Constraint Programming

Basic idea: Programming with constraints, i.e. constraint solving embedded in a programming language.

Constraints: linear, non-linear, finite domain, Boolean, ...

Programming: logic, functional, object-oriented, imperative, concurrent, ...

Mathematical programming vs. computer programming.

Systems: Prolog III/IV, CHIP, ECLIPSE, ILOG, CHOCO, Gecode, JaCoP, MiniZinc ...

Finite Domain Constraints

Constraint satisfaction problem (CSP)

- $n$ variables $x_1, ..., x_n$
- For each variable $x_j$ a finite domain $D_j$ of possible values, often $D_j \subset \mathbb{N}$.
- $m$ constraints $C_1, ..., C_m$, where $C_j \subseteq D_{i_1} \times \cdots \times D_{i_k}$ is a relation between $k_i$ variables $x_{i_1}, ..., x_{i_k}$. Write also $C_{i_1, ..., i_k}$.
- A solution is an assignment of a value $D_j$ to $x_j$, for each $j = 1, ..., n$, such that all relations $C_i$ are satisfied.

Coloring Problem

- Decide whether a map can be colored by 3 colors such that neighboring regions get different colors.
- For each region a variable $x_j$ with domain $D_j = \{\text{red, green, blue}\}$.
- For each pair of variables $x_i, x_j$ corresponding to two neighboring regions, a constraint $x_i \neq x_j$.
- NP-complete problem.

Resolution by Backtracking

- Instantiate the variables in some order.
- As soon as all variables in a constraint are instantiated, determine its truth value.
- If the constraint is not satisfied, backtrack to the last variable whose domain contains unassigned values, otherwise continue instantiation.

Efficiency Problems

Mackworth 77

1. If the domain $D_j$ of a variable $x_j$ contains a value $v$ that does not satisfy $C_j$, this will be the cause of repeated instantiation followed by immediate failure.

2. If we instantiate the variables in the order $x_1, x_2, ..., x_n$, and for $x_i = v$ there is no value $w \in D_j$, for $j > i$, such that $C_j(v, w)$ is satisfied, then backtracking will try all values for $x_j$, fail and try all values for $x_{j-1}$ (and for each value of $x_{j-1}$ again all values for $x_j$) and so on until it tries all combinations of values for $x_{i+1}, ..., x_j$ before finally discovering that $v$ is not a possible value for $x_j$.

The identical failure process may be repeated for all other sets of values for $x_1, ..., x_{i-1}$ with $x_i = v$. 

Local Consistency

- Consider CSP with unary and binary constraints only.
- **Constraint graph G**
  - For each variable \(x_i\), a node \(i\).
  - For each pair of variables \(x_i, x_j\) occurring in the same binary constraint, two arcs \((i, j)\) and \((j, i)\).
- The node \(i\) is **consistent** if \(C_i(v)\), for all \(v \in D_i\).
- The arc \((i, j)\) is **consistent**, if for all \(v \in D_i\) with \(C_i(v)\) there exists \(w \in D_j\) with \(C_j(w)\) such that \(C_{ij}(v, w)\).
- The graph is **node consistent** resp. **arc consistent** if all its nodes (resp. arcs) are consistent.

Arc Consistency

**Algorithm AC-3** (Mackworth 77):
begin
for \(i \leftarrow 1\) until \(n\) do \(D_i \leftarrow \{v \in D_i \mid C_i(v)\}\);
\(Q \leftarrow \{(i, j) \mid (i, j) \in \text{arcs}(G), i \neq j\}\)
while \(Q\) not empty do
begin
    select and delete an arc \((i, j)\) from \(Q\);
    if \(\text{REVISE}(i, j)\) then
        \(Q \leftarrow Q \cup \{(k, i) \mid (k, i) \in \text{arcs}(G), k \neq i, k \neq j\}\)
end
end

**Arc Consistency (2)**

procedure \(\text{REVISE}(i, j)\):
begin
    \(\text{DELETE} \leftarrow \text{false}\)
    for each \(v \in D_i\) do
        if there is no \(w \in D_j\) such that \(C_{ij}(v, w)\) then
            begin
                delete \(v\) from \(D_i\);
                \(\text{DELETE} \leftarrow \text{true}\)
            end;
    return \(\text{DELETE}\)
end
Crossword Puzzle

Word List

1 Across
Aft
Laser
Hoses 4
Laser 5
Sails 1
Sheet 25
Steer 24

2 Down
Hoses 4
Laser 5
Sails 1
Sheet 25
Steer 24

3 Down
Hoses 3
Laser 6
Sails 12
Sheet 28
Steer

4 Across
Heel 10
Hike
Keel 11
Knot 9
Line 19

5 Down
Heel 10
Hike
Keel 11
Knot 9
Line 19

6 Down
Aft 17
Ale 18
Eel 20
Lee
Tie 19

7 Across
Aft 17
Ale 18
Eel 20
Lee
Tie 19

8 Across
Hoses 27
Laser
Sails 22
Sheet 23
Steer 24

Solution

Lookahead

Apply local consistency dynamically during search

- **Forward Checking:** After assigning to $x$ the value $v$, eliminate for all uninstantiated variables $y$ the values from $D_y$ that are incompatible with $v$.

- **Partial Lookahead:** Establish arc consistency for all $(y, y')$, where $y, y'$ have not been instantiated yet and $y$ will be instantiated before $y'$.

- **Full Lookahead:** Establish arc consistency for all uninstantiated variables.
n-Queens Problem

Place \( n \) queens in an \( n \times n \) chessboard such that no two queens threaten each other.

- **Variables** \( x_i, i = 1, \ldots, n \) with domain \( D_i = \{1, \ldots, n\} \) indicating the column of the queen in line \( i \).

- **Constraints**
  - \( x_i \neq x_j \), for \( 1 \leq i < j \leq n \) (vertical)
  - \( x_i \neq x_j + (j - i) \), for \( 1 \leq i < j \leq n \) (diagonal 1)
  - \( x_i \neq x_j - (j - i) \), for \( 1 \leq i < j \leq n \) (diagonal 2)

**Forward Checking**

**Partial Lookahead**
Full Lookahead

Typical structure of a constraint program

1. Declare the variables and their domains
2. State the constraints
3. Enumeration (labeling)

The constraint solver achieves only local consistency.
In order to get global consistency, the domains have to be enumerated.

Labeling

1. Assigning to the variables their possible values and constructing the corresponding search tree.
2. Important questions
   1. In which order should the variables be instantiated (variable selection)?
   2. In which order should the values be assigned to a selected variable (value selection)?
3. Static vs. dynamic orderings
4. Heuristics

Dynamic variable/value orderings

1. Variable orderings
   - Choose the variable with the smallest domain "first fail"
   - Choose the variable with the smallest domain that occurs in most of the constraints "most constrained"
   - Choose the variable which has the smallest/largest lower/upper bound on its domain.
• Value orderings
  - Try first the minimal value in the current domain.
  - Try first the maximal value in the current domain.
  - Try first some value in the middle of the current domain.

Some constraint programming systems

<table>
<thead>
<tr>
<th>System</th>
<th>Avail.</th>
<th>Constraints</th>
<th>Language</th>
<th>Web site</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-prolog</td>
<td>comm.</td>
<td>FinDom</td>
<td>Prolog</td>
<td><a href="http://www.probp.com">www.probp.com</a></td>
</tr>
<tr>
<td>CHiP</td>
<td>comm.</td>
<td>FinDom, Boolean, Linear Q</td>
<td>Prolog, C, C++</td>
<td><a href="http://www.cosytec.com">www.cosytec.com</a></td>
</tr>
<tr>
<td>Choco</td>
<td>free</td>
<td>FinDom</td>
<td>Java</td>
<td>choco.emn.fr</td>
</tr>
<tr>
<td>Eclipse</td>
<td>free non-profit</td>
<td>FinDom, Hybrid</td>
<td>Prolog</td>
<td>eclipseclp.org</td>
</tr>
<tr>
<td>Gecode</td>
<td>free</td>
<td>FinDom</td>
<td>C++</td>
<td><a href="http://www.gecode.org">www.gecode.org</a></td>
</tr>
<tr>
<td>GNU Prolog</td>
<td>free</td>
<td>FinDom</td>
<td>Prolog</td>
<td>gnu-prolog.inria.fr</td>
</tr>
<tr>
<td>JaCoP</td>
<td>free</td>
<td>FinDom</td>
<td>Java</td>
<td>jacop.osolpro.com</td>
</tr>
<tr>
<td>MiniZinc</td>
<td>free</td>
<td>FinDom, Arithmetic</td>
<td></td>
<td>g12.cs.mu.oz.au/minizinc</td>
</tr>
<tr>
<td>Mozart</td>
<td>free</td>
<td>FinDom</td>
<td>Oz</td>
<td><a href="http://www.mozart-oz.org">www.mozart-oz.org</a></td>
</tr>
<tr>
<td>NCL</td>
<td>comm.</td>
<td>FinDom</td>
<td></td>
<td><a href="http://www.enginest.com">www.enginest.com</a></td>
</tr>
<tr>
<td>Prolog IV</td>
<td>free</td>
<td>FinDom, Arithmetic</td>
<td>Prolog</td>
<td>prolog-heritage.org</td>
</tr>
<tr>
<td>SCIP</td>
<td>free</td>
<td>Hybrid</td>
<td></td>
<td>scip.zib.de</td>
</tr>
<tr>
<td>Sicstus</td>
<td>comm.</td>
<td>FinDom, Boolean, linear R/Q</td>
<td>Prolog</td>
<td><a href="http://www.sics.se/sicstus/">www.sics.se/sicstus/</a></td>
</tr>
</tbody>
</table>