Principles of Constraint Programming

Krzysztof R. Apt

Chapter 3
Constraint Programming in a Nutshell

Objectives

- 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.

Projections

- **Given**: variables $X := x_1, \ldots, x_n$ with the domains $D_1, \ldots, D_n$.

  Consider

  – $d := (d_1, \ldots, d_n) \in D_1 \times \ldots \times D_n$,

  – subsequence $Y := x_{i_1}, \ldots, x_{i_\ell}$ of X.

  Denote $(d_{i_1}, \ldots, d_{i_\ell})$ by $d[Y]$.

  $d[Y]$: **projection** of d on Y.

  In particular: $d[x_i] = d_i$.

- **Note** For a CSP

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

  $(d_1, \ldots, d_n) \in D_1 \times \ldots \times D_n$ 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 $\mathcal{P}_1, \ldots, \mathcal{P}_m$ 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,
  - no domain of it is empty.
- CSP is **failed** if
  - it contains the false constraint $\perp$,

  or

  - some of its domains is empty.

Constraint Programming: Basic Framework

Formulate your problem as a CSP;

Solve:

```
VAR CONTINUE: BOOLEAN;
CONTINUE:= TRUE;
WHILE CONTINUE AND NOT Happy DO
  Preprocess;
  Constraint Propagation;
  IF NOT Happy
  THEN
    IF Atomic
    THEN
      CONTINUE:= FALSE
    ELSE
      Split;
      Proceed by Cases
    END
  END
END
```

- CONTINUE is local to Solve.
- Proceed by Cases leads to a recursive call of **Solve** for each newly formed CSP.

Preprocess

Bring to desired syntactic form.

- **Example**: Constraints on **reals**.

  Desired syntactic form: no repeated occurrences of a variable.

$$\frac{ax^7 + bx^5y + cy^{10} = 0}{ax^7 + z + cy^{10} = 0,\ bx^5y = z}$$

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 $< \epsilon$.

Atomic

Check

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

Split

- Split a **domain.**
  - D finite (**Enumeration**)

$$\frac{x \in D}{x \in \{a\} \mid x \in D - \{a\}}$$

  - D finite (**Labeling**)

$$\frac{x \in \{a_1, \ldots, a_k\}}{x \in \{a_1\} \mid \ldots \mid x \in \{a_k\}}$$

  - D interval of reals (**Bisection**)

$$\frac{x \in [a..b]}{x \in [a..\frac{a+b}{2}] \mid x \in [\frac{a+b}{2}..b]}$$

Split

- Split a **constraint.**
  - Disjunctive constraints

Example:

$$\texttt{Start[task}_1\texttt{]} + \texttt{Duration[task}_1\texttt{]} \le \texttt{Start[task}_2\texttt{]} \vee$$
$$\texttt{Start[task}_2\texttt{]} + \texttt{Duration[task}_2\texttt{]} \le \texttt{Start[task}_1\texttt{]}$$

  - Constraints in "compound" form

Example:

$$\frac{|p(\bar{x})| = a}{p(\bar{x}) = a \mid p(\bar{x}) = -a}$$

$$\frac{C_1 \vee C_2}{C_1 \mid C_2}$$

Effect of Split

- Each call to SPLIT replaces current CSP $\mathcal{P}$ by CSP's $\mathcal{P}_1, \ldots, \mathcal{P}_n$ such that the union of $\mathcal{P}_1, \ldots, \mathcal{P}_n$ is equivalent to $\mathcal{P}$.
- **Example**: Enumeration.

  It replaces

  $$\langle \mathcal{C} \ ; \ \mathcal{DE}, x \in D \rangle$$

  by

  $$\langle \mathcal{C}' \ ; \ \mathcal{DE}, x \in \{a\} \rangle$$

  and

  $$\langle \mathcal{C}'' \ ; \ \mathcal{DE}, x \in D - \{a\} \rangle.$$

  where $\mathcal{C}'$ and $\mathcal{C}''$ are restrictions of the constraints from $\mathcal{C}$ to the new domains.
- SPLIT also determines in which operation is to be applied next.

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**).

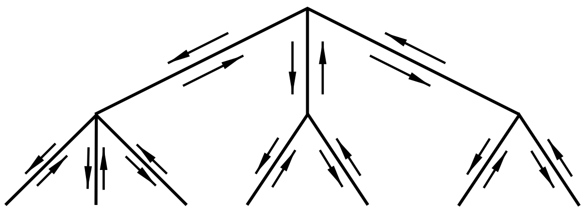
– For a domain being an integer interval: select the middle value.

Proceed by Cases

Various **search** techniques.

- Backtracking,
- Branch and bound,
- Can be combined with Constraint Propagation
- Intelligent backtracking

Backtracking



Here

- 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** (here maximal) solution.
- Takes into account **objective function**.
- One maintains **currently best value** of the objective function in variable `bound`.
- `bound` initialized to $-\infty$ and updated each time a better solution found.
- Used in combination with **heuristic function**.
- Conditions on heuristic function h:

  A. If ψ is a direct descendant of ϕ, then

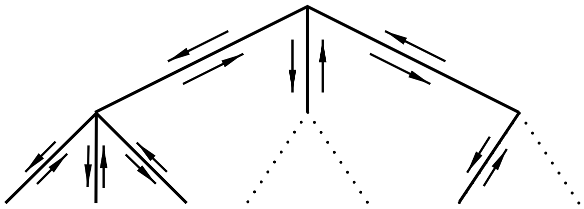
  $$h(\psi) \leq h(\phi).$$

  B. If ψ is solved CSP with singleton set domains, then

  $$obj(\psi) \leq h(\psi).$$

- h allows us to prune the search tree.

Illustration



Constraint Propagation

Intuition: Replace a CSP by an equivalent one that is "simpler".

Constraint propagation performed by repeatedly **reducing**

- domains

and/or

- constraints

while maintaining **equivalence**.

Reduce a Domain: Examples

- Arbitrary CSP's.

  Projection rule:

  Take a constraint C. 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.

$$\frac{\langle x < y \; ; \; x \in [50..200], y \in [0..100]\rangle}{\langle x < y \; ; \; x \in [50..\mathbf{99}], y \in [\mathbf{51}..100]\rangle}$$

  More generally:

$$\frac{\langle x < y \; ; \; x \in [l_x..h_x], y \in [l_y..h_y]\rangle}{\langle x < y \; ; \; x \in [l_x..h'_x], y \in [l'_y..h_y]\rangle}$$

  where

$$h'_x = min(h_x, h_y - 1),$$
$$l'_y = max(l_y, l_x + 1.$$

Repeated Domain Reduction: Example

Consider

$\langle \mathbf{x} < \mathbf{y}, y < z \,;\, x \in [50..200], y \in [0..100], z \in [0..100] \rangle$.

Apply above rule to $x < y$:

$\langle x < y, \mathbf{y} < \mathbf{z} \,;\; x \in [50..99], y \in [51..100], z \in [0..100] \rangle$.

Apply it now to $y < z$:

$\langle \mathbf{x} < \mathbf{y}, y < z \,;\; x \in [50..99], y \in [51..99], z \in [52..100] \rangle$.

Apply it again to $x < y$:

$\langle x < y, y < z \,;\; x \in [50..98], y \in [51..99], z \in [52..100] \rangle$.

Reduce Constraints

Usually by introducing **new** constraints.

- Transitivity of $<$

$$\frac{\langle x < y, y < z \; ; \; \mathcal{DE} \rangle}{\langle x < y, y < z, x < z \; ; \; \mathcal{DE} \rangle}$$

  This rule introduces new constraint, $x < z$.

- Resolution rule.

  Let

  $\overline{\neg x} := x$,

  $\bar{x} := \neg x$.

  C_1 and C_2 clauses (disjunctions of literals)

$$\frac{\langle C_1 \vee L, C_2 \vee \bar{L} \; ; \; \mathcal{DE} \rangle}{\langle C_1 \vee L, C_2 \vee \bar{L}, C_1 \vee C_2 \; ; \; \mathcal{DE} \rangle}$$

  This rule introduces new constraint,

  clause $C_1 \vee C_2$.

Constraint Propagation Algorithms

- Deal with scheduling of **atomic** atomic reduction steps.
- Try to avoid useless applications of atomic reduction steps
- Stopping criterion for general CSP's:

  a **local consistency** notion.

  Example:

  Projection rule, so:

  Take a constraint C. 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.

  Corresponding local consistency notion:

  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):

- $x = y$,
- $\neg x = y$,
- $x \wedge y = z$,
- $x \vee y = z$.

Preprocessing:

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

Constraint propagation:

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

Write as

$$x \wedge y = z, z = 1 \rightarrow x = 1, y = 1.$$

Boolean Constraints: Ctd

EQU	*1*	$x = y, x = 1 \rightarrow y = 1$
EQU	*2*	$x = y, y = 1 \rightarrow x = 1$
EQU	*3*	$x = y, x = 0 \rightarrow y = 0$
EQU	*4*	$x = y, y = 0 \rightarrow x = 0$
NOT	*1*	$\neg x = y, x = 1 \rightarrow y = 0$
NOT	*2*	$\neg x = y, x = 0 \rightarrow y = 1$
NOT	*3*	$\neg x = y, y = 1 \rightarrow x = 0$
NOT	*4*	$\neg x = y, y = 0 \rightarrow x = 1$
AND	*1*	$x \wedge y = z, x = 1, y = 1 \rightarrow z = 1$
AND	*2*	$x \wedge y = z, x = 1, z = 0 \rightarrow y = 0$
AND	*3*	$x \wedge y = z, y = 1, z = 0 \rightarrow x = 0$
AND	*4*	$x \wedge y = z, x = 0 \rightarrow z = 0$
AND	*5*	$x \wedge y = z, y = 0 \rightarrow z = 0$
AND	*6*	$x \wedge y = z, z = 1 \rightarrow x = 1, y = 1$
OR	*1*	$x \vee y = z, x = 1 \rightarrow z = 1$
OR	*2*	$x \vee y = z, x = 0, y = 0 \rightarrow z = 0$
OR	*3*	$x \vee y = z, x = 0, z = 1 \rightarrow y = 1$
OR	*4*	$x \vee y = z, y = 0, z = 1 \rightarrow x = 1$
OR	*5*	$x \vee y = z, y = 1 \rightarrow z = 1$
OR	*6*	$x \vee y = z, z = 0 \rightarrow x = 0, y = 0$

Boolean Constraints: Ctd

Split:

- Choose the most constrained variable.
- Apply the labeling rule:

$$\frac{x \in \{0,1\}}{x \in \{0\} \mid x \in \{1\}}$$

Proceed by cases: backtrack.

Example: Polynomial Constraints on Integer Intervals

Domains: integer intervals $[a..b]$.

$$[a..b] := \{x \in \mathcal{Z} \mid a \leq x \leq b\}.$$

Constraints:

$$s = 0,$$

s is a polynomial (in possibly 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:
$x = 112, y = 832, z = 128.$
Then $2 \cdot x \cdot y - z = 186240.$

Polynomial Constraints on Integer Intervals, Ctd

Desired syntactic form:

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

Preprocess:

Use appropriate transformation rules.

Example:

$$\frac{\langle \Sigma_{i=1}^{n} m_i = 0 \ ; \ \mathcal{DE} \rangle}{\langle \Sigma_{i=1}^{n} v_i = 0, \ m_1 = v_1, \ldots, \ m_n = v_n \ ; \ \mathcal{DE}, v_1 \in \mathcal{Z}, \ldots, v_n \in \mathcal{Z} \rangle}$$

where

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

Happy:

an optimal solution w.r.t. the objective function was found.

Polynomial Constraints on Integer Intervals, Ctd

Constraint propagation:
uses interval arithmetic.

X, Y sets of integers.

- 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:

$$X/Y := \{u \in \mathcal{Z} \mid \exists x \in X \exists y \in Y \; u \cdot y = x\}.$$

For integer a and $op \in \{+, -, \cdot, /\}$ identify
$a \; op \; X$ with $\{a\} \; op \; X$,
$X \; op \; a$ with $X \; op \; \{a\}$.

Interval Arithmetic, ctd

Note 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

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

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

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

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

$$[-3..5]/[-1..2] = \mathcal{Z}.$$

Turning Sets to Intervals

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

Examples:

$int([3..3] \cdot [1..2]) = [3..6]$,

$int([3..5]/[-1..2]) = [-5..5]$,

$int([-3..5]/[-1..2]) = \mathcal{Z}$.

Rule for Linear Equality

Intuition:

$$\sum_{i=1}^{n} a_i x_i = b$$

implies that for $j \in [1..n]$

$$x_j = \frac{b - \Sigma_{i \in [1..n] - \{j\}} a_i x_i}{a_j}$$

$$\textit{LINEAR EQUALITY}$$

$$\frac{\langle \Sigma_{i=1}^{n} a_i x_i = b \; ; \; x_1 \in D_1, \ldots, x_n \in D_n \rangle}{\langle \Sigma_{i=1}^{n} a_i x_i = b \; ; \; \ldots, x_j \in D'_j, \ldots \rangle}$$

where $j \in [1..n]$, and

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

Multiplication Rules

MULTIPLICATION 1

$$\frac{\langle x \cdot y = z \ ; \ x \in D_x, y \in D_y, z \in D_z \rangle}{\langle x \cdot y = z \ ; \ x \in D_x, y \in D_y, z \in D_z \cap int(D_x \cdot D_y) \rangle}$$

MULTIPLICATION 2

$$\frac{\langle x \cdot y = z \ ; \ x \in D_x, y \in D_y, z \in D_z \rangle}{\langle x \cdot y = z \ ; \ x \in D_x \cap int(D_z / D_y), y \in D_y, z \in D_z \rangle}$$

MULTIPLICATION 3

$$\frac{\langle x \cdot y = z \ ; \ x \in D_x, y \in D_y, z \in D_z \rangle}{\langle x \cdot y = z \ ; \ x \in D_x, y \in D_y \cap int(D_z / D_x), z \in D_z \rangle}$$

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 it 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:

$$\frac{x \in [a..b]}{x \in [a..\lfloor\frac{a+b}{2}\rfloor] \mid x \in [\lfloor\frac{a+b}{2}\rfloor + 1..b]}$$

  where $a < b$.
- Combine it with the following heuristic:

  choose the variable with the smallest interval domain.

Proceed by cases: branch and bound.

More on Interval Arithmetic

Given objective function *obj*.
obj^+: extension of *obj* to function from sets of integers to sets of integers.

Defined by induction using interval arithmetic.

Example Suppose

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

Then

$$obj^+(X, Y) = X \cdot X \cdot Y - 3 \cdot X \cdot Y \cdot Y + 5.$$

Lemma Given:

— *obj*: arithmetic expression,

— $X_1, \ldots, X_n$ integer intervals.

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

Heuristic Function

Take

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

Define

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

Thanks to Lemma h satisfies conditions **A** and **B** for the heuristic function.

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.