Solving Numeric Constraints

Michel RUEHER

CeP Project I3S–CNRS

Université de Nice–Sophia Antipolis

November 2006

Overview

- ## Interval Programming
  - Interval arithmetic
  - Interval Analysis methods

Overview

- ## Interval Programming
  - Interval arithmetic
  - Interval Analysis methods

- ## Constraint Programming
  - Overall scheme
  - Local consistencies
  - Quantified constraints
  - Global Constraints

1. Interval Programming

- $\longrightarrow$ Basics on interval arithmetics
- $\longrightarrow$ Interval Newton-like methods for solving a multi-variate system of non-linear equations

1.1 Interval arithmetics : notations

- $\mathbf{c_j(x_1, \ldots, x_n)}$: a relation over the reals; $\mathcal{C}$: the set of constraints

1.1 Interval arithmetics : notations

- $\mathbf{c_j}(\mathbf{x_1}, \ldots, \mathbf{x_n})$: a relation over the reals; $\mathcal{C}$: the set of constraints
- $\mathbf{X}$ or $\mathbf{D}_x$: the domain of variable $\mathbf{x}$; $\mathcal{D}$: the set of domains of all the variables

1.1 Interval arithmetics : notations

◇ $\mathbf{c_j(x_1, \ldots, x_n)}$: a relation over the reals; $\mathcal{C}$: the set of constraints

◇ $\mathbf{X}$ or $\mathbf{D}_x$: the domain of variable $\mathbf{x}$; $\mathcal{D}$: the set of domains of all the variables

◇ $\mathbb{R}$: the set of real numbers; $\mathbb{F}$: the set of floating point numbers $\mathbf{a}^+$ (resp. $\mathbf{a}^-$) : the smallest (resp. largest) number of $\mathbb{F}$ strictly greater (resp. lower) than $\mathbf{a}$

1.1 Interval arithmetics : notations

- $\mathbf{c_j(x_1,\ldots,x_n)}$: a relation over the reals; $\mathcal{C}$: the set of constraints
- $\mathbf{X}$ or $\mathbf{D}_x$: the domain of variable $\mathbf{x}$; $\mathcal{D}$: the set of domains of all the variables
- $I\!R$: the set of real numbers; $I\!F$: the set of floating point numbers $\mathbf{a}^+$ (resp. $\mathbf{a}^-$) : the smallest (resp. largest) number of $I\!F$ strictly greater (resp. lower) than $\mathbf{a}$
- $\mathbf{X} = [\underline{\mathbf{X}}, \overline{\mathbf{X}}]$ is the set of real numbers $\mathbf{x}$ verifying $\underline{\mathbf{X}} \leq \mathbf{x} \leq \overline{\mathbf{X}}$

1.1 Interval arithmetics : notations

- $\diamond$ $\mathbf{c_j(x_1, \ldots, x_n)}$: a relation over the reals; $\mathcal{C}$: the set of constraints
- $\diamond$ $\mathbf{X}$ or $\mathbf{D}_x$: the domain of variable $\mathbf{x}$; $\mathcal{D}$: the set of domains of all the variables
- $\diamond$ $I\!R$: the set of real numbers; $I\!F$: the set of floating point numbers
  $\mathbf{a}^+$ (resp. $\mathbf{a}^-$) : the smallest (resp. largest) number of $I\!F$ strictly greater (resp. lower) than $\mathbf{a}$
- $\diamond$ $\mathbf{X} = [\underline{\mathbf{X}}, \overline{\mathbf{X}}]$ is the set of real numbers $\mathbf{x}$ verifying $\underline{\mathbf{X}} \leq \mathbf{x} \leq \overline{\mathbf{X}}$
- $\diamond$ $\mathbf{x}, \mathbf{y}$: real variables or vectors; $\mathbf{X}, \mathbf{Y}$ interval variables or vectors

1.1 Interval arithmetics : basic definitions (1)

Interval arithmetic (Moore-1966) is based on the representation of variables as intervals

1.1 Interval arithmetics : basic definitions (1)

Interval arithmetic (Moore-1966) is based on the representation of variables as intervals

Let f be a real-valued function of n unknowns $\mathbf{X} = \{x_1, \ldots, x_n\}$, an **interval evaluation $\mathbf{F}$** of f for given ranges $\{X_1, \ldots, X_n\}$ for the unknowns is an interval Y such that

$$\forall \mathbf{X} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\} \in \mathcal{X} = \{\mathbf{X_1}, \ldots, \mathbf{X_n}\} \qquad \underline{\mathbf{Y}} \leq \mathbf{f}(\mathbf{X}) \leq \overline{\mathbf{Y}}$$

$\underline{\mathbf{Y}}, \overline{\mathbf{Y}}$ are lower and upper bounds for the values of f when the values of the unknowns are restricted to the box $\mathcal{X}$

1.1 Interval arithmetics : basic definitions (2)

A relation over the intervals $C : \mathcal{I}^n \longrightarrow \mathcal{B}ool$ is an interval extension of the relation $c : \mathcal{R}^n \longrightarrow \mathcal{B}ool$ iff:

$$\forall \mathbf{I_1}, \ldots, \mathbf{I_n} \in \mathcal{I} : \mathbf{r_1} \in \mathbf{I_1}, \ldots, \mathbf{r_n} \in \mathbf{I_n} \;\&\; \mathbf{c}(\mathbf{r_1}, \ldots, \mathbf{r_n}) \Rightarrow \mathbf{C}(\mathbf{I_1}, \ldots, \mathbf{I_n})$$

1.1 Interval arithmetics : basic definitions (2)

A relation over the intervals $C : \mathcal{I}^n \longrightarrow \mathcal{B}ool$ is an interval extension of the relation $c : \mathcal{R}^n \longrightarrow \mathcal{B}ool$ iff:
$\forall \mathbf{I_1}, \ldots, \mathbf{I_n} \in \mathcal{I} : \mathbf{r_1} \in \mathbf{I_1}, \ldots, \mathbf{r_n} \in \mathbf{I_n} \;\&\; \mathbf{c}(\mathbf{r_1}, \ldots, \mathbf{r_n}) \Rightarrow \mathbf{C}(\mathbf{I_1}, \ldots, \mathbf{I_n})$

For instance, $\mathbf{I_1} \doteq \mathbf{I_2} \Leftrightarrow (\mathbf{I_1} \cap \mathbf{I_2}) \neq \emptyset$ is an interval extension of the relation $=$ over the real numbers

Interval arithmetics : natural interval extension (1)

$\longrightarrow$ In general, it is not possible to compute **the exact enclosure** of the range for an arbitrary function over the real number

Interval arithmetics : natural interval extension (1)

→ In general, it is not possible to compute **the exact enclosure** of the range for an arbitrary function over the real number

→ The interval extension of a function is an interval function that computes an **outer approximation** of the range of the function over a domain

Interval arithmetics : natural interval extension (2)

The natural interval extension of a real function f is defined by replacing all the mathematical operators in f by their interval equivalents to obtain $\mathbf{F}$

Interval arithmetics : natural interval extension (2)

The natural interval extension of a real function f is defined by replacing all the mathematical operators in f by their interval equivalents to obtain $\mathbf{F}$

- $[\mathbf{a}, \mathbf{b}] \ominus [\mathbf{c}, \mathbf{d}] = [\mathbf{a} - \mathbf{d}, \mathbf{b} - \mathbf{c}]$

Interval arithmetics : natural interval extension (2)

The natural interval extension of a real function f is defined by replacing all the mathematical operators in f by their interval equivalents to obtain $\mathbf{F}$

- $[\mathbf{a}, \mathbf{b}] \ominus [\mathbf{c}, \mathbf{d}] = [\mathbf{a} - \mathbf{d}, \mathbf{b} - \mathbf{c}]$
- $[\mathbf{a}, \mathbf{b}] \oplus [\mathbf{c}, \mathbf{d}] = [\mathbf{a} + \mathbf{c}, \mathbf{b} + \mathbf{d}]$

Interval arithmetics : natural interval extension (2)

The natural interval extension of a real function f is defined by replacing all the mathematical operators in f by their interval equivalents to obtain $\mathbf{F}$

- $[\mathbf{a}, \mathbf{b}] \ominus [\mathbf{c}, \mathbf{d}] = [\mathbf{a} - \mathbf{d}, \mathbf{b} - \mathbf{c}]$
- $[\mathbf{a}, \mathbf{b}] \oplus [\mathbf{c}, \mathbf{d}] = [\mathbf{a} + \mathbf{c}, \mathbf{b} + \mathbf{d}]$
- $[\mathbf{a}, \mathbf{b}] \otimes [\mathbf{c}, \mathbf{d}] = [\mathrm{min}(\mathbf{ac}, \mathbf{ad}, \mathbf{bc}, \mathbf{bd}), \mathrm{max}(\mathbf{ac}, \mathbf{ad}, \mathbf{bc}, \mathbf{bd})]$

Interval arithmetics : natural interval extension (2)

The natural interval extension of a real function f is defined by replacing all the mathematical operators in f by their interval equivalents to obtain $\mathbf{F}$

- $[\mathbf{a},\mathbf{b}] \ominus [\mathbf{c},\mathbf{d}] = [\mathbf{a}-\mathbf{d},\mathbf{b}-\mathbf{c}]$
- $[\mathbf{a},\mathbf{b}] \oplus [\mathbf{c},\mathbf{d}] = [\mathbf{a}+\mathbf{c},\mathbf{b}+\mathbf{d}]$
- $[\mathbf{a},\mathbf{b}] \otimes [\mathbf{c},\mathbf{d}] = [\min(\mathbf{ac},\mathbf{ad},\mathbf{bc},\mathbf{bd}),\max(\mathbf{ac},\mathbf{ad},\mathbf{bc},\mathbf{bd})]$
- $[\mathbf{a},\mathbf{b}] \oslash [\mathbf{c},\mathbf{d}] = [\min(\frac{\mathbf{a}}{\mathbf{c}},\frac{\mathbf{a}}{\mathbf{d}},\frac{\mathbf{b}}{\mathbf{c}},\frac{\mathbf{b}}{\mathbf{d}}),\max(\frac{\mathbf{a}}{\mathbf{c}},\frac{\mathbf{a}}{\mathbf{d}},\frac{\mathbf{b}}{\mathbf{c}},\frac{\mathbf{b}}{\mathbf{d}})]$ ***if*** $\mathbf{0} \notin [\mathbf{c},\mathbf{d}]$

Interval extension : properties

$\longrightarrow$ If $0 \notin F(\mathcal{X})$, then no value exists in the box $\mathcal{X}$ such that $f(\mathbf{X}) = 0$
$\Leftrightarrow$ **the equation $f(\mathbf{X})$ has no root in the box $\mathcal{X}$**

Interval extension : properties

$\longrightarrow$ If $0 \notin F(\mathcal{X})$, then no value exists in the box $\mathcal{X}$ such that $f(\mathbf{X}) = 0$ $\Leftrightarrow$ **the equation $f(\mathbf{X})$ has no root in the box $\mathcal{X}$**

$\longrightarrow$ Interval arithmetics can be implemented taking into account round-off errors

Interval extension : properties

$\longrightarrow$ If $0 \notin F(\mathcal{X})$, then no value exists in the box $\mathcal{X}$ such that $f(\mathbf{X}) = 0$
$\Leftrightarrow$ **the equation $f(\mathbf{X})$ has no root in the box $\mathcal{X}$**

$\longrightarrow$ Interval arithmetics can be implemented taking into account round-off errors

$\longrightarrow$ Interval arithmetic preserves **inclusion monotonicity**:

$$\mathbf{Y} \subseteq \mathbf{X} \quad \Rightarrow \quad \mathbf{F}(\mathbf{Y}) \subseteq \mathbf{F}(\mathbf{X})$$

Interval extension : properties

→ If $0 \notin F(\mathcal{X})$, then no value exists in the box $\mathcal{X}$ such that $f(\mathbf{X}) = 0$
$\Leftrightarrow$ **the equation $f(\mathbf{X})$ has no root in the box $\mathcal{X}$**

→ Interval arithmetics can be implemented taking into account round-off errors

→ Interval arithmetic preserves **inclusion monotonicity**:

$$\mathbf{Y} \subseteq \mathbf{X} \quad \Rightarrow \quad \mathbf{F}(\mathbf{Y}) \subseteq \mathbf{F}(\mathbf{X})$$

but interval arithmetics is **sub-distributive**:

$$\mathbf{X}(\mathbf{Y} + \mathbf{X}) \subseteq \mathbf{XY} + \mathbf{XZ}$$

Interval extension : limitations

→ The **wrapping effect**, which overestimates by a unique vector the image of an interval vector (which is in general not a vector)

Interval extension : limitations

→ The **wrapping effect**, which overestimates by a unique vector the image of an interval vector (which is in general not a vector)

→ The **dependency problem**, which is due to the independence the different occurences of some variable during the interval evaluation of an expression

Interval extension : limitations

→ The **wrapping effect**, which overestimates by a unique vector the image of an interval vector (which is in general not a vector)

→ The **dependency problem**, which is due to the independence the different occurences of some variable during the interval evaluation of an expression

Example :

Consider $X = [0, 5]$

$\mathbf{X} - \mathbf{X} = [\mathbf{0} - \mathbf{5}, \mathbf{5} - \mathbf{0}] = [-\mathbf{5}, \mathbf{5}]$ instead of $[0, 0]$!

Interval extension : limitations

→ The **wrapping effect**, which overestimates by a unique vector the image of an interval vector (which is in general not a vector)

→ The **dependency problem**, which is due to the independence the different occurences of some variable during the interval evaluation of an expression

Example :

Consider $X = [0, 5]$

$\mathbf{X - X = [0 - 5, 5 - 0] = [-5, 5]}$ instead of $[0, 0]$!

$\mathbf{X^2 - X = [0, 25] - [0, 5] = [-5, 25]}$

Interval extension : limitations

→ The **wrapping effect**, which overestimates by a unique vector the image of an interval vector (which is in general not a vector)

→ The **dependency problem**, which is due to the independence the different occurences of some variable during the interval evaluation of an expression

Example :

Consider $X = [0, 5]$

$\mathbf{X - X = [0 - 5, 5 - 0] = [-5, 5]}$ instead of $[0, 0]$!

$\mathbf{X^2 - X = [0, 25] - [0, 5] = [-5, 25]}$

$\mathbf{X(X - 1) = [0, 5]([0, 5] - [1, 1]) = [0, 5][-1, 4] = [-5, 20]}$

Interval extension : using different literal forms (1)

$\longrightarrow$ Factorized form (Horner for polynomial system) or distributed form

Interval extension : using different literal forms (1)

$\longrightarrow$ Factorized form (Horner for polynomial system) or distributed form

$\longrightarrow$ First-order Taylor development of $\mathbf{f}$

$$\mathbf{F}_{\mathsf{tay}}(\mathbf{X}) = \mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{X}).(\mathbf{X} - \mathbf{x})$$

with $\forall \mathbf{x} \in \mathbf{X}$, $\mathbf{J}()$ being the Jacobian of $\mathbf{f}$

Interval extension : using different literal forms (2)

$\longrightarrow$ In general, first order Taylor extensions yield a better enclosure than the natural extension on small intervals

Interval extension : using different literal forms (2)

$\longrightarrow$ In general, first order Taylor extensions yield a better enclosure than the natural extension on small intervals

$\longrightarrow$ Taylor extensions have a quadratic convergence whereas the natural extension has a linear convergence

Interval extension : using different literal forms (2)

$\longrightarrow$ In general, first order Taylor extensions yield a better enclosure than the natural extension on small intervals

$\longrightarrow$ Taylor extensions have a quadratic convergence whereas the natural extension has a linear convergence

$\longrightarrow$ In general, neither $\mathbf{F}_{\mathsf{nat}}$ nor $\mathbf{F}_{\mathsf{tay}}$ won't allow to compute the exact range of a function $\mathbf{f}$

Interval extension : using different literal forms (2)

→ In general, first order Taylor extensions yield a better enclosure than the natural extension on small intervals

→ Taylor extensions have a quadratic convergence whereas the natural extension has a linear convergence

→ In general, neither $\mathbf{F}_{\mathsf{nat}}$ nor $\mathbf{F}_{\mathsf{tay}}$ won't allow to compute the exact range of a function $\mathbf{f}$

Interval extension : using different literal forms (3)

Consider $f(x) = 1 - x + x^2$, and $\mathbf{X} = [0, 2]$

Interval extension : using different literal forms (3)

Consider $f(x) = 1 - x + x^2$, and $\mathbf{X} = [0, 2]$

$$\begin{aligned} \mathbf{f}_{\mathsf{tay}}([\mathbf{0}, \mathbf{2}]) &= \mathbf{f}(\mathbf{x}) + (\mathbf{2X} - \mathbf{1})(\mathbf{X} - \mathbf{x}) \\ &= \mathbf{f}(\mathbf{1}) + (\mathbf{2}[\mathbf{0}, \mathbf{2}] - \mathbf{1})([\mathbf{0}, \mathbf{2}] - \mathbf{1}) = [-\mathbf{2}, \mathbf{4}] \end{aligned}$$

Interval extension : using different literal forms (3)

Consider $f(x) = 1 - x + x^2$, and $\mathbf{X} = [0, 2]$

$$\mathbf{f}_{\mathsf{tay}}([\mathbf{0}, \mathbf{2}]) = \mathbf{f}(\mathbf{x}) + (\mathbf{2X} - \mathbf{1})(\mathbf{X} - \mathbf{x})$$
$$= \mathbf{f}(\mathbf{1}) + (\mathbf{2}[\mathbf{0}, \mathbf{2}] - \mathbf{1})([\mathbf{0}, \mathbf{2}] - \mathbf{1}) = [-\mathbf{2}, \mathbf{4}]$$

$$\mathbf{f}([\mathbf{0}, \mathbf{2}]) = \mathbf{1} - \mathbf{X} + \mathbf{X}^{\mathbf{2}} = \mathbf{1} - [\mathbf{0}, \mathbf{2}] + [\mathbf{0}, \mathbf{2}]^{\mathbf{2}} = [-\mathbf{1}, \mathbf{5}]$$

Interval extension : using different literal forms (3)

Consider $f(x) = 1 - x + x^2$, and $\mathbf{X} = [0, 2]$

$$\begin{aligned}\mathbf{f}_{\mathsf{tay}}([\mathbf{0}, \mathbf{2}]) &= \mathbf{f}(\mathbf{x}) + (\mathbf{2X} - \mathbf{1})(\mathbf{X} - \mathbf{x}) \\ &= \mathbf{f}(\mathbf{1}) + (\mathbf{2}[\mathbf{0}, \mathbf{2}] - \mathbf{1})([\mathbf{0}, \mathbf{2}] - \mathbf{1}) = [-\mathbf{2}, \mathbf{4}]\end{aligned}$$

$$\mathbf{f}([\mathbf{0}, \mathbf{2}]) = \mathbf{1} - \mathbf{X} + \mathbf{X}^2 = \mathbf{1} - [\mathbf{0}, \mathbf{2}] + [\mathbf{0}, \mathbf{2}]^2 = [-\mathbf{1}, \mathbf{5}]$$

$$\mathbf{f}_{\mathsf{factor}}([\mathbf{0}, \mathbf{2}]) = \mathbf{1} + \mathbf{X}(\mathbf{X} - \mathbf{1}) = \mathbf{1} + [\mathbf{0}, \mathbf{2}]([\mathbf{0}, \mathbf{2}] - \mathbf{1}) = [-\mathbf{1}, \mathbf{3}]$$

Interval extension : using different literal forms (3)

Consider $f(x) = 1 - x + x^2$, and $\mathbf{X} = [0, 2]$

$$\mathbf{f}_{\mathsf{tay}}([\mathbf{0}, \mathbf{2}]) = \mathbf{f(x) + (2X - 1)(X - x)}$$
$$= \mathbf{f(1) + (2[0, 2] - 1)([0, 2] - 1)} = [\mathbf{-2}, \mathbf{4}]$$

$$\mathbf{f}([\mathbf{0}, \mathbf{2}]) = \mathbf{1 - X + X^2 = 1 - [0, 2] + [0, 2]^2} = [\mathbf{-1}, \mathbf{5}]$$

$$\mathbf{f}_{\mathsf{factor}}([\mathbf{0}, \mathbf{2}]) = \mathbf{1 + X(X - 1) = 1 + [0, 2]([0, 2] - 1)} = [\mathbf{-1}, \mathbf{3}]$$

whereas the range of f over $\mathbf{X} = [\mathbf{0}, \mathbf{2}]$ is $[\mathbf{3/4}, \mathbf{3}]$

1.2 Interval Analysis methods

Goal : to determine the zeros of a system of n equations $f_i(x_1,\ldots,x_n)$ in n unknowns x_i inside the interval vector $X=\{X_1,\ldots,X_n\}$ with $x_i \in X_i$ for $i=1,\ldots,n$

1.2 Interval Analysis methods

Goal : to determine the zeros of a system of n equations $f_i(x_1, \ldots, x_n)$ in n unknowns x_i inside the interval vector $X = \{X_1, \ldots, X_n\}$ with $x_i \in X_i$ for $i = 1, \ldots, n$

$\longrightarrow$ **Gauss-Seidel iterative method**

1.2 Interval Analysis methods

Goal : to determine the zeros of a system of n equations $f_i(x_1, \ldots, x_n)$ in n unknowns x_i inside the interval vector $X = \{X_1, \ldots, X_n\}$ with $x_i \in X_i$ for $i = 1, \ldots, n$

$\longrightarrow$ **Gauss-Seidel iterative method**

$\longrightarrow$ **Interval Newton algorithm**

Gauss-Seidel iterative method

Consider the case of interval linear equations :

$$\mathbf{A}.\mathbf{x} = \mathbf{b}$$

with $\mathbf{A}$ an interval matrix and $\mathbf{b}$ an interval vector

Gauss-Seidel iterative method

Consider the case of interval linear equations :

$$\mathbf{A.x = b}$$

with $\mathbf{A}$ an interval matrix and $\mathbf{b}$ an interval vector

For each unknowns X_i, the Gauss-Seidel algorithm is defined by the following iterative process:

$$\mathbf{X_i^{k+1} = (b_i - \sum_{j=1}^{i-1} A_{i,j} X_j^{k+1} - \sum_{j=i+1}^{n} A_{i,j} X_j^{k}) / A_{i,i} \cap X_i^k}$$

Pre-conditioning $\longrightarrow$ to shrink the width of the intervals

Interval Newton algorithm (1)

Principle of the Newton operator :

Consider $\mathbf{f} : \mathcal{R} \longrightarrow \mathcal{R}$, the mean value theorem says :

$\exists \mathbf{a} \in [\mathbf{v}, \mathbf{u}] : \mathbf{f}(\mathbf{u}) - \mathbf{f}(\mathbf{v}) = (\mathbf{u} - \mathbf{v})\mathbf{f}'(\mathbf{a})$ and thus,
$\mathbf{f} \ : \ \mathbf{v} = \mathbf{u} - \frac{\mathbf{f}(\mathbf{u})}{\mathbf{f}'(\mathbf{a})}$ if $\mathbf{v}$ is a zero of $\mathbf{f}$

If $\mathbf{a} \in \mathbf{I}$ then $\mathbf{f}(\mathbf{a}) \in \mathbf{F}(\mathbf{I})$, and $\mathbf{v} \in \tilde{\mathbf{u}} - \frac{\mathbf{F}(\tilde{\mathbf{u}})}{\mathbf{F}'(\mathbf{I})} = \mathbf{N}(\mathbf{F}, \mathbf{F}', \tilde{\mathbf{u}}, \mathbf{I})$

Interval Newton algorithm (1)

Principle of the Newton operator :

Consider $\mathbf{f} : \mathcal{R} \rightarrow \mathcal{R}$, the mean value theorem says :

$\exists \mathbf{a} \in [\mathbf{v}, \mathbf{u}] : \mathbf{f}(\mathbf{u}) - \mathbf{f}(\mathbf{v}) = (\mathbf{u} - \mathbf{v})\mathbf{f}'(\mathbf{a})$ and thus,

$\mathbf{f} \; : \; \mathbf{v} = \mathbf{u} - \frac{\mathbf{f}(\mathbf{u})}{\mathbf{f}'(\mathbf{a})}$ if $\mathbf{v}$ is a zero of $\mathbf{f}$

If $\mathbf{a} \in \mathbf{I}$ then $\mathbf{f}(\mathbf{a}) \in \mathbf{F}(\mathbf{I})$, and $\mathbf{v} \in \tilde{\mathbf{u}} - \frac{\mathbf{F}(\tilde{\mathbf{u}})}{\mathbf{F}'(\mathbf{I})} = \mathbf{N}(\mathbf{F}, \mathbf{F}', \tilde{\mathbf{u}}, \mathbf{I})$

If $\mathbf{v}$ is a zero of $\mathbf{f}$ then $\mathbf{v} \in \mathbf{I_n} \quad (\mathbf{n} \geq \mathbf{1})$ where

$$\mathbf{I_0} = \mathbf{I}$$
$$\mathbf{I_{i+1}} = \mathbf{N}(\mathbf{F}, \mathbf{F}', \mathbf{center}(\mathbf{I_i}), \mathbf{I}) \cap \mathbf{I_i}$$
$$\mathbf{I_n} = \mathbf{I_{n+1}}$$

Interval Newton algorithm (2)

The Interval Newton algorithm is used to solve **non-linear systems** with

$$\mathbf{X_{k+1} = N(\tilde{x}_k, X_k) \cap X_k} \quad \text{with} \quad \mathbf{N(\tilde{x}_k, X_k) = \tilde{x}_k - A.f(\tilde{x}_k)}$$

where $\mathbf{A = [F^{'}(X_k)]^{-1}}$ and $\mathbf{\tilde{x}_k \in X_k}$ (e.g., the mid-point of $\mathbf{X_k}$)

Interval Newton algorithm (2)

The Interval Newton algorithm is used to solve **non-linear systems** with

$$\mathbf{X_{k+1}} = \mathbf{N}(\tilde{\mathbf{x}}_\mathbf{k}, \mathbf{X_k}) \cap \mathbf{X_k} \quad \text{with} \quad \mathbf{N}(\tilde{\mathbf{x}}_\mathbf{k}, \mathbf{X_k}) = \tilde{\mathbf{x}}_\mathbf{k} - \mathbf{A}.\mathbf{f}(\tilde{\mathbf{x}}_\mathbf{k})$$

where $\mathbf{A} = [\mathbf{F}^{'}(\mathbf{X_k})]^{-1}$ and $\tilde{\mathbf{x}}_\mathbf{k} \in \mathbf{X_k}$ (e.g., the mid-point of $\mathbf{X_k}$)

Properties :

◇ If $\mathbf{N}(\tilde{\mathbf{x}}_\mathbf{k}, \mathbf{X_k}) \cap \mathbf{X_k} = \emptyset$, then the system $\mathbf{F}$ has no solution in $\mathbf{X_k}$

◇ if $\mathbf{N}(\tilde{\mathbf{x}}_\mathbf{k}, \mathbf{X_k})_\mathbf{k} \subset \mathbf{X_k}$, there is one or more solution in X_{k+1}

Interval Newton algorithm (2)

The Interval Newton algorithm is used to solve **non-linear systems** with

$$\mathbf{X_{k+1}} = \mathbf{N}(\mathbf{\tilde{x}_k}, \mathbf{X_k}) \cap \mathbf{X_k} \quad \text{with} \quad \mathbf{N}(\mathbf{\tilde{x}_k}, \mathbf{X_k}) = \mathbf{\tilde{x}_k} - \mathbf{A}.\mathbf{f}(\mathbf{\tilde{x}_k})$$

where $\mathbf{A} = [\mathbf{F}^{'}(\mathbf{X_k})]^{-1}$ and $\mathbf{\tilde{x}_k} \in \mathbf{X_k}$ (e.g., the mid-point of $\mathbf{X_k}$)

Properties :

◇ If $\mathbf{N}(\mathbf{\tilde{x}_k}, \mathbf{X_k}) \cap \mathbf{X_k} = \emptyset$, then the system $\mathbf{F}$ has no solution in $\mathbf{X_k}$
◇ if $\mathbf{N}(\mathbf{\tilde{x}_k}, \mathbf{X_k})_\mathbf{k} \subset \mathbf{X_k}$, there is one or more solution in X_{k+1}

Matrix $\mathbf{A} = [\mathbf{F}^{'}(\mathbf{X_k})]^{-1}$ may be costly to compute
… to determine $N(\tilde{x}_k, X_k) \rightarrow$ solve the linear system:

$$\mathbf{F}^{'}(\mathbf{X_k})(\mathbf{N}(\mathbf{\tilde{x}_k}, \mathbf{X_k}) - \mathbf{\tilde{x}_k}) = -\mathbf{f}(\mathbf{\tilde{x}_k})$$

2. Constraint Programming

Numeric CSP $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ **:**

2. Constraint Programming

Numeric CSP $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ **:**

◇ $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a set of variables

2. Constraint Programming

Numeric CSP $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ **:**

◇ $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a set of variables

◇ $\mathcal{D} = \{D_{x_1}, \ldots, D_{x_n}\}$ is a set of domains
(D_{x_i} contains all acceptable values for variable x_i)

2. Constraint Programming

Numeric CSP $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ **:**

◇ $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a set of variables

◇ $\mathcal{D} = \{D_{x_1}, \ldots, D_{x_n}\}$ is a set of domains
(D_{x_i} contains all acceptable values for variable x_i)

◇ $\mathcal{C} = \{c_1, \ldots, c_m\}$ is a set of constraints

2.1 Overall scheme

The constraint programming framework is based on a **branch & prune** schema which is best viewed as an iteration of two steps:

2.1 Overall scheme

The constraint programming framework is based on a **branch & prune** schema which is best viewed as an iteration of two steps:

1. **Pruning the search space**

2.1 Overall scheme

The constraint programming framework is based on a **branch & prune** schema which is best viewed as an iteration of two steps:

1. **Pruning the search space**

2. **Making a choice to generate two (or more) sub-problems**

2.1 Overall scheme

The constraint programming framework is based on a **branch & prune** schema which is best viewed as an iteration of two steps:

1. **Pruning the search space**

2. **Making a choice to generate two (or more) sub-problems**

◇ The pruning step $\longrightarrow$ **reduces an interval** when it can prove that the upper bound or the lower bound does not satisfy some constraint

2.1 Overall scheme

The constraint programming framework is based on a **branch & prune** schema which is best viewed as an iteration of two steps:

1. **Pruning the search space**

2. **Making a choice to generate two (or more) sub-problems**

◇ The pruning step $\longrightarrow$ **reduces an interval** when it can prove that the upper bound or the lower bound does not satisfy some constraint

◇ The branching step $\longrightarrow$ **splits the interval** associated to some variable in two intervals (often with the same width)

2.2 Local consistencies (1)

$\longrightarrow$ Informally speaking, a constraint system C satisfies a partial consistency property if **a relaxation of C is consistent**

2.2 Local consistencies (1)

$\longrightarrow$ Informally speaking, a constraint system C satisfies a partial consistency property if **a relaxation of C is consistent**

Consider $\mathbf{X} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}]$ and $\mathbf{C}(\mathbf{x}, \mathbf{x_1}, \ldots, \mathbf{x_n}) \in \mathcal{C}$: if $\mathbf{C}(\mathbf{x}, \mathbf{x_1}, \ldots, \mathbf{x_n})$ does not hold for any values $\mathbf{a} \in [\underline{\mathbf{x}}, \mathbf{x}']$, then X may be shrinked to $\mathbf{X} = [\mathbf{x}', \overline{\mathbf{x}}]$

2.2 Local consistencies (1)

→ Informally speaking, a constraint system C satisfies a partial consistency property if **a relaxation of C is consistent**

Consider $\mathbf{X} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}]$ and $\mathbf{C}(\mathbf{x}, \mathbf{x_1}, \ldots, \mathbf{x_n}) \in \mathcal{C}$: if $\mathbf{C}(\mathbf{x}, \mathbf{x_1}, \ldots, \mathbf{x_n})$ does not hold for any values $\mathbf{a} \in [\underline{\mathbf{x}}, \mathbf{x}']$, then X may be shrinked to $\mathbf{X} = [\mathbf{x}', \overline{\mathbf{x}}]$

→ A **constraint C_j is AC-like-consistent** if for any variable x_i in $\mathcal{X}_j$, **the bounds $\underline{D_i}$ and $\overline{D_i}$ have a support** in the domains of all other variables of $\mathcal{X}_j$

Local consistencies used in `BNR-prolog`, `Interlog`, `CLP(BNR)`, `PrologIV`, `UniCalc`, `Ilog Solver`, `Numerica`, `Icos`, `RealPaver` are AC-like-consistencies

2.2 Local consistencies (2)

Local consistencies are conditions that filtering algorithms must satisfy

2.2 Local consistencies (2)

Local consistencies are conditions that filtering algorithms must satisfy

$\longrightarrow$ fixed point algorithm defined by the sequence $\{\mathcal{D}_k\}$ of domains generated by the iterative application of an operator

$$\mathbf{Op} : \mathbb{I}(\mathbb{R})^{\mathbf{n}} \longrightarrow \mathbb{I}(\mathbb{R})^{\mathbf{n}}$$

$$\boxed{\mathcal{D}_k = \begin{cases} \mathcal{D} & \text{if } k = 0 \\ Op(\mathcal{D}_{k-1}) & \text{if } k > 0 \end{cases}}$$

2.2 Local consistencies (3)

Properties of the operator Op :

$\diamond$ $\mathbf{Op}(\mathcal{D}) \subseteq \mathcal{D}$ (contractance)

2.2 Local consistencies (3)

Properties of the operator Op :

- $\diamond$ $\mathbf{Op}(\mathcal{D}) \subseteq \mathcal{D}$ (contractance)
- $\diamond$ $\mathbf{O}p$ is conservative; that is, it cannot remove any solution

2.2 Local consistencies (3)

Properties of the operator Op :

- $\mathbf{Op}(\mathcal{D}) \subseteq \mathcal{D}$ (contractance)
- $\mathbf{O}p$ is conservative; that is, it cannot remove any solution
- $\mathcal{D}' \subseteq \mathcal{D} \Rightarrow \mathbf{Op}(\mathcal{D}') \subseteq \mathbf{Op}(\mathcal{D})$ (monotonicity)

2.2 Local consistencies (3)

Properties of the operator Op :

- $\diamond$ $\mathbf{Op}(\mathcal{D}) \subseteq \mathcal{D}$ (contractance)
- $\diamond$ $\mathbf{O}p$ is conservative; that is, it cannot remove any solution
- $\diamond$ $\mathcal{D}' \subseteq \mathcal{D} \Rightarrow \mathbf{Op}(\mathcal{D}') \subseteq \mathbf{Op}(\mathcal{D})$ (monotonicity)

The limit of the sequence $\{\mathcal{D}_k\}$, which corresponds to the greatest fixed point of the operator $\mathbf{O}p$

2.2 Local consistencies (4)

→ **2B–consistency** (also known as hull consistency) only requires to check the Arc–Consistency property **for each bound** of the intervals

2.2 Local consistencies (4)

$\longrightarrow$ **2B–consistency** (also known as hull consistency) only requires to check the Arc–Consistency property **for each bound** of the intervals

$\longrightarrow$ **Box–consistency** is a coarser relaxation of Arc–Consistency than 2B–consistency ... **but Box–consistency algorithms actually achieve a stronger filtering than 2B–consistency**

2.2 Local consistencies (4)

→ **2B–consistency** (also known as hull consistency) only requires to check the Arc–Consistency property **for each bound** of the intervals

→ **Box–consistency** is a coarser relaxation of Arc–Consistency than 2B–consistency ... **but Box–consistency algorithms actually achieve a stronger filtering than 2B–consistency**

→ **Strong consistencies** : no bound of the domains can be removed with a local consistency filtering algorithm

2B–consistency (1)

Variable x is 2B–consistency for constraint $\mathbf{f(x, x_1, \ldots, x_n) = 0}$ if the lower (resp. upper) bound of the domain $\mathbf{X}$ is the smallest (resp. largest) solution of $\mathbf{f(x, x_1, \ldots, x_n)}$

2B–CONSISTENCY (1)

Variable x is 2B–consistency for constraint $\mathbf{f(x, x_1, \ldots, x_n) = 0}$ if the lower (resp. upper) bound of the domain $\mathbf{X}$ is the smallest (resp. largest) solution of $\mathbf{f(x, x_1, \ldots, x_n)}$

Definition : 2B–consistency

Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a CSP and $\mathbf{C} \in \mathcal{C}$ a k-ary constraint over $(\mathbf{X_1, \ldots, X_k})$

$\mathbf{C}$ is 2B–consistency iff :

$$\forall \mathbf{i}, \mathbf{X_i} = \Box\{\mathbf{\tilde{x}_i} \mid \exists \mathbf{\tilde{x}_1} \in \mathbf{X_1}, \ldots, \exists \mathbf{\tilde{x}_{i-1}} \in \mathbf{X_{i-1}}, \exists \mathbf{\tilde{x}_{i+1}} \in \mathbf{X_{i+1}}, \ldots, \exists \mathbf{\tilde{x}_k} \in \mathbf{X_k} \text{ such that } \mathbf{c}(\mathbf{\tilde{x}_1}, \ldots, \mathbf{\tilde{x}_{i-1}}, \mathbf{\tilde{x}_i}, \mathbf{\tilde{x}_{i+1}} \ldots, \mathbf{\tilde{x}_k}) \text{ holds}\}$$

A CSP is 2B–consistency iff all its constraints are 2B–consistency

Box–consistency (1)

Variable $\mathbf{x}$ is Box–Consistent for constraint $\mathbf{f(x, x_1, \ldots, x_n) = 0}$ if the bounds of the domain of x correspond to the leftmost and the rightmost zero of the optimal interval extension of $\mathbf{f(x, x_1, \ldots, x_n)}$

Box–consistency (1)

Variable $\mathbf{x}$ is Box–Consistent for constraint $\mathbf{f(x, x_1, \ldots, x_n) = 0}$ if the bounds of the domain of x correspond to the leftmost and the rightmost zero of the optimal interval extension of $\mathbf{f(x, x_1, \ldots, x_n)}$

Definition : Box–consistency

Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$be a CSP and $\mathbf{C} \in \mathcal{C}$ a k-ary constraint over $\mathbf{(X_1, \ldots, X_k)}$

$\mathbf{C}$ is Box–Consistent if, for all $\mathbf{X_i}$ the following relations hold :

1. $\mathbf{C(X_1, \ldots, X_{i-1}, [\underline{X_i}, \underline{X_i}^{+}), X_{i+1}, \ldots, X_k)}$
2. $\mathbf{C(X_1, \ldots, X_{i-1}, (\overline{X_i}^{-}, \overline{X_i}], X_{i+1}, \ldots, X_k)}$

Local consistency filtering (1)

Algorithms that achieve a local consistency filtering are based upon projection functions

LOCAL CONSISTENCY FILTERING (1)

Algorithms that achieve a local consistency filtering are based upon projection functions

◇ **Solution functions** expresse the variable $\mathbf{x_i}$ in terms of the other variables of the constraint. The solution functions of $\mathbf{x + y = z}$ are: $\mathbf{f_x = z - y, f_y = z - x, f_z = x + y}$

Local consistency filtering (1)

Algorithms that achieve a local consistency filtering are based upon projection functions

- **Solution functions** expresse the variable $\mathbf{x_i}$ in terms of the other variables of the constraint. The solution functions of $\mathbf{x + y = z}$ are: $\mathbf{f_x = z - y, f_y = z - x, f_z = x + y}$

- An approximation of the projection of the constraint over $\mathbf{X_i}$ given a domain $\mathcal{D}$ can be computed with any interval extension of this solution function $\longrightarrow$ we have a way to compute $\pi_{\mathbf{j,i}}(\mathcal{D})$

LOCAL CONSISTENCY FILTERING (1)

Algorithms that achieve a local consistency filtering are based upon projection functions

- **Solution functions** expresse the variable $\mathbf{x_i}$ in terms of the other variables of the constraint. The solution functions of $\mathbf{x+y=z}$ are: $\mathbf{f_x = z - y, f_y = z - x, f_z = x + y}$

- An approximation of the projection of the constraint over $\mathbf{X_i}$ given a domain $\mathcal{D}$ can be computed with any interval extension of this solution function $\longrightarrow$ we have a way to compute $\pi_{\mathbf{j,i}}(\mathcal{D})$

- For complex constraints, no analytic solution function may exist Consider $\mathbf{x + \log(x) = 0}$

Local consistency filtering (2)

1. Analytic functions always exist when the variable to express in terms of the others appears only once in the constraint

Local consistency filtering (2)

1. Analytic functions always exist when the variable to express in terms of the others appears only once in the constraint
   $\longrightarrow$ **considers that each occurrence is a different new variable**

   For $\mathbf{x} + \mathbf{log}(\mathbf{x}) = \mathbf{0}$ we obtain $\mathbf{x_1} + \log(\mathbf{x_2}) = \mathbf{0}$
   Thus $\mathbf{f_{x_1}} = -\log(\mathbf{x_2})$, $\mathbf{f_{x_2}} = \exp^{-\mathbf{x_1}}$
   and $\pi_{\mathbf{x+log(x)=0,x}}(\mathbf{X}) = -\log(\mathbf{X}) \cap \exp^{-\mathbf{X}}$

LOCAL CONSISTENCY FILTERING (2)

1. Analytic functions always exist when the variable to express in terms of the others appears only once in the constraint
   $\longrightarrow$ **considers that each occurrence is a different new variable**

   For $\mathbf{x + \log(x) = 0}$ we obtain $\mathbf{x_1} + \log(\mathbf{x_2}) = \mathbf{0}$
   Thus $\mathbf{f_{x_1}} = -\log(\mathbf{x_2})$, $\mathbf{f_{x_2}} = \exp^{-\mathbf{x_1}}$
   and $\pi_{\mathbf{x+\log(x)=0,x}}(\mathbf{X}) = -\log(\mathbf{X}) \cap \exp^{-\mathbf{X}}$

   ◇ This approach is used for computing **2B–consistency filtering** (the initial constraints are decomposed into primitive constraints)

Local consistency filtering (2)

1. Analytic functions always exist when the variable to express in terms of the others appears only once in the constraint
   $\longrightarrow$ **considers that each occurrence is a different new variable**

   For $\mathbf{x} + \mathbf{log}(\mathbf{x}) = \mathbf{0}$ we obtain $\mathbf{x_1} + \log(\mathbf{x_2}) = \mathbf{0}$
   Thus $\mathbf{f_{x_1}} = -\log(\mathbf{x_2})$, $\mathbf{f_{x_2}} = \exp^{-\mathbf{x_1}}$
   and $\pi_{\mathbf{x+log(x)=0,x}}(\mathbf{X}) = -\log(\mathbf{X}) \cap \exp^{-\mathbf{X}}$

   ◇ This approach is used for computing **2B–consistency filtering** (the initial constraints are decomposed into primitive constraints)

   ◇ Decomposition does not change the semantics of the initial constraints system but **it amplifies the dependency problem**

Local consistency filtering (3)

2. Transformation of the constraint $\mathbf{C_j(x_{j_1}, \ldots x_{j_k})}$ into $\mathbf{k}$ mono-variable constraints $\mathbf{C_{j,l}, l = 1 \ldots k}$ by substituting their intervals for the other variables

Local consistency filtering (3)

2. Transformation of the constraint $\mathbf{C_j(x_{j_1}, \ldots x_{j_k})}$ into $\mathbf{k}$ mono-variable constraints $\mathbf{C_{j,l}, l = 1 \ldots k}$ by substituting their intervals for the other variables

→ The two extremal zeros of $\mathbf{C_{j,l}}$ can be found by a **dichotomy algorithm combined with a mono-variable version of the interval Newton method**

Local consistency filtering (3)

2. Transformation of the constraint $\mathbf{C_j(x_{j_1}, \ldots x_{j_k})}$ into $\mathbf{k}$ mono-variable constraints $\mathbf{C_{j,l}, l = 1 \ldots k}$ by substituting their intervals for the other variables

$\longrightarrow$ The two extremal zeros of $\mathbf{C_{j,l}}$ can be found by a **dichotomy algorithm combined with a mono-variable version of the interval Newton method**

$\longrightarrow$ This approach is **well adapted for Box–consistency filtering**

Local consistency filtering (4)

3. Use the Taylor extension to transform the constraint into an interval linear constraint. $f(X) = 0$ becomes

$$\mathbf{f(c)} + \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{nat}\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x_i}}\right)(\mathbf{X}) * (\mathbf{X_i} - \mathbf{c_i}) = \mathbf{0}$$

   where $\mathbf{c} = \mathbf{m}(\mathbf{X})$. The derivatives are evaluated over a box $\mathcal{D}$ that contains $\mathbf{X}$, $\mathcal{D}$ is considered as constant, and with $\mathbf{c} = \mathbf{m}(\mathcal{D})$

Local consistency filtering (4)

3. Use the Taylor extension to transform the constraint into an interval linear constraint. $f(X) = 0$ becomes

$$\mathbf{f(c) + \sum_{i=1}^{n} nat(\frac{\partial f}{\partial x_i})(X) * (X_i - c_i) = 0}$$

where $\mathbf{c} = \mathbf{m}(\mathbf{X})$. The derivatives are evaluated over a box $\mathcal{D}$ that contains $\mathbf{X}$, $\mathcal{D}$ is considered as constant, and with $\mathbf{c} = \mathbf{m}(\mathcal{D})$

$\longrightarrow$ The equation becomes an interval linear equation in X, which does not contain multiple occurrences

Local consistency filtering (4)

3. Use the Taylor extension to transform the constraint into an interval linear constraint. $f(X) = 0$ becomes

$$\mathbf{f(c) + \sum_{i=1}^{n} nat(\frac{\partial f}{\partial x_i})(X) * (X_i - c_i) = 0}$$

where $\mathbf{c = m(X)}$. The derivatives are evaluated over a box $\mathcal{D}$ that contains $\mathbf{X}$, $\mathcal{D}$ is considered as constant, and with $\mathbf{c = m(\mathcal{D})}$

$\longrightarrow$ The equation becomes an interval linear equation in X, which does not contain multiple occurrences

$\longrightarrow$ **Solving the squared interval linear system allows much more precise approximations of projections to be computed**

Stronger consistencies, 3B–Consistency (1)

3B–Consistency, a relaxation of path consistency, checks whether 2B–Consistency can be enforced when the domain of a variable is reduced to the value of one of its bounds in the whole system

Stronger consistencies, 3B–Consistency (2)

Definition : 3B–Consistency
Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a CSP and $\mathbf{x}$ a variable of $\mathcal{X}$ with $\mathbf{D_x} = [\mathbf{a}, \mathbf{b}]$. Let also:

◇ Let $\mathbf{P}_{\mathbf{D^1_x} \leftarrow [\mathbf{a}, \mathbf{a}^+)}$ be the CSP derived from $\mathbf{P}$ by substituting $\mathbf{D_x}$ in $\mathcal{D}$ with $\mathbf{D^1_x} = [\mathbf{a}, \mathbf{a}^+)$

Stronger consistencies, 3B–Consistency (2)

Definition : 3B–Consistency
Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a CSP and $\mathbf{x}$ a variable of $\mathcal{X}$ with $\mathbf{D_x} = [\mathbf{a}, \mathbf{b}]$. Let also:

◇ Let $\mathbf{P}_{\mathbf{D_x^1} \leftarrow [\mathbf{a}, \mathbf{a}^+)}$ be the CSP derived from $\mathbf{P}$ by substituting $\mathbf{D_x}$ in $\mathcal{D}$ with $\mathbf{D_x^1} = [\mathbf{a}, \mathbf{a}^+)$

◇ Let $\mathbf{P}_{\mathbf{D_x^2} \leftarrow (\mathbf{b}^-, \mathbf{b}]}$ lbe the CSP derived from $\mathbf{P}$ by substituting $\mathbf{D_x}$ in $\mathcal{D}$ with $\mathbf{D_x^2} = (\mathbf{b}^-, \mathbf{b}]$

Stronger consistencies, 3B–Consistency (2)

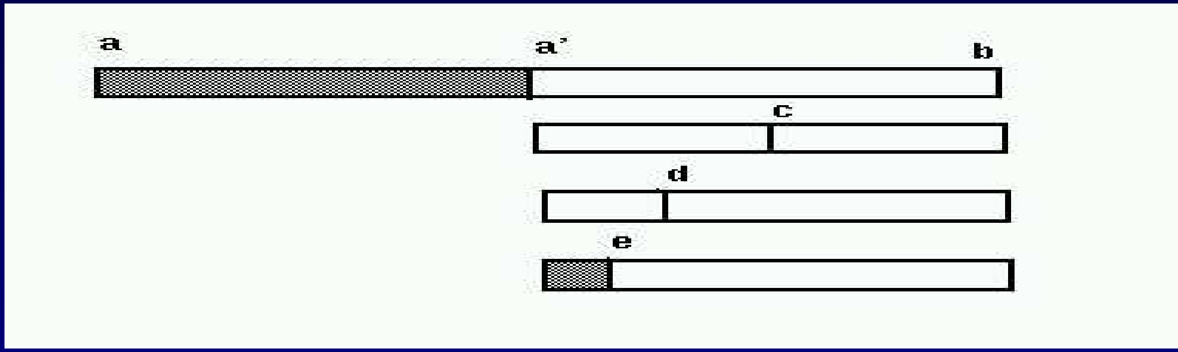
Definition : 3B–Consistency

Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a CSP and $\mathbf{x}$ a variable of $\mathcal{X}$ with $\mathbf{D_x} = [\mathbf{a}, \mathbf{b}]$. Let also:

- $\diamond$ Let $\mathbf{P}_{\mathbf{D_x^1} \leftarrow [\mathbf{a}, \mathbf{a}^+)}$ be the CSP derived from $\mathbf{P}$ by substituting $\mathbf{D_x}$ in $\mathcal{D}$ with $\mathbf{D_x^1} = [\mathbf{a}, \mathbf{a}^+)$
- $\diamond$ Let $\mathbf{P}_{\mathbf{D_x^2} \leftarrow (\mathbf{b}^-, \mathbf{b}]}$ lbe the CSP derived from $\mathbf{P}$ by substituting $\mathbf{D_x}$ in $\mathcal{D}$ with $\mathbf{D_x^2} = (\mathbf{b}^-, \mathbf{b}]$

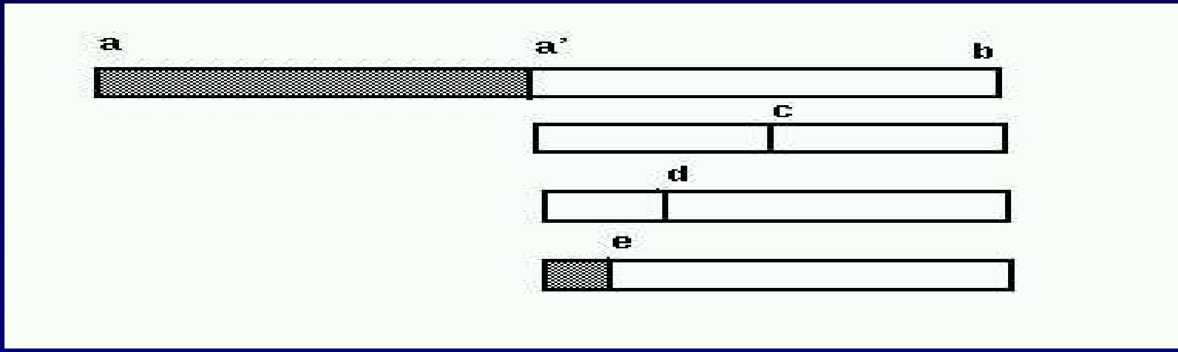
$\mathbf{X}$ is 3B–Consistent iff $\mathbf{\Phi_{2B}}(\mathbf{P}_{\mathbf{x} \leftarrow [\underline{\mathbf{x}}, \underline{\mathbf{x}}^+)}) \neq \mathbf{P}_\emptyset$ and
$\mathbf{\Phi_{2B}}(\mathbf{P}_{\mathbf{x} \leftarrow (\overline{\mathbf{x}}^-, \overline{\mathbf{x}}]}) \neq \mathbf{P}_\emptyset$

Stronger consistencies, 3B–Consistency (3)



Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a CSP and $\mathbf{D_x} = [\mathbf{a}, \mathbf{b}]$, if $\mathbf{\Phi_{2B}(P_{D_x \leftarrow [a, \frac{a+b}{2}]}) = \emptyset}$

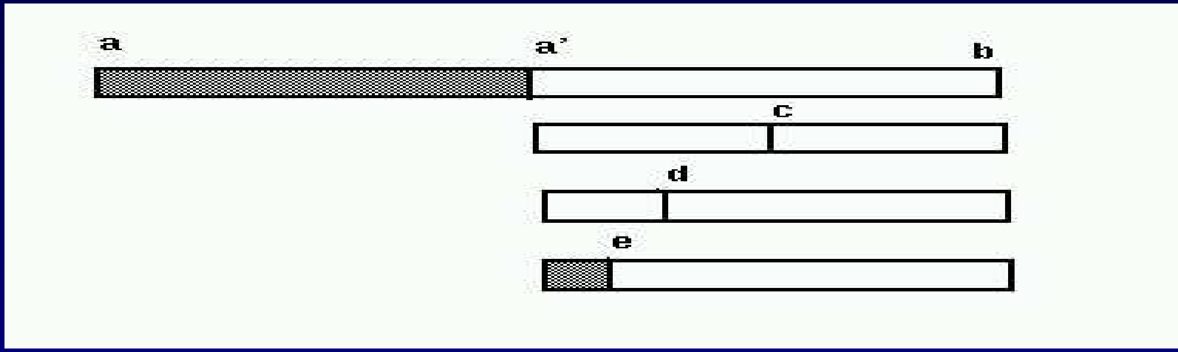
Stronger consistencies, 3B–Consistency (3)



Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a CSP and $\mathbf{D_x} = [\mathbf{a}, \mathbf{b}]$, if $\mathbf{\Phi_{2B}(P_{D_x \leftarrow [a, \frac{a+b}{2}]})} = \emptyset$

◇ then the part $[\mathbf{a}, \frac{\mathbf{a+b}}{\mathbf{2}})$ de $\mathbf{D_x}$ will be removed and the filtering process continues on the interval $[\frac{\mathbf{a+b}}{\mathbf{2}}, \mathbf{b}]$

Stronger consistencies, 3B–Consistency (3)

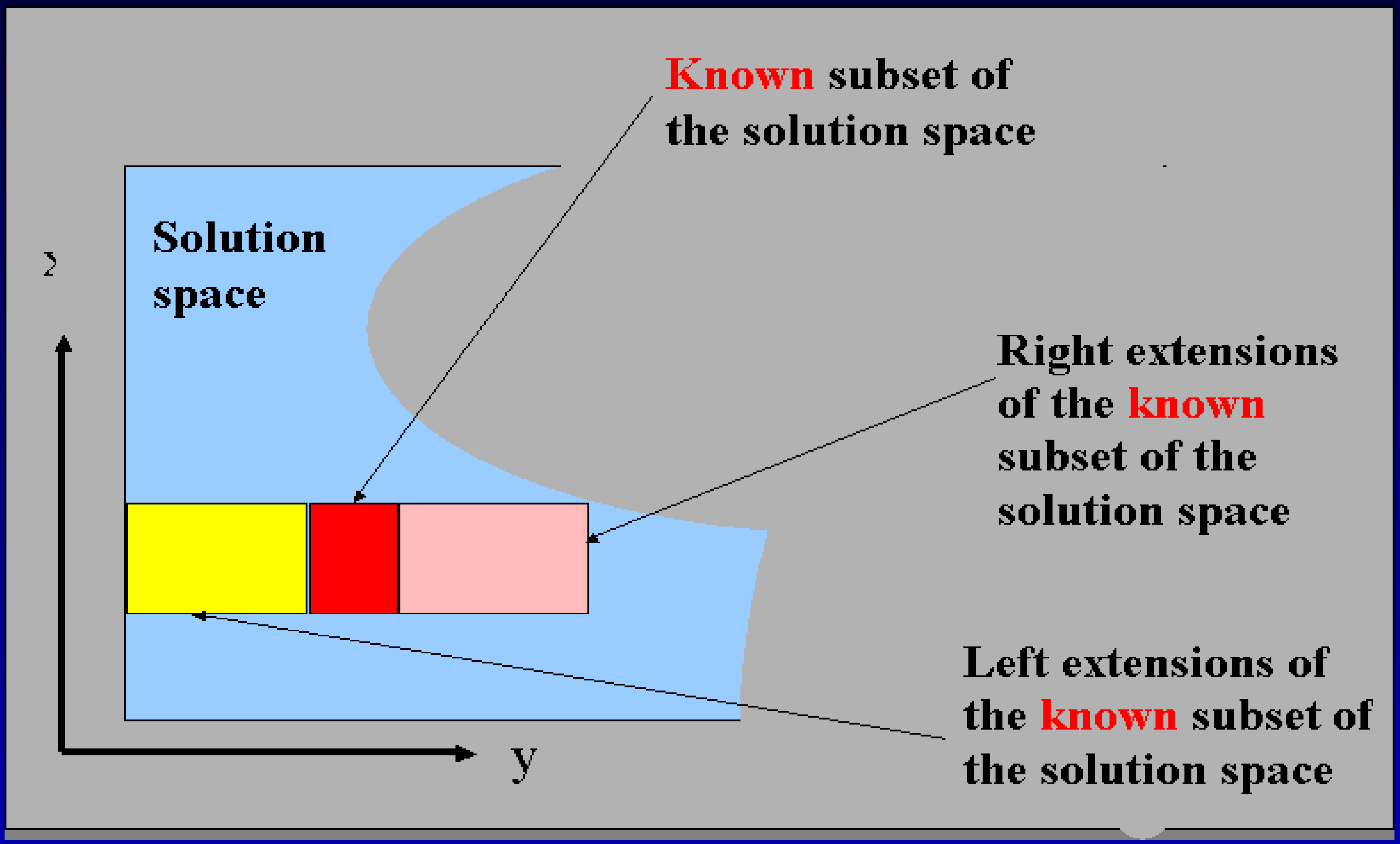


Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a CSP and $\mathbf{D_x} = [\mathbf{a}, \mathbf{b}]$, if $\mathbf{\Phi_{2B}}(\mathbf{P}_{\mathbf{D_x} \leftarrow [\mathbf{a}, \frac{\mathbf{a}+\mathbf{b}}{2}]}) = \emptyset$

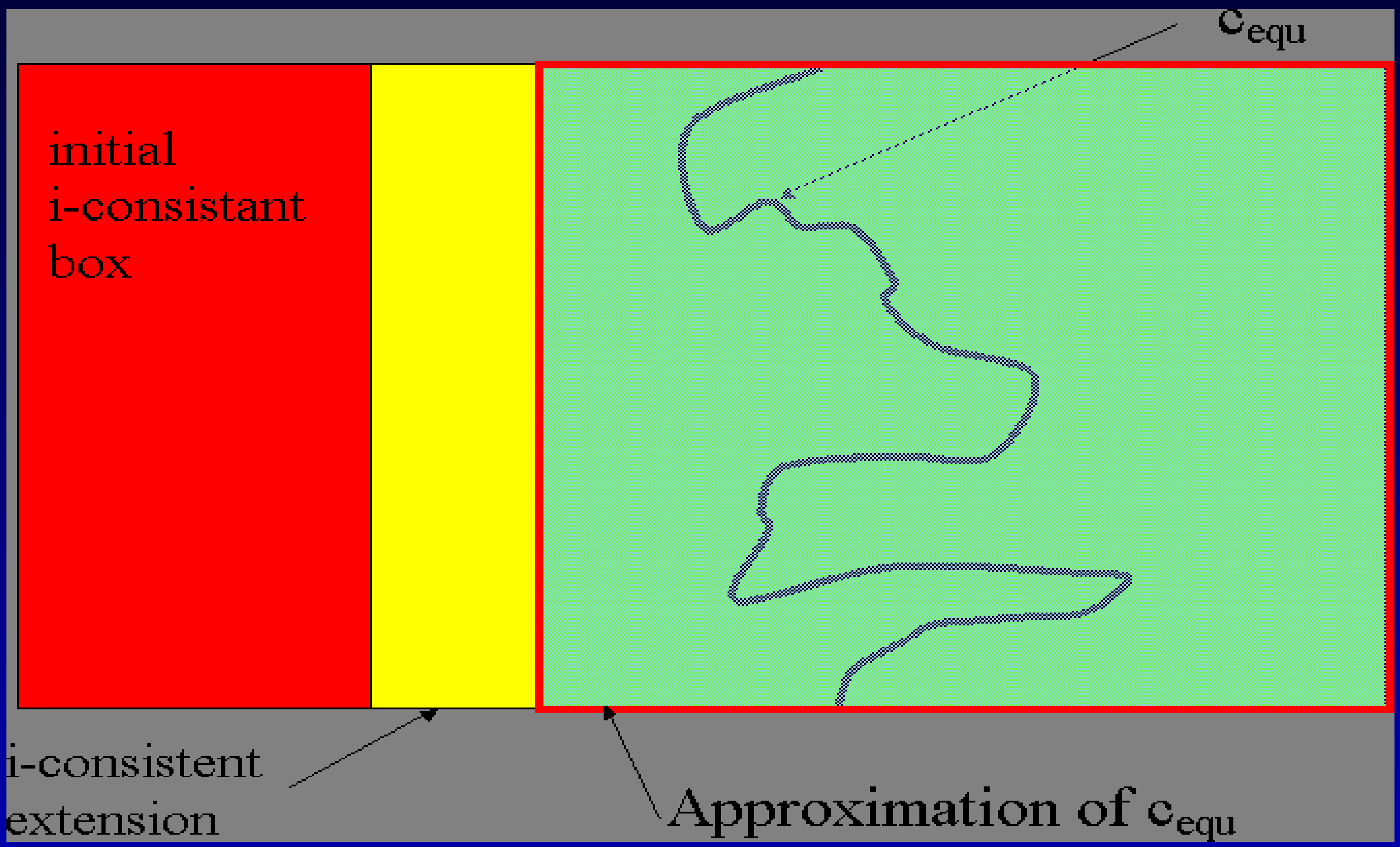
◇ then the part $[\mathbf{a}, \frac{\mathbf{a}+\mathbf{b}}{2})$ de $\mathbf{D_x}$ will be removed and the filtering process continues on the interval $[\frac{\mathbf{a}+\mathbf{b}}{2}, \mathbf{b}]$

◇ otherwise, the filtering process continues on the interval $[\mathbf{a}, \frac{3\mathbf{a}+\mathbf{b}}{4}]$.

2.3. Quantified constraints (1)



2.3. Quantified constraints (2)



2.3. Quantified constraints (3)

◇ $\forall \mathbf{x} \in \mathbf{D_x} : \ \mathbf{x} + \mathbf{x_1} \ < \mathbf{5} \quad \textbf{with} \quad \mathbf{D_x} = [-\mathbf{2}, \mathbf{2}], \mathbf{D_{x_1}} \ = [\mathbf{1}, \mathbf{5}]$

$$\neg(\forall \mathbf{x} \in \mathbf{D_x} : \ \mathbf{x} + \mathbf{x_1} \ < \mathbf{5}) - > \mathbf{x} + [\mathbf{1}, \mathbf{5}] \geq \mathbf{5} \ \Rightarrow \mathbf{x} \geq \mathbf{0}$$

$\mathbf{D_x} \in [-\mathbf{2}, \mathbf{0}) \Rightarrow \forall \mathbf{x} \in \mathbf{D_x} : \ \mathbf{x} + \mathbf{x_1} \ < \mathbf{5}$ holds

2.3. Quantified constraints (3)

◇ $\forall \mathbf{x} \in \mathbf{D_x} : \ \mathbf{x} + \mathbf{x_1} \ < \mathbf{5} \quad \textbf{with} \quad \mathbf{D_x} = [-\mathbf{2}, \mathbf{2}], \mathbf{D_{x_1}} \ = [\mathbf{1}, \mathbf{5}]$

$$\neg(\forall \mathbf{x} \in \mathbf{D_x} : \ \mathbf{x} + \mathbf{x_1} \ < \mathbf{5}) - > \mathbf{x} + [\mathbf{1}, \mathbf{5}] \geq \mathbf{5} \ \Rightarrow \mathbf{x} \geq \mathbf{0}$$

$\mathbf{D_x} \in [-\mathbf{2}, \mathbf{0}) \Rightarrow \forall \mathbf{x} \in \mathbf{D_x} : \ \mathbf{x} + \mathbf{x_1} \ < \mathbf{5}$ holds

◇ $\forall (\mathbf{x} \in \mathbf{X}) \quad \forall (\mathbf{y} \in \mathbf{Y}) \quad \exists (\mathbf{z} \in \mathbf{Z}) : \quad (\mathbf{z} = \mathbf{x} + \mathbf{y})$

2.3. Quantified constraints (3)

◇ $\forall \mathbf{x} \in \mathbf{D_x} : \ \mathbf{x} + \mathbf{x_1} \ < \mathbf{5} \quad \textbf{with} \quad \mathbf{D_x} = [-\mathbf{2}, \mathbf{2}], \mathbf{D_{x_1}} \ = [\mathbf{1}, \mathbf{5}]$

$$\neg(\forall \mathbf{x} \in \mathbf{D_x} : \ \mathbf{x} + \mathbf{x_1} \ < \mathbf{5}) - > \mathbf{x} + [\mathbf{1}, \mathbf{5}] \geq \mathbf{5} \ \Rightarrow \mathbf{x} \geq \mathbf{0}$$

$\mathbf{D_x} \in [-\mathbf{2}, \mathbf{0}) \Rightarrow \forall \mathbf{x} \in \mathbf{D_x} : \ \mathbf{x} + \mathbf{x_1} \ < \mathbf{5}$ holds

◇ $\forall(\mathbf{x} \in \mathbf{X}) \quad \forall(\mathbf{y} \in \mathbf{Y}) \quad \exists(\mathbf{z} \in \mathbf{Z}) : \quad (\mathbf{z} = \mathbf{x} + \mathbf{y})$

◇ Modal intervals

2.4 Global Constraints (1)

◇ **"Syntactical" approach**

To handle an approximation of the whole constraint system with the **simplex algorithm**

$\longrightarrow$ replace each non linear term by a new variable

2.4 Global Constraints (1)

◇ **"Syntactical" approach**

To handle an approximation of the whole constraint system with the **simplex algorithm**

$\longrightarrow$ replace each non linear term by a new variable

$\longrightarrow$ introduce redundant linear constraints to get a **tight approximation** of the non-linear terms

2.4 Global Constraints (1)

◇ **“Syntactical” approach**

To handle an approximation of the whole constraint system with the **simplex algorithm**

$\longrightarrow$ replace each non linear term by a new variable

$\longrightarrow$ introduce redundant linear constraints to get a **tight approximation** of the non-linear terms

$\longrightarrow$ solving a linear relaxation with the simplex algorithm

2.4 Global Constraints (2)

◇ **“Semantic” approach**

⟶ Distance constraint