Constraint (Logic) Programming languages

Two motivational examples

- 1) A combinatorial problem
- 2) An optimization problem

Example 1

Problem: arrange three 1s, three 2, … three 9s in sequence so that for all i∈[1,9] there are exactly i numbers between successive occurrences of i

A solution:

1,9,1,2,1,8,2,4,6,2,7,9,4,5,8,6,3,4,7,5,3,9,6,8,3,5,7

A Prolog program which solves previous problem:

```
sequence([_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_]).
% Sequence(X) -> X is a lit of 27 variables

question(S):-
    sequence(S),
    sublist([9,_,_,_,_,_,_,_,_,_,9,_,_,_,_,_,_,_,_,_,9],S),
    sublist([8,_,_,_,_,_,_,_,_,8,_,_,_,_,_,_,_,_,8] ,S),
    sublist([7,_,_,_,_,_,_,_,7,_,_,_,_,_,_,_,7] ,S),
    sublist([6,_,_,_,_,_,_,6,_,_,_,_,_,_,6] ,S),
    sublist([5,_,_,_,_,_,5,_,_,_,_,_,5] ,S),
    sublist([4,_,_,_,_,4,_,_,_,_,4] ,S),
    sublist([3,_,_,_,3,_,_,_,3] ,S),
    sublist([2,_,_,2,_,_,2] ,S),
    sublist([1,_,1,_,1] ,S).
% S is a solution
```

Example 2: Cakes problem (a simplified production planning problem)

We need to bake some cakes for a fete at our local school.

We know how to make two sorts of cakes.

1. A banana cake which takes 250g of self-raising flour, 2 mashed bananas, 75g sugar and 100g of butter,
2. A chocolate cake which takes 200g of self-raising flour, 75g of cocoa, 150g sugar and 150g of butter.

We can sell a chocolate cake for $4.50 and a banana cake for $4.00.

We have 4kg self-raising flour, 6 bananas, 2kg of sugar, 500g of butter and 500g of cocoa.

The question is how many of each sort of cake should we bake for the fete in order to maximize the profit.

A MiniZinc program which solves the Cakes problem

CAKES ≡ [DOWNLOAD]

```
% Baking cakes for the school fete

var 0..100: b; % no. of banana cakes
var 0..100: c; % no. of chocolate cakes

% flour
constraint 250*b + 200*c <= 4000;
% bananas
constraint 2*b  <= 6;
% sugar
constraint 75*b + 150*c <= 2000;
% butter
constraint 100*b + 150*c <= 500;
% cocoa
constraint 75*c <= 500;

% maximize our profit
solve maximize 400*b + 450*c;

output ["no. of banana cakes = \(b)\n",
         "no. of chocolate cakes = \(c)\n"];
```

Figure 3: Model for determining how many banana and chocolate cakes to bake for the school fete

Logic Programming

Bob Kowalski: "Algorithm = Logic + Control"

In traditional programming:

programmer takes care of both aspects

In declarative programming:

programmer takes care only of Logic:

interpreter of the language takes care of Control

Declarative programming: three paradigms

1. Logic Programming (Prolog …)

2. Functional programming (ML, Haskel, OCAML …)

3. Constraint Programming (MinZinc, CLP, ILOG, …)

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. E.C. Freuder

Declarative programming

The task of the programmer is to specify the problem to be solved: we have to ``declare'':

what we want to obtain

and not

how we achieve that

Constraint Reasoning

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:
  $x \in \{0, 1, \ldots, 9\} \;\wedge\; x \geq 5 \;\rightarrow\; x \in \{5, 6, 7, 8, 9\}$

Constraint Programming

Robust, flexible, maintainable software faster.

- *Declarative modeling by constraints:*
  Description of properties and relationships between partially known objects.
  Correct handling of precise and imprecise, finite and infinite, partial and full information.
- *Automatic constraint reasoning:*
  Propagation of the effects of new information (as constraints).
  Simplification makes implicit information explicit.
- *Solving combinatorial problems efficiently:*
  Easy Combination of constraint solving with search and optimization.

Terminology

Language is first-order logic with equality.

- *Constraint:*
  Conjunction of atomic constraints (predicates)
  E.g., $4X + 3Y = 10 \ \wedge \ 2X - Y = 0$
- *Constraint Problem (Query):*
  A given, initial constraint
- *Constraint Solution (Answer):*
  A valuation for the variables in a given constraint problem that satisfies all constraints of the problem. E.g., $X = 1 \wedge Y = 2$

In general, a normal/solved form of, e.g., the problem
$4X + 3Y + Z = 10 \ \wedge \ 2X - Y = 0$ simplifies into
$Y + Z = 10 \ \wedge \ 2X - Y = 0$

Constraint Reasoning and Constraint Programming

A generic framework for

- Modeling
  - with partial information
  - with infinite information
- Reasoning
  - with new information
- Solving
  - combinatorial problems

Two main classes of problems

Constraint Satisfaction Problems (CSP)

- studied in artificial intelligence since 1970s
- A CSP is defined by:

  a finite set of variables $\{X_1, \ldots, X_n\}$
  a set of values (Domain) for each variable $D(X_1), \ldots D(X_n)$
  A set of constraints $\{C1_, \ldots, C_m\}$
- A solution to a CSP is a complete assignment to all the variables that satisfies the constraints.
- e.g. $X \in \{1,2\} \wedge Y \in \{1,2\} \wedge Z \in \{1,2\} \wedge X = Y \wedge X \neq Z \wedge Y > Z$

Constraint Optimization Problems (COP)

- A COP is a CSP defined on the variables $\{X_1, \ldots, X_n\}$ and domains $D(X_1), \ldots D(X_n)$, together with an objective function $f : D(X_1), \ldots D(X_n) \rightarrow D$. A solution to a COP is a solution of the CSP that optimize the value of f.

Two main family of constraint languages

Constraint Logic Programming (CLP)

- developed in the mid-1980s. Simple idea: add constraints (and the related solvers) to logic programming
- two declarative paradigms together: constraint solving and logic programming
- more expressive, flexible, and in general more efficient than logic programs
- e.g. $X - Y = 3 \ \wedge \ X + Y = 7$ leads to $X = 5 \ \wedge \ Y = 2$
- e.g. $X < Y \ \wedge \ Y < X$ fails without the need to know values
- modern Prolog implementations are CLP

Imperative languages with constraints

- Integration of constraints (and the related solvers) to imperative languages
- Commercial distributions and applications
- One example (free): MinZinc

Constraint Solving

The *solver* is an essential part of constraint languages: it solves the constraints by adapting and combining existing efficient algorithms from

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

There are many different constraint domains, with different solvers.

Early History of Constraint Programming

60s, 70s Constraint networks in artificial intelligence.
70s Logic programming (Prolog).
80s Constraint logic programming.
80s Concurrent logic programming.
90s Concurrent constraint programming.
90s Commercial applications.

Application Domains

- Modeling
- Executable Specifications
- Solving combinatorial problems
  - Scheduling, Planning, Timetabling
  - Configuration, Layout, Placement, Design
  - Analysis: Simulation, Verification, Diagnosis

of software, hardware and industrial processes.

Application Domains (cont)

- Artificial Intelligence
  - Machine Vision
  - Natural Language Understanding
  - Temporal and Spatial Reasoning
  - Theorem Proving
  - Qualitative Reasoning
  - Robotics
  - Agents
  - Bio-informatics

Applications in Research

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

Early Commercial Applications

- *Lufthansa:* Short-term staff planning.
- *Hong Kong 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.

In *Decision Support Systems* for Planning, Configuration, for Design, Analysis.