Lecture 2

CP in a Nutshell
Outline

- Introduce notion of equivalence of CSP's
- Provide intuitive introduction to general methods of Constraint Programming
- Introduce basic framework for Constraint Programming
- Illustrate this framework by 2 examples
Projection

Given: variables $X := x_1, ..., x_n$ with domains $D_1, ..., D_n$
Consider
- $d := (d_1, ..., d_n) \in D_1 \times ... \times D_n$
- subsequence $Y := x_{i_1}, ..., x_{i_l}$ of $X$

Denote $(d_{i_1}, ..., d_{i_l})$ by $d[Y]$: projection of $d$ on $Y$

In particular: $d[x_i] = d_i$

Note: For a CSP

$$\mathcal{P} := \langle C ; x_1 \in D_1, ..., x_n \in D_n \rangle$$

$(d_1, ..., d_n) \in D_1 \times ... \times D_n$ is a solution to $\mathcal{P}$ iff for each constraint $C$ of $\mathcal{P}$ on a sequence of variables $Y$

$d[Y] \in C$
Equivalence of CSP's

- $\mathcal{P}_1$ and $\mathcal{P}_2$ are equivalent if they have the same set of solutions

- CSP's $\mathcal{P}_1$ and $\mathcal{P}_2$ are equivalent w.r.t. $X$ iff
  \[ \{d[X] \mid d \text{ is a solution to } \mathcal{P}_1\} = \{d[X] \mid d \text{ is a solution to } \mathcal{P}_2\} \]

- Union of $\mathcal{P}_1$, ..., $\mathcal{P}_m$ is equivalent w.r.t. $X$ to $\mathcal{P}_0$ if
  \[ \{d[X] \mid d \text{ is a solution to } \mathcal{P}_0\} = \bigcup_{i=1}^{m} \{d[X] \mid d \text{ is a solution to } \mathcal{P}_i\} \]
Solved and Failed CSP's

- $C$ a constraint on variables $y_1, \ldots, y_k$ with domains $D_1, \ldots, D_k$ (so $C \subseteq D_1 \times \ldots \times D_k$):
  - $C$ is **solved** if $C = D_1 \times \ldots \times D_k$

- CSP is **solved** if
  - all its constraints are solved, and
  - no domain of it is empty

- CSP is **failed** if
  - it contains the false constraint $\bot$, or
  - some of its domains is empty
procedure solve
var continue := true
begin
   while continue and not happy do
       Preprocess;
       Constraint Propagation;
       if not happy then
           if Atomic then continue := false
           else
               Split; Proceed by Cases
           end-if
   end-while
end
Preprocess

Bring to desired syntactic form

- Example: Constraints on reals
  - Desired syntactic form: no repeated occurrences of a variable

\[ ax^7 + bx^5y + cy^{10} = 0 \]

\[ \rightarrow ax^7 + z + cy^{10} = 0, \ bx^5y = z \]
Happy

- Found a solution
- Found all solutions
- Found a solved form from which one can generate all solutions
- Determined that no solution exists (inconsistency)
- Found best solution
- Found all best solutions
- Reduced all interval domains to sizes $< \varepsilon$
Atomic and Split

- Check whether CSP is amenable for splitting, or
- whether search ‘under’ this CSP is still needed

Split a domain:

- \( D \) finite (Enumeration)
  \[
  \begin{align*}
  x \in D & \quad | \quad x \in D - \{ a \} \\
  x \in \{ a \} & \quad | \quad x \in \{ a_k \}
  \end{align*}
  \]

- \( D \) finite (Labeling)
  \[
  \begin{align*}
  x \in \{ a_1, \ldots, a_k \} & \quad | \quad x \in \{ a_k \}
  \end{align*}
  \]

- \( D \) interval of reals (Bisection)
  \[
  \begin{align*}
  x \in [a..b] & \quad | \quad x \in \left[ \frac{a+b}{2} \right] + 1..b
  \end{align*}
  \]
Split a constraint:

- Disjunctive constraints
  \[
  \frac{C_1 \lor C_2}{C_1 \mid C_2}
  \]

- Constraints in “compound” form
  Example:
  \[
  |p(\bar{x})|=a \\
  p(\bar{x})=a \mid p(\bar{x})=-a
  \]
Effect of Split

- Each Split replaces current CSP $\mathcal{P}$ by CSP's $\mathcal{P}_1$, ..., $\mathcal{P}_n$ such that the union of $\mathcal{P}_1$, ..., $\mathcal{P}_n$ is equivalent to $\mathcal{P}$.

- Example:
  Enumeration replaces $\langle C ; \mathcal{D}\mathcal{E}, x \in D \rangle$ by $\langle C' ; \mathcal{D}\mathcal{E}, x \in \{a\} \rangle$ and $\langle C'' ; \mathcal{D}\mathcal{E}, x \in D - \{a\} \rangle$ where $C'$ and $C''$ are restrictions of the constraints from $C$ to the new domains.
Heuristics

Which
- variable to choose
- value to choose
- constraint to split

Examples:
- Select a variable that appears in the largest number of constraints (most constrained variable)
- For a domain being an integer interval: select the middle value
Proceed by Cases

Various search techniques
- Backtracking
- Branch and bound
- Can be combined with Constraint Propagation
- Intelligent backtracking
Backtracking

- Nodes generated “on the fly”
- Nodes are CSP's
- Leaves are CSP's that are solved or failed
Branch and Bound

- Modification of backtracking aiming at finding the optimal solution
- Takes into account objective function
- Maintain currently best value of the objective function in variable bound initialized to $-\infty$ and updated each time a better solution found
- Used in combination with heuristic function

Conditions on heuristic function $h$:
- If $\psi$ is a direct descendant of $\phi$, then
  \[ h(\psi) \leq h(\phi) \]
- If $\psi$ is solved CSP with singleton set domains, then
  \[ \text{obj}(\psi) \leq h(\psi) \]

$h$ allows us to prune the search tree
Illustration

$h(\psi) \leq \text{bound}$
Constraint Propagation

Replace a CSP by an equivalent one that is “simpler”

Constraint propagation performed by repeatedly reducing
- domains
- constraints
while maintaining equivalence
Reduce a Domain: Examples

- Projection rule:
  Take a constraint $C$ and choose a variable $x$ of it with domain $D$.
  Remove from $D$ all values for $x$ that do not participate in a solution to $C$.

- Linear inequalities on integers:

  \[
  \begin{align*}
  &x < y; x \in [50..200], y \in [0..100] \\
  &x < y; x \in [50..99], y \in [51..100]
  \end{align*}
  \]
Repeated Domain Reduction: Example

Consider
\[ x < y, \ y < z ; \ x \in [50..200], \ y \in [0..100], \ z \in [0..100] \]

Apply above rule to \( x < y \):
\[ x < y, \ y < z ; \ x \in [50..99], \ y \in [51..100], \ z \in [0..100] \]

Apply it now to \( y < z \):
\[ x < y, \ y < z ; \ x \in [50..99], \ y \in [51..99], \ z \in [52..100] \]

Apply it again to \( x < y \):
\[ x < y, \ y < z ; \ x \in [50..98], \ y \in [51..99], \ z \in [52..100] \]
Reduce Constraints

Usually by introducing new constraints!

- Transitivity of $\prec$:
  \[
  \frac{\langle x < y, y < z; \mathcal{D} \mathcal{E} \rangle}{\langle x < y, y < z, x < z; \mathcal{D} \mathcal{E} \rangle}
  \]
  This rule introduces new constraint $x < z$

- Resolution rule:
  \[
  \frac{\langle C_1 \vee L, C_2 \vee \bar{L}; \mathcal{D} \mathcal{E} \rangle}{\langle C_1 \vee L, C_2 \vee \bar{L}, C_1 \vee C_2; \mathcal{D} \mathcal{E} \rangle}
  \]
  This rule introduces new constraint $C_1 \vee C_2$
Constraint Propagation Algorithms

- Deal with scheduling of atomic reduction steps
- Try to avoid useless applications of atomic reduction steps
- Stopping criterion for general CSP's: a local consistency notion

Example:
Local consistency criterion corresponding to the projection rule is Hyper-arc consistency:
For every constraint $C$ and every variable $x$ with domain $D$, each value for $x$ from $D$ participates in a solution to $C$. 
Example: Boolean Constraints

Happy: found all solutions
Desired syntactic form (for preprocessing):
- $x = y$
- $\neg x = y$
- $x \land y = z$
- $x \lor y = z$

Preprocessing:

$$\frac{x \land s = z}{x \land y = z, s = y}$$

Constraint propagation:

$$\frac{x \land y = z; x \in D_x, y \in D_y, z \in \{1\}}{; x \in D_x \cap \{1\}, y \in D_y \cap \{1\}, z \in \{1\}}$$

(write as $x \land y = z, z = 1 \Rightarrow x = 1, y = 1$)
Boolean Constraints, ctd

- $x = y, x = 1 \iff y = 1$
- $x = y, y = 1 \iff x = 1$
- $x = y, x = 0 \iff y = 0$
- $x = y, y = 0 \iff x = 0$
- $x \land y = z, x = 1, y = 1 \iff z = 1$
- $x \land y = z, x = 1, z = 0 \iff y = 0$
- $x \land y = z, y = 1, z = 0 \iff x = 0$
- $x \land y = z, y = 0 \iff z = 0$
- $x \land y = z, z = 1 \iff x = 1, y = 1$
- $\neg x = y, x = 1 \iff y = 0$
- $\neg x = y, x = 0 \iff y = 1$
- $\neg x = y, y = 1 \iff x = 0$
- $\neg x = y, y = 0 \iff x = 1$
- $x \lor y = z, x = 1 \iff z = 1$
- $x \lor y = z, x = 0, y = 0 \iff z = 0$
- $x \lor y = z, y = 0, z = 1 \iff x = 1$
- $x \lor y = z, z = 0 \iff x = 0, y = 0$
- $x \lor y = z, y = 1 \iff z = 1$
- $x \lor y = z, z = 1 \iff y = 1$
Boolean Constraints, ctd

**Split:**
- Choose the most constrained variable
- Apply the labeling rule:

\[
x \in \{0,1\} \quad \rightarrow \quad x \in \{0\} \quad \text{or} \quad x \in \{1\}
\]

**Proceed by cases:** backtrack
Example: Polynomial Constraints on Integer Intervals

Domains: integer intervals \([a..b]\)
\[a..b] := \{x \in \mathbb{Z} \mid a \leq x \leq b\]

Constraints:
\[s = 0\]
s is a polynomial (possibly in several variables) with integer coefficients
Example:
\[2 \cdot x^5 \cdot y^2 \cdot z^4 + 3 \cdot x \cdot y^3 \cdot z^5 - 4 \cdot x^4 \cdot y^6 \cdot z^2 + 10 = 0\]

Objective function: a polynomial
Example

Find a solution to
\[ x^3 + y^2 - z^3 = 0 \]
in \([1..1000]\) such that
\[ 2 \cdot x \cdot y - z \]
is maximal.

Answer:
\[ x = 112, \ y = 832, \ z = 128 \]
Desired syntactic form:

- $\sum_{i=1}^{n} a_i x_i = b$
- $x \cdot y = z$

**Preprocess:**

Use appropriate transformation rules

**Example:**

\[
\left\{ \sum_{i=1}^{n} m_i = 0 ; D E \right\}
\]

\[
\sum_{i=1}^{n} v_i = 0, m_1 = v_1, \ldots, m_n = v_n ; D E, v_1 \in \mathbb{Z}, \ldots, v_n \in \mathbb{Z}
\]

where

- some $m_i$ is not of the form $ax_i$
- $v_1, \ldots, v_n$ do not appear in $DE$

**Happy:** found an optimal solution w.r.t. the objective function
Polynomial Constraints on Integer Intervals, ctd

**Constraint propagation:** uses interval arithmetic

- **addition:**
  \[ X + Y := \{ x + y \mid x \in X, y \in Y \} \]

- **subtraction:**
  \[ X - Y := \{ x - y \mid x \in X, y \in Y \} \]

- **multiplication:**
  \[ X \cdot Y := \{ x \cdot y \mid x \in X, y \in Y \} \]

- **division:**
  \[ X/Y := \{ u \in \mathbb{Z} \mid \exists x \in X \exists y \in Y : u \cdot y = x \} \]
Interval Arithmetic, ctd

Given: $X$, $Y$ integer intervals, $a$ an integer
- $X \cap Y$, $X + Y$, $X - Y$ are integer intervals
- $X/\{a\}$ is an integer interval
- $X \cdot Y$ does not have to be an integer interval, even if $X = \{a\}$ or $Y = \{a\}$
- $X/Y$ does not have to be an integer interval

Examples:

$[2..4] + [3..8] = [5..12]$

$[3..7] - [1..8] = [-5..6]$

$[3..3] \cdot [1..2] = \{3, 6\}$

$[3..5]/[-1..2] = \{-5, -4, -3, 2, 3, 4, 5\}$

$[-3..5]/[-1..2] = \mathbb{Z}$
Turning Sets to Intervals

\[ \text{int}(X) := \begin{cases} 
\text{smallest int. interval} \supseteq X & \text{if } X \text{ finite} \\
\mathbb{Z} & \text{otherwise}
\end{cases} \]

Examples:
\[
\text{int}([3..3] \cdot [1..2]) = [3..6] \\
\text{int}([3..5]/[-1..2]) = [-5..5] \\
\text{int}([-3..5]/[-1..2]) = \mathbb{Z}
\]
Rule for Linear Equality

\[
\begin{align*}
\sum_{i=1}^{n} a_i x_i &= b ; x_1 \in D_1, \ldots, x_n \in D_n \\
\langle \sum_{i=1}^{n} a_i x_i &= b ; \ldots, x_j \in D'_j, \ldots \rangle
\end{align*}
\]

where \( j \in [1..n] \), and

\[
D'_j := D_j \cap \frac{b - \sum_{i \in [1..n]-\{j\}} \text{int}(a_i \cdot D_i)}{a_j}
\]
Multiplication Rules

Multiplication 1
\[ \langle x \cdot y = z ; x \in D_x, y \in D_y, z \in D_z \rangle \]
\[ \langle x \cdot y = z ; x \in D_x, y \in D_y, z \in D_z \cap \text{int}(D_x \cdot D_y) \rangle \]

Multiplication 2
\[ \langle x \cdot y = z ; x \in D_x, y \in D_y, z \in D_z \rangle \]
\[ \langle x \cdot y = z ; x \in D_x \cap \text{int}(D_z/D_y), y \in D_y, z \in D_z \rangle \]

Multiplication 3
\[ \langle x \cdot y = z ; x \in D_x, y \in D_y, z \in D_z \rangle \]
\[ \langle x \cdot y = z ; x \in D_x \cap \text{int}(D_z/D_x), y \in D_y, z \in D_z \rangle \]
Effect of Multiplication Rules

Consider

\[ \langle x \cdot y = z \ ; \ x \in [1..20], y \in [9..11], z \in [155..161] \rangle \]

Using Multiplication Rules we can transform this to

\[ \langle x \cdot y = z \ ; \ x \in [16..16], y \in [10..10], z \in [160..160] \rangle \]
Polynomial Constraints on Integer Intervals, ctd

Split:
- Choose the variable with the smallest interval domain
- Apply the bisection rule:

\[
\begin{align*}
x & \in [a..b] \\
x & \in [a..\frac{a+b}{2}] \mid x \in \left[\frac{a+b}{2}+1..b\right]
\end{align*}
\]

where \(a < b\)

- Proceed by cases: branch and bound
More on Interval Arithmetic

Given objective function $\text{obj}$.

$\text{obj}^+$: extension of $\text{obj}$ to function from sets of integers to sets of integers.

Example: $\text{obj}(x,y) := x^2 \cdot y - 3x \cdot y^2 + 5$

Then $\text{obj}^+(X,Y) = X \cdot X \cdot Y - 3 \cdot X \cdot Y \cdot Y + 5$

Lemma
Consider integer intervals $X_1, \ldots, X_n$

- $\text{obj}^+(X_1, \ldots, X_n)$ is a finite set of integers
- For all $a_i \in X_i$, $i \in [1..n]$ 
  $\text{obj}(a_1, \ldots, a_n) \in \text{obj}^+(X_1, \ldots, X_n)$
- For all $Y_i \subseteq X_i$, $i \in [1..n]$ 
  $\text{obj}^+(Y_1, \ldots, Y_n) \subseteq \text{obj}^+(X_1, \ldots, X_n)$
Heuristic Function

Take

- $\mathcal{P} := \langle C ; x_1 \in D_1, ..., x_n \in D_n \rangle$, with $D_1, ..., D_n$ integer intervals
- $\text{obj}$: polynomial with variables $x_1, ..., x_n$

Define

$$h(\mathcal{P}) := \max(\text{obj}^+(D_1, ..., D_n))$$

Thanks to the preceding lemma, $h$ satisfies the conditions for the heuristic function (cf. Slide 15).
Objectives

- Introduce notion of equivalence of CSP's
- Provide intuitive introduction to general methods of Constraint Programming
- Introduce a basic framework for Constraint Programming
- Illustrate this framework by 2 examples