declarative paradigms: constraint solving and logic programming
  - more expressive, flexible, and in general more efficient than logic programs
  - **Example 1**: $X - Y = 3 \ \wedge \ X + Y = 7$ leads to $X = 5 \ \wedge \ Y = 2$
  - **Example 2**: $X < Y \ \wedge \ Y < X$ fails without the need to know values

In short

LP = Logic + Control

CP = Model + Reasoner

Constraint Satisfaction

- How do we answer the question?
  - **Generate-and-Test in LP**: impractical, facts used in passive manner only
  - **Constrain-and-generate in CLP**: use facts in active manner to reduce the search space (constraints)

The Idea

- **Example – Combination Lock:**
  0 1 2 3 4 5 6 7 8 9
  Greater or equal 5.
  Prime number.
- **Declarative problem** representation by variables and constraints:
  $x \in \{0, 1, \ldots, 9\} \ \wedge \ x \geq 5 \ \wedge \ \text{prime}(x)$
- **Constraint propagation and simplification** reduce search space:
  $x \in \{0, 1, \ldots, 9\} \ \wedge \ x \geq 5 \ \rightarrow \ x \in \{5, 6, 7, 8, 9\}$
  $x \in \{5, 6, 7, 8, 9\} \ \wedge \ \text{prime}(x) \ \rightarrow \ x \in \{5, 7\}$

Constraint Programming

- **Declarative modeling by constraints:**
  - What are Constraints?
  - **Formally**: A relation between objects (problem variables)
  - **Informally**: A statement on how the value of one or more variables restrict the possible values of others.
- **Automatic constraint reasoning:**
  Propagation of the effects of new information.
  Simplification makes implicit information explicit.
- **Solving combinatorial problems efficiently:**
  Easy Combination of constraint solving with search and optimization.

Programming with Constraints versus Imperative Paradigm

Imperative Paradigm

- A program is viewed as operating on a store which holds the **(concrete) values** of the variables (e.g. `X=3`).
- The program progresses and achieves the results **by changing the values** of variables in the store.

Constraint Programming

- The store holds the **possible values** of the variables, i.e. the variables could be **partially** bound (eg. `X < 4`).
- The program does not change the value of a variable, instead it **reduces the possible values** of it.

Crypto-Arithmetic Puzzles

$$\begin{array}{cccccc} & S & E & N & D \\ + & M & O & R & E \\ \hline M & O & N & E & Y \end{array}$$

- Replace each letter by a different digit
- So that the equation above is correct
- The numbers should not start with a 0.

Crypto-Arithmetic Puzzles: Java Program

```
class sendmoremoney
{
 public static void main(String args[])
 {
  int num = 99999999;
  int s,e,n,d,m,o,r,y, count = 0;
  while(num > 0)
  {
   int num2 = num;
   --num;

   y = num2 % 10; num2 /= 10;
   r = num2 % 10; num2 /= 10;
   o = num2 % 10; num2 /= 10;
   m = num2 % 10; num2 /= 10;
   d = num2 % 10; num2 /= 10;
   n = num2 % 10; num2 /= 10;
   e = num2 % 10; num2 /= 10;
   s = num2 % 10; num2 /= 10;
```

Crypto-Arithmetic Puzzles: Java Program

```
    if(s == e || s == n || s == d || s == m || s == o || s == r ||
       s == y || e == n || e == d || e == m || e == o || e == r ||
       e == y || n == d || n == m || n == o || n == r || n == y ||
       d == m || d == o || d == r || d == y || m == o || m == r ||
       m == y || o == r || o == y || r == y || s == 0 || m == 0)
     continue;
    int send = s * 1000 + e * 100 + n * 10 + d;
    int more = m * 1000 + o * 100 + r * 10 + e;
    int money = m * 10000 + o * 1000 + n * 100 + e * 10 + y;

    if(send + more == money)
     System.out.println(" A solution "
          + send + " " + more + " " + money);
   }
  }
 }
```

Crypto-Arithmetic Puzzles: Constraint Program

A CLP Program Modeling the Problem

```
solve(Vars):-
        Vars=[S,E,N,D,M,O,R,Y], % variable generation
        Vars :: 0..9,
        alldifferent(Vars),     % constraint generation
        S #\= 0,
        M #\= 0,
                  1000*S+100*E+10*N+D
                + 1000*M+100*O+10*R+E
        #= 10000*M+1000*O+100*N+10*E+Y.
```

Solving the Puzzle

```
?- solve([S,E,N,D,M,O,R,Y]).

M = 1, O = 0, S = 9, E in 4..7,
N in 5..8, D in 2..8, R in 2..8, Y in 2..8 ?

?- solve([S,E,N,D,M,O,R,Y]), labeling([],[S,E,N,D,M,O,R,Y]).

D = 7, E = 5, M = 1, N = 6,
O = 0, R = 8, S = 9, Y = 2 ?
```

Constraint Propagation using Interval reasoning

Constraint propagation is an **inference rule** for finite domain problems that reduces the domains of variables:

- Given the following constraints:

$$\begin{aligned} X &\in \{0,\ldots,9\} \\ Y &\in \{0,\ldots,9\} \\ X+Y &= 9 \\ 2X+4Y &= 24 \end{aligned}$$

- We can conclude:

  Eq.2 $\Rightarrow X \in \{0,\ldots,9\}, Y \in \{2,\ldots,6\}$
  Eq.1 $\Rightarrow X \in \{3,\ldots,7\}, Y \in \{2,\ldots,6\}$
  Eq.2 $\Rightarrow X \in \{4,\ldots,6\}, Y \in \{3,4\}$
  Eq.1 $\Rightarrow X \in \{5,6\}, Y \in \{3,4\}$
  Eq.2 $\Rightarrow X \in \{6\}, Y \in \{3\}$

SEND + MORE = MONEY (Propagation)

- The equation below initiates **propagation**

  1000*S + 100*E + 10*N + D + 1000*M + 100*O + 10*R + E = 10000*M + 1000*O + 100*N + 10*E + Y

- Transform the equation into a simpler (equivalent) one:

  1000*S + 91*E + D + 10*R = 9000*M + 900*O + 90*N + Y

- M should be less than or equal to
  $\frac{1}{9} * max(S) + \frac{91}{9000} * max(E) + \frac{1}{9000} * \max(D) + \frac{1}{900} * max(R) - \frac{1}{10} * min(O) - \frac{1}{100} * min(N) - \frac{1}{9000} * min(Y)$

- Given the current domain this implies that
  $M \leq \frac{9}{9} + \frac{91*9}{9000} + \frac{9}{9000} + \frac{9}{900} - \frac{0}{10} - \frac{0}{100} - \frac{0}{9000} = 1.102$

- Thus the domain of M is updated to `[1..1]`

- S should be greater than $9 * min(M) + \frac{9}{10} * min(O) + \frac{9}{100} * min(N) + \frac{1}{1000} * min(Y) - \frac{91}{1000} * max(E) - \frac{1}{1000} * max(D) - \frac{1}{100} * max(R)$

- Given the current domain, we infer that $S \geq 8.082$

Aspects of Constraint Logic Programming

Theoretical

Logical Foundation – First-Order Logic

Conceptual

Sound Modeling

Practical

Efficient Algorithms/Implementations

Combination of different Solvers

Assets of Constraint Programming

Adaption and combination of existing efficient algorithms from

- **Mathematics**
  - Operations research
  - Graph theory
  - Algebra
- **Computer science**
  - Finite automata
  - Automatic proving
  - Artificial Intelligence
- **Economics**
- **Linguistics**

Application Domains

- **Modeling**
- **Executable Specifications**
- **Solving combinatorial problems**
  - Scheduling, Planning, Timetabling
  - Configuration, Layout, Placement, Design
  - Analysis: Simulation, Verification, Diagnosis
- **Artificial Intelligence**
  - Machine Vision
  - Natural Language Understanding
  - Temporal and Spatial Reasoning
  - Theorem Proving
  - Qualitative Reasoning
  - Robotics
  - Agents
  - Bioinformatics

Research Applications

- **Computer Science:** Program Analysis, Robotics, Agents
- **Molecular Biology, Biochemistry, Bioinformatics:** Protein Folding, Genomic Sequencing
- **Economics:** Scheduling
- **Linguistics:** Parsing
- **Medicine:** Diagnosis Support
- **Physics:** System Modeling
- **Geography:** Geo-Information-Systems

Commercial Applications

- **Lufthansa:** Short-term staff planning.
- **Hongkong Container Harbor:** Resource planning.
- **Renault:** Short-term production planning.
- **Nokia:** Software configuration for mobile phones.
- **Airbus:** Cabin layout.
- **Siemens:** Circuit verification.
- **Caisse d'epargne:** Portfolio management.