Contraintes sur les Domaines Continus

Michel RUEHER

11 décembre 2007

1. Arithmétique des Intervalles
2. Propriétés d'un CSP Numérique
3. 2–B–Consistance et 2–B(w)–Consistance
4. 3B–Consistance
5. Box–Consistance et Bound–Consistance

1 Arithmétique des Intervalles

Notations

- x, y, z : variables définies sur les réels ; X, Y, Z : variables définies sur les intervalles ;
- u, v : constantes dans $\mathcal{R}$;
- f, g : fonctions définies sur les réels ;
- F, G : fonctions définies sur les intervalles ;
- c : contrainte définie sur les réels ;
- C : relation définie sur les intervalles ;
- $\Phi_{cstc}(P)$: fermeture (ou le filtrage) par la consistance $cstc$ (où $cstc$ est $2B$, Box, $3B$, $Bound$) de P .

Notations (suite)

- $\mathcal{R}^\infty = \mathcal{R} \cup \{-\infty, +\infty\}$: ensemble des réels étendu aux symboles des infinis.
- $\overline{\mathbb{F}}$: un sous-ensemble fini de $\mathcal{R}^\infty$ contenant en particulier $\{-\infty, +\infty\}$;
  ($\overline{\mathbb{F}}$: un ensemble de nombres flottants utilisés dans une implémentation machine)
- a, b : constantes dans $\overline{\mathbb{F}}$

- a^+ (resp. a^-) : plus petit (resp. plus grand) nombre de $\overline{\mathbb{F}}$ strictement plus grand (resp. plus petit) que a.

Bases du calcul d'intervalles

◇ Un **intervalle** $[a, b]$ avec $a, b \in \overline{\mathbb{F}}$ est l'ensemble de nombres réels $\{r \in \mathcal{R} \mid a \leq r \leq b\}$

◇ $\tilde{r}$: plus petit intervalle de $\overline{\mathbb{F}}$ contenant un nombre réel r

◇ $\square S$ est le plus petit intervalle I tel que $S \subseteq I$
(S est un sous ensemble de $\mathcal{R}$)

◇ $\mathcal{I}$: ensemble de tous les intervalles

◇ $\mathcal{U}(\mathcal{I})$: ensemble de toutes les unions d'intervalles

Incidence de la précision des calculs

Le terme plus petit (w.r.t. inclusion) sous-ensemble doit être interprêté en fonction de la précision des calculs lors des opérations en virgule flottante.

Hypothèses :

1. Tous les résultats des calculs sont arrondis vers l'extérieur
2. Erreur maximale lors du calcul d'une borne d'un intervalle est toujours inférieure à un flottant
   (peut nécessiter de recourir à des "grands flottants")

Extension aux intervalles

• Une **fonction** sur les intervalles $F : \mathcal{I}^n \rightarrow \mathcal{I}$ est une extension aux intervalles de $f : \mathcal{R}^n \rightarrow \mathcal{R}$ ssi :

$$\forall I_1, \ldots, I_n \in \mathcal{I} : r_1 \in I_1, \ldots, r_n \in I_n \Rightarrow f(r_1, \ldots, r_n) \in F(I_1, \ldots, I_n).$$

• Une **relation** sur les intervalles $C : \mathcal{I}^n \rightarrow \mathcal{B}ool$ est une extension aux intervalles de la relation $c : \mathcal{R}^n \rightarrow \mathcal{B}ool$ ssi :

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

Exemple :

Soit $I_1 \doteq I_2 \Leftrightarrow (I_1 \cap I_2) \neq \emptyset$

La relation $\doteq$ est une extension aux intervalles de la relation d'égalité sur les réels.

Extension naturelle aux intervalles

Une fonction sur les intervalles $F : \mathcal{I}^n \rightarrow \mathcal{I}$ est une extension naturelle aux intervalles de $f : \mathcal{R}^n \rightarrow \mathcal{R}$ ssi F est l'extension aux intervalles de f obtenue en remplaçant dans f :

- ◇ chaque constante k par son extension naturelle aux intervalles $\tilde{k}$
- ◇ chaque variable par une variable sur les intervalles
- ◇ chaque opération arithmétique par son extension optimale aux intervalles (opération qui calcule le plus petit intervalle qui conserve l'ensemble des solutions)

Extensions optimales aux intervalles pour les opérations de base sur les intervalles (Moore)

Soit $\odot$ un opérateur binaire parmi $\{+, -, *, /\}$ et
$[a,b] \odot [c,d] = \{x \odot y$ tel que $a \leq x \leq b$ et $c \leq y \leq d\}$ alors :

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

◇ Les extensions optimales aux intervalles peuvent être définies pour quasiment toutes les fonctions élémentaires

◇ **la forme syntaxique d'une fonction a une très forte influence** sur la précision de l'extension naturelle aux intervalles

Extension naturelle aux intervalles : exemples

Soit les fonctions sur les réels :

- $f_1(x) : x^2 - x$
- $f_2(x) : x(x-1)$

Extension naturelle aux intervalles :

- $F_1([0,5]) = [-5, 25]$
- $F_2([0,5]) = [-5, 20]$

Valeur de l'extension optimale aux intervalles de f_1 et f_2 sur $[0,5]$ est $[-0.25, 20]$
(la dérivée de f_1 et f_2 s'annulle pour $x = 0.5$).

2 Propriétés d'un CSP Numérique

Les opérateurs $\{+, *\}$ sont commutatifs et associatifs mais la *distributivité n'est pas vérifiée*. Seule la **sous-distributivité** est vérifiée :

- ◇ Soit $\mathcal{I}^n \rightarrow \mathcal{I}$ l'extension naturelle aux intervalles de : $\mathcal{R}^n \rightarrow \mathcal{R}$ et $\mathbf{f_{sol}} = \square\{\mathbf{f}(\mathbf{r_1}, \ldots, \mathbf{r_n}) \mid \mathbf{r_1} \in \mathbf{I_1}, \ldots, \mathbf{r_n} \in \mathbf{I_n}\}$ Si chaque $\mathbf{r_i}$ a une seule occurrence dans $\mathbf{f}$

  $$\textbf{alors } \mathbf{f_{sol}} = \mathbf{F}(\mathbf{I_1}, \ldots, \mathbf{I_n})$$
  $$\textbf{sinon } \mathbf{f_{sol}} \subseteq \mathbf{F}(\mathbf{I_1}, \ldots, \mathbf{I_n})$$
- ◇ Soit $C : \mathcal{I}^n \rightarrow \mathcal{B}ool$ l'extension naturelle aux intervalles d'une équation $\mathbf{c} : \mathcal{R}^\mathbf{n} \rightarrow \mathcal{B}\mathbf{ool}$. Si chaque $\mathbf{x_i}$ a une seule occurrence dans c, alors :
  $\mathbf{C}(\mathbf{D_1}, \ldots, \mathbf{D_n}) \Leftrightarrow (\exists \mathbf{x_1} \in \mathbf{D_1}, \ldots, \exists \mathbf{x_n} \in \mathbf{D_n} \mid \mathbf{c}(\mathbf{x_1}, \ldots, \mathbf{x_n}))$

Limites de la sous-distributivité

Exemple :

a) $[1,2] * ([1,2] - [1,2]) = [-2,2]$

$[1,2] * [1,2] - [1,2] * [1,2] = [-3,3]$

b) $f_1 : \mathcal{R} \to \mathcal{R}$ $\qquad$ $F_1 : I \to I$

$x \to x - x$ $\qquad$ $i \to i - i$

Si $I_0 = [10,20]$ alors

$F_1(I_0) = [-10,10]$

CSP sur les domaines continus

- $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ où $\mathcal{X} = \{x_1, \ldots, x_n\}$ est un ensemble de variables
- $\mathcal{D} = \{D_{x_1}, \ldots, D_{x_n}\}$ est un ensemble de domaines (D_{x_i} est le domaine contenant toutes les valeurs potentielles de la variable x_i)
- $\mathcal{C} = \{c_1, \ldots, c_m\}$ est un ensemble de contraintes

On note $Var(C)$ le sous ensemble de $\mathcal{X}$ des variables de C.

Relations entre CSP

- $\mathcal{D}' \subseteq \mathcal{D}$ signifie que $D'_{x_i} \subseteq D_{x_i}$ pour tout $i \in 1..n$

- Le CSP $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ est plus petit que $P' = (\mathcal{X}, \mathcal{D}', \mathcal{C})$ si $\mathcal{D} \subseteq \mathcal{D}'$, on note $P \prec P'$

- $P_\emptyset$ représente la classe des CSP vides (CSP avec au moins un domaine vide)
  Par convention $P_\emptyset$ est le plus petit des CSP.

Approximation d'un CSP

Il est en général impossible de calculer toutes les solutions d'un CSP
$\rightarrow$ calcul d'une approximation de cet ensemble de solutions.

Soit A un domaine d'approximation sur $\mathcal{R}$, ρ une relation n-aire sur $\mathcal{R}$. La fonction $N : A^n \rightarrow A^n$ est un opérateur de narrowing pour la relation ρ ssi pour tout $u, v \in A^n$ on a :

1. $N(u) \subset u$ (*contractance*)
2. $u \cap \rho \subset N(u)$ (*correction*)
3. $u \subset v \Rightarrow N(u) \subset N(v)$ (*monotonie*)

Algorithme standard de *narrowing*

1 **IN–1** (**in** $\mathcal{C}$, **inout** $\mathcal{D}$)
2 $Q \leftarrow \{< x_i, C_j > \mid C_j \in \mathcal{C}$ and $x_i \in Var(C_j)\}$
3 **while** $Q \neq \emptyset$
4 extract $< x_i, C_j >$ from Q
5 $\mathcal{D}' \leftarrow$ `narrowing`$(\mathcal{D}, x_i, C_j)$
6 **if** $\mathcal{D}' \neq \mathcal{D}$ **then**
7 $\mathcal{D} \leftarrow \mathcal{D}'$
8 $Q \leftarrow Q \cup \{< x_l, C_k > |(x_l, x_i) \in Var(C_k)\}$
10 **endif**
11 **endwhile**

Consistances partielles

◇ les consistances **strictement locales**
ne travaillent que sur une seule contrainte

◇ les consistances **partielles** qui ne sont pas strictement locales
→ vérifient certaines propriétés sur un sous-ensemble du système de contraintes

3 2–B–Consistance et 2–B(w)–Consistance

La 2–B–Consistance : approximation de la consistance d'arc

Intuitions :

◇ La contrainte c est 2B–Consistante pour la variable x, de domaine $D_x = [a, b]$, s'il existe des valeurs dans les domaines de toutes les autres variables qui satisfont c lorsque x est instancié avec a et lorsque x est instancié avec b.

◇ si c peut s'écrire sous la forme $f(x) = 0$, alors a et b sont respectivement le plus petit et le plus grand zéro de $f(x)$

2B–Consistance

Soit $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ un CSP et $c \in \mathcal{C}$ une contrainte k-aire sur les variables $(x_1, \ldots, x_k)$. La contrainte c est 2B–Consistante ssi :

$$\forall i, D_{x_i} = \square\{v_i \in D_{x_i} \mid \exists v_1 \in D_{x_1}, \ldots, \exists v_{i-1} \in D_{x_i-1}, \exists v_{i+1} \in D_{x_i+1}, \ldots \exists v_k \in D_{x_k} \text{ tel que } c(v_1, \ldots, v_{i-1}, x_i, v_{i+1} \ldots, v_k) \}$$

Un CSP est 2B–Consistant ssi toutes ses contraintes sont 2B–Consistantes

Consistance d'arc versus 2–B–Consistance

La 2B–Consistance est une consistance plus faible que la consistance d'arc :

- ◇ La consistance d'arc impose une condition sur tous les éléments des domaines
- ◇ La 2–B–Consistance impose une condition aux seules bornes de l'intervalle qui contient les domaines

Exemple

$P = \{\{x, y\}, \{D_x, D_y\}, \{x = y^2\}\}$ avec $D_x = [1, 4],\ D_y = [-2, +2]$

P est 2–B–Consistant, mais pas consistant d'arc car la valeur 0 de D_y n'a pas de support dans D_x

Relation entre la 2B–Consistance et l'extension aux intervalles d'une contrainte

Soit $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ un CSP tel qu'aucune contrainte de $\mathcal{C}$ ne contienne des occurrences multiples des variables de $\mathcal{X}$,
soit $c \in \mathcal{C}$ une contrainte d'arité k définie sur $(x_1, \ldots, x_k)$, c est 2–B–Consistant ssi $\forall x_i \in \{x_1, \ldots, x_k\}$, avec $D_{x_i} = [a, b]$:

- $C(D_{x_1}, \ldots, D_{x_{i-1}}, [a, a^+), D_{x_{i+1}}, \ldots, D_{x_k})$
- $C(D_{x_1}, \ldots, D_{x_{i-1}}, (b^-, b], D_{x_{i+1}}, \ldots, D_{x_k})$

où $[a, a^+)$ et $(b^-, b]$ sont des intervalles semi-ouverts.

Filtrage par 2–B–Consistance

Définition 3.1 (filtrage par 2B–Consistance)**.** *Un filtrage par 2–B–Consistance d'un CSP* $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ *est un CSP* $P' = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ *tel que :*

1. $P' \equiv P$*, i.e.* P *et* P' *ont les mêmes solutions ;*
2. P' *est 2–B–Consistant*
3. $P' \preceq P$
4. $\mathcal{D}' \subset \mathcal{D}$ *et les domaines de* $\mathcal{D}'$ *sont les plus grands domaines tel que* P' *soit un filtrage par 2–B–Consistance de* P

Le filtrage par 2–B–Consistance existe toujours et est unique

On note $P' = \Phi_{2B}(\mathrm{P})$.

Calcul de la 2–B–Consistance

$\rightarrow$ Approximation des fonctions de projection (utilisées dans `narrowing`$(\mathcal{D}, x, C)$ de `IN-1`)

Soit c une contrainte k-aire, la projection $\pi_i(c, D_1 \times \cdots \times D_k)$ sur x_i de l'ensemble des solutions de c dans l'espace délimité par $D_1 \times \cdots \times D_k$ est :

$$\pi_i(c, D_1 \times \cdots \times D_k) =$$
$$\{v_i \in I_i \mid \exists < v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_k > \in$$
$$I_1 \times \cdots \times I_{i-1} \times I_{i+1}, \ldots \times I_k \textit{ et } c(v_1, \ldots, v_i, \ldots, v_k)\}$$

La projection d'une contrainte sur une variable n'est en général pas un intervalle mais une union d'intervalles

Approximation de la projection

$AP_i(c, I_1 \times \cdots \times I_k)$ est une approximation minimale de $\pi_i(c, I_1 \times \cdots \times I_k)$ ssi $AP_i(c, I_1 \times \cdots \times I_k)$

$$= \square\ \pi_i(c, I_1 \times \cdots \times I_k)$$
$$= [Min\ \pi_i(c, I_1 \times \cdots \times I_k), Max\ \pi_i(c, I_1 \times \cdots \times I_k)].$$

($AP_i(c, I_1 \times \cdots \times I_k)$ est le plus petit intervalle contenant la projection $\pi_i(c, I_1 \times \cdots \times I_k)$)

Une contrainte c est 2B–Consistante sur $< I_1, \ldots, I_k >$ si pour tout i dans $\{1, \ldots, k\}$, $I_i = AP_i(c, I_1 \times \cdots \times I_k)$

Calcul de AP_i

◇ Le calcul de AP_i est immédiat pour les fonctions monotones : évaluation de la fonction de projection pour les bornes de chaque D_i.

◇ AP_i ne peut pas être calculée directement dans le cas général où il est difficile de définir les fonctions Min et Max
→ décomposition des contraintes en fonctions primitives pour lesquelles il est facile d'identifier les parties monotones.

Exemples d'évaluation de fonctions de projection

(a) Soit $x = y + z^2$ et $D_x = [0, 6], D_y = [5, 9], D_z = [-2, 4]$.
La fonction de projection pour y est $D'_y \leftarrow (D_x - D_z^2) \cap D_y$.
D'ou : $D'_y = ([0, 6] - [0, 16]) \cap [5, 9] = [5, 6]$.
(pour le calcul de D_z^2 on a utilisé l'extension optimale aux intervalles de la fonction puissance et non le produit)

(b) Soit $x = y^2$ et $D'_y \leftarrow \sqrt{D_x} \cap D_y$
L'évaluation de la fonction de projection demande dans ce cas une analyse des parties monotones :

- Si $D_x = [4, 9]$ et $D_y = [-1, 5]$ alors D'_y vaut $[2, 3]$ et non $[-1, 3]$ car aucune des valeurs entre -1 et $\sqrt{D_x}$ n'a de support dans D_x.
- Par contre, si $D_x = [4, 9]$ et $D_y = [-2, 5]$ alors D'_y vaut $[-2, 3]$ car -2 a pour support $-\sqrt{4}$.

Décomposition d'un système de contraintes

◇ La décomposition d'un système de contraintes $\mathcal{C}$ en un système de contraintes primitives $decomp(\mathcal{C})$ ne change pas la sémantique : $\mathcal{C}$ et $decomp(\mathcal{C})$ ont les mêmes solutions.

◇ la portée des vérifications effectuées par la 2B–Consistante est réduite par la décomposition : si une variable x possède des occurrences multiples dans une contrainte $c \in \mathcal{C}$, alors les différentes occurrences de x dans $decomp(c)$ pourront varier indépendamment les unes des autres
$\rightarrow$ le filtrage par 2B–Consistance de $decomp(\mathcal{C})$ sera plus faible que celui de $\mathcal{C}$

Décomposition d'un système de contraintes (suite)

Soit c : $x_1 + x_2 - x_1 = 0$ avec $D_{x_1} = [-1, 1],\ D_{x_2} = [0, 1]$

$\rightarrow$ $decomp(c) = \{x_1 + x_2 - x_3 = 0, x_1 = x_3\}$.

Chaque fonction de projection de $decomp(c)$ peut être évaluée à l'aide des opérations du calcul d'intervalle,
e.g., $AP_1(x_1 + x_2 - x_3 = 0, D_{x_1}, D_{x_2}, D_{x_3})$ est $D_{x_1} \cap (D_{x_3} \ominus D_{x_2})$

La contrainte c n'est pas 2–B–Consistante (pas de support pour $x_2 = 1$*) alors que* $decomp(c)$ *est 2–B–Consistant (les valeurs* $x_1 = -1$ *et* $x_3 = 0$ *satisfont* $x_1 + x_2 - x_3 = 0$ *lorsque* $x_2 = 1$*)*

Algorithme de calcul de la 2–B–Consistance

$\rightarrow$ Calcul des fonctions extremum dans la fonction `narrowing` de l'algorithme `IN-1`

1 **function narrow** $(\mathcal{D}, x_i, C_j)$: set of domains
2 $\quad m \leftarrow Min_{x_i}(C, D_{x_i})$
3 $\quad M \leftarrow Max_{x_i}(C, D_{x_i})$
4 $\quad$ **return** $\mathcal{D}[D_{x_i} \leftarrow [m, M]]$

FIG. 1 – Fonction `narrowing` pour le calcul de la 2–B–Consistance

Exemple 3.1 (filtrage par 2–B–Consistance)**.**
$c_1 : x + y = 1, c_2 : y = 2 * z, D_x = [-4, 5], D_y = [3, +\infty], D_z = [1, 2]$.
La file des couples <contrainte,variable> à traiter est
$Q = \{a :< c_1, x >, b :< c_1, y >, c :< c_2, y >, d :< c_2, z >\}$

1. *Sélection de a,* $D_x \leftarrow [-4, 2]$
   Aucun élément n'est ajouté dans Q
2. *Sélection de b,* $D_y \leftarrow [3, 5]$
   Aucun élément n'est ajouté dans Q
3. *Sélection de c,* $D_y \leftarrow [3, 4]$
   $Q = \{d :< c_2, z >, a :< c_1, x >\}$
4. *Sélection de d,* $D_z \leftarrow [1.5, 2]$
   Aucun élément n'est ajouté dans Q
5. *Sélection de a,* $D_x \leftarrow [-3, -2]$
   Aucun élément n'est ajouté dans Q *et* $Q = \emptyset$

Arrêt prématuré de l'algorithme de propagation

En cas de convergences asymptotiques, il n'est pas réaliste de vouloir réduire tous les intervalles jusqu'à ce qu'aucun élément de $\overline{\mathbb{F}}$ (i.e., aucun flottant) ne puisse être enlevé

$\rightarrow$ **arrêt de la propagation** avant d'atteindre le point fixe.

Exemple 3.2. *Soit :*

$X = 2 \times Y$

$Y = X$

$D_X = [-10, 10], D_Y = [-10, 10]$

La 2B-consistance va opérer les réductions suivantes :

$D_Y = [-5, 5]$	$D_X = [-5, 5]$
$D_Y = [-2.5, 2.5,$	$D_X = [-2.5, 2.5]$
$D_Y = [-1.25, 1.25]$	$D_X = [-1.25, 1.25]$
$D_Y = [-0.625, 0.625]$	$D_X = [-0.625, 0.625]$
.....	

il est préférable d'arrêter la propagation avant d'atteindre le point fixe ...

“Largeur” de borne

a^{+w} dénote le $(w+1)^{ieme}$ floattant après a.

a^{-w} dénote le $(w+1)^{ieme}$ floattant avant a.

2B(w)–Consistance

Soit $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ un CSP, $x \in \mathcal{X}$, $D_x = [a, b]$, w un entier positif. D_x est 2–B(w)–Consistant si pour toute contrainte $c(x, x_1, \ldots, x_k)$ de $\mathcal{C}$:

1) $\exists v \in [a, a^{+w}),\ \exists v_1, \ldots, v_k \in D_{x_1} \times \cdots \times D_{x_k} \mid c(v, v_1, \ldots, v_k)$

2) $\exists v' \in (b^{-w}, b],\ \exists v'_1, \ldots, v'_k \in D_{x_1} \times \cdots \times D_{x_k} \mid c(v', v'_1, \ldots, v'_k)$

Un CSP est 2–B(w)–Consistant ssi tous ses domaines sont 2–B(w)–Consistants

Algorithme de filtrage par 2B(w)–Consistance : modification de la fonction `Narrow` pour que les réductions de domaine effectuées soient supérieures à w.

Problèmes de la 2B(w)–Consistance

- ◇ Le filtrage par 2–B(w)–Consistance dépend de l'ordre d'évaluation des fonctions de projection (pas de point fixe)
- ◇ Il n'y a pas de rapport direct entre la valeur de w et la précision du filtrage
- ◇ Le mode de décomposition d'un système de contraintes influence l'ordre d'évaluation des fonctions de projection et peut donc affecter le résultat d'un filtrage par 2B(w_1)–Consistance
- ◇ Si les nombres manipulés ont des valeurs très hétérogènes, il est indispensable d'utiliser une valeur relative pour w

Exemple

$c_1 : x = y, \quad c_2 : x = z, \quad c_3 : x^2 = 4$

$D_x = [-10, 2], D_y = [-2, 2], D_z = [-1.5, 2]$ et $w = 1$

◇ Ordre 1 : $< c_1, c_2, c_3 >$

D'où $Q = \{< c_1, x >, < c_1, y >, < c_2, x >, < c_2, z >, < c_3, x >$

1. Sélection de $< c_1, x >$, $D_x \leftarrow [-\mathbf{2}, \mathbf{2}]$
2. Sélection de $< c_1, y >$, D_y ne change pas
   Aucun élément n'est ajouté dans Q
3. Sélection de $< c_2, x >$, D_x ne change pas car la réduction est inférieure à w
4. Sélections de $< c_2, z >, < c_3, x >$ ne permettent d'effectuer aucune réduction

Exemple (suite)

$c_1 : x = y, \quad c_2 : x = z, \quad c_3 : x^2 = 4$

$D_x = [-10, 2], D_y = [-2, 2], D_z = [-1.5, 2]$ et $w = 1$

◇ Ordre 2 : $< c_2, c_1, c_3 >$

D'où $Q = \{< c_2, x >, < c_2, z >, < c_1, x >, < c_1, y >, < c_3, x >$

1. Sélection de $< c_2, x >$, $D_x \leftarrow [-1.5, 2]$
   Aucun élément n'est ajouté dans Q
2. Sélection de $< c_2, z >, < c_1, x >, < c_1, y >$ ne permettent d'effectuer aucune réduction
3. Sélection de $< c_3, x >$, $D_x \leftarrow [\mathbf{2}, \mathbf{2}]$
   $< c_1, y >$ et $< c_2, z >$ sont ajoutés dans Q
4. Sélection de $< c_1, y >$, $D_y \leftarrow [\mathbf{2}, \mathbf{2}]$
5. Sélection de $< c_2, z >$, $D_z \leftarrow [\mathbf{2}, \mathbf{2}]$

Remarque :

$\mathbf{\Phi_{2B(w_1)}(P)} \rightarrow \mathbf{D_x = [a_1, b_1]}$ et $\mathbf{\Phi_{2B(w_2)}(P)} \rightarrow \mathbf{D_x = [a_2, b_2]}$
et $\mathbf{w_1 < w_2} \not\rightarrow [\mathbf{a_1, b_1}] \subset [\mathbf{a_2, b_2}]$

Un w plus grand peut empêcher la fonction de projection d'une contrainte c_j de réduire le domaine d'une variable x et ... permettre ainsi une réduction ultérieure plus significative !

Exemple :

$f_1 : y \leftarrow 0.71 \times x \quad f_2 : y \leftarrow 0.6 \times x + z$

$D_x = [0, 10] \quad Dy = [-10, 20] \quad D_Z = [0, 0.9]$

- ◇ Si $w = 2$, f_1 peut réduire D_y à $[0, 7.1]$ et f_2 ne peut plus effectuer aucune réduction
- ◇ Si $w = 3$, f_1 ne peut effectuer aucune réduction et f_2 peut réduire D_y à $[0, 6.9]$

4 3B–Consistance

Principe de la 3B–Consistance : vérifier si le système de contraintes ne devient pas trivialement inconsistant lorsqu'une variable est instanciée par la valeur d'une des bornes de l'intervalle qui lui est associée

$\rightarrow$ la 3B–Consistance cherche à vérifier si une borne peut effectivement être solution.

3B–Consistance (définition)

Soit $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ un CSP et x une variable de $\mathcal{X}$ de domaine $[a, b]$ et soit :

- $\diamond$ $P_{D_x^1 \leftarrow [a, a^+)}$ le CSP derivé de P en substituant D_x dans $\mathcal{D}$ par $D_x^1 = [a, a^+)$;
- $\diamond$ $P_{D_x^2 \leftarrow (b^-, b]}$ le CSP derivé de P en substituant D_x dans $\mathcal{D}$ par $D_x^2 = (b^-, b]$.

D_x est 3B–Consistant ssi $\Phi_{2B}(P_{D_x^1}) \neq P_\emptyset$ et $\Phi_{2B}(P_{D_x^2}) \neq P_\emptyset$.

Un CSP est 3–B–Consistant ssi tous ses domaines sont 3–B–Consistants

k–B–Consistance : généralisation de la 3–B–Consistance

Filtrage par 3B–Consistance

Un filtrage par 3–B–Consistance de P est un CSP P' qui vérifie :

1. $P' \equiv P$,
2. P' est 3–B–Consistant,
3. $P' \preceq P$,
4. $\forall \ P'' = (\mathcal{V}, \mathcal{D}'', \mathcal{C}), \{P'' \equiv P \ \wedge \ (P''$ est 3–B–Consistant$) \ \wedge \ P'' \preceq P \ \wedge \ P' \preceq P''\} \Rightarrow D'' = D'$.

Algorithme de filtrage par 3B–consistance = une suite de tentatives de refutation

Soit $D_x = [a, b]$ dans un CSP P, si $\Phi_{3B}(P_{D_x \leftarrow [a, \frac{a+b}{2}]}) = \emptyset$,

- $\diamond$ alors la portion $[a, \frac{a+b}{2})$ de D_x est éliminée et le filtrage se poursuit sur l'intervalle $[\frac{a+b}{2}, b]$;
- $\diamond$ sinon le filtrage se poursuit avec l'intervalle $[a, \frac{a+b}{4}]$.

L'algorithme s'arrête lorsque le point fixe est atteint ou lorsque l'intervalle qu'on essaye de refuter devient plus petit qu'une valeur absolue (ou relative) w fixée.

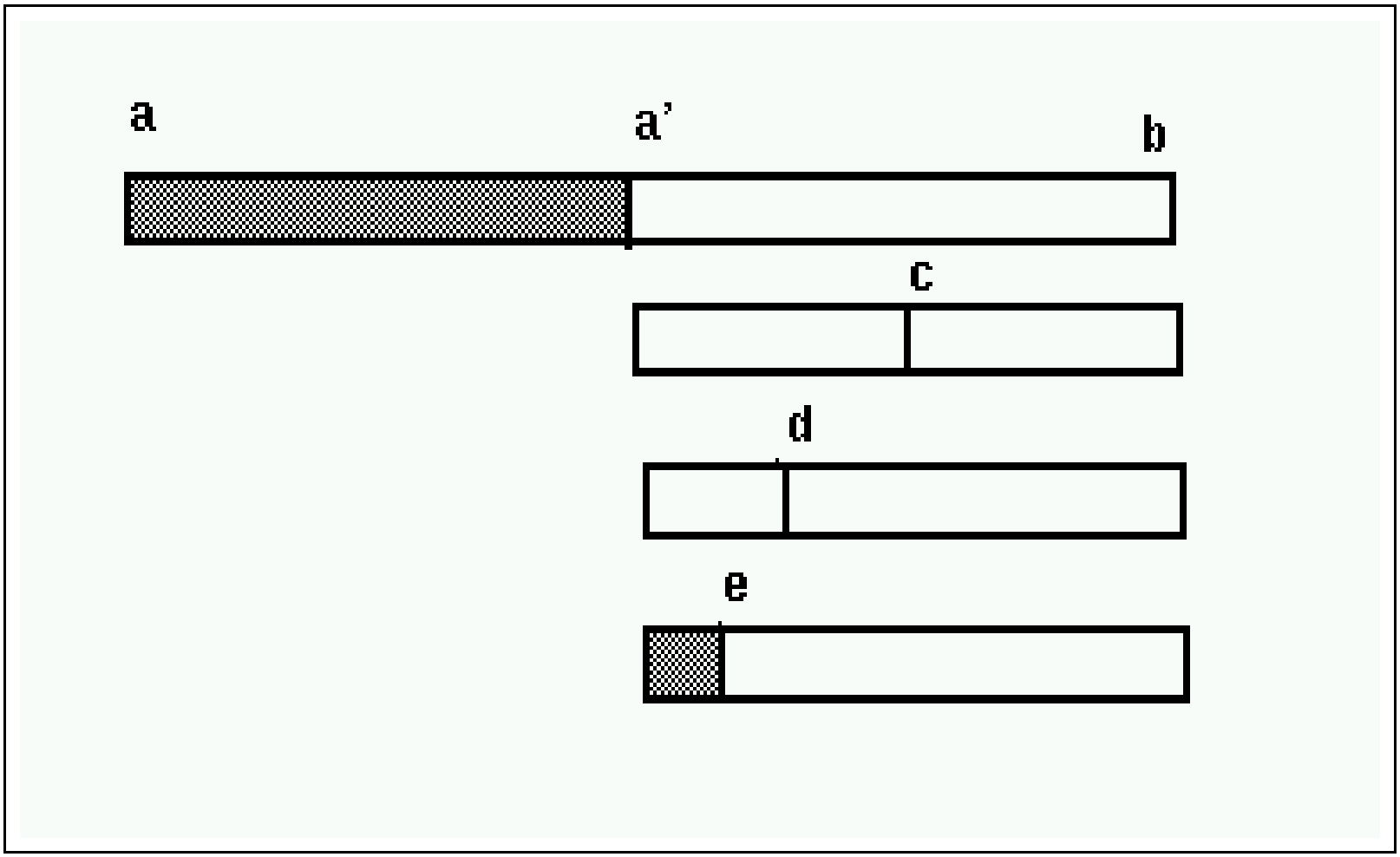

FIG. 2 – Exemple de réduction de la borne gauche

function 3B–filtering(P)

% Pré-condition : $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ est un CSP

% Post-condition : le résultat est un CSP $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ 3B consistant

% $\mathcal{S}$: Vecteur des Δ utilisés pour la réduction des domaines
(initialisation : $S_{x_i} \leftarrow \overline{D_{x_i}} - \underline{D_{x_i}}$)

% n : cardinal de $\mathcal{V}$

1 $\overrightarrow{PF} \leftarrow false$ % $\overrightarrow{PF}$: vecteur de booléens qui indique si le point fixe est atteint pour chaque domaine

2 **while** $\neg\overrightarrow{PF}$ **do**

3 **for** $i = 1 \ldots n$ **do**

4 **if** $\neg PF_{x_i}$ **then** $S_{x_i} \leftarrow \min(S_{x_i}, size(D_{x_i}))/2$

% Valeur initiale des S_{x_i} : maxvalue

5 **if** $S_{x_i} \leq w$ **then** $PF_{x_i} \leftarrow true$

6 $S_{x_i} \leftarrow w$

7 $\mathcal{D} \leftarrow$ 3B–fixpoint(P, S)

8 **return** P

function 3B–fixpoint(P, S)

% Pré-condition : $P = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ est un CSP

% Post-condition : le résultat est un CSP $P = (\mathcal{V}, \mathcal{D}, \mathcal{C}) \mid P = \Phi_2(P)$

1 $PF \leftarrow false$ % PF : booléen qui indique si le point fixe est atteint

2 **while** $\neg PF$ **do**

3 $PF \leftarrow true$

4 **for** $i = 1 \ldots n$ **do**% n cardinal de $\mathcal{V}$

5 $P_{borne\_inf} = (\mathcal{V}, \{D_{x_1}, \ldots, D_{x_{i-1}}, [a, a + S_{x_i}], D_{x_{i+1}}, \ldots, D_{x_n}\}, \mathcal{C})$

6 **if** 2B–filtering$(P_{borne_inf}) = P_\varnothing$ **then**

7 $PF \leftarrow false$

8 $D_{x_i} \leftarrow D_{x_i} \setminus [a, a + S_{x_i}]$

9 $\mathcal{D} \leftarrow$ 2B–filtering(P)

10 $P_{borne\_sup} = (\mathcal{V}, \{D_{x_1}, \ldots, D_{x_{i-1}}, [b - S_{x_i}, b], D_{x_{i+1}}, \ldots, D_{x_n}\}, \mathcal{C})$

11 **if** 2B–filtering$(P_{borne_sup}) = P_\varnothing$ **then**

12 $PF \leftarrow false$

13 $D_{x_i} \leftarrow D_{x_i} \setminus [b - S_{x_i}, b]$

14 $\mathcal{D} \leftarrow$ 2B–filtering(P)

15 **return** $\mathcal{D}$

Exemple

Soit $\mathcal{C} = \{x - y = 0, x + y = 100\}, D_x = D_y = [0, 100]$.
Ce système est 2–B–Consistant mais non 3–B–Consistant : la 3–B–Consistance permet d'approximer avec précision l'unique solution.

Remarques :

- ◇ la 3–B–Consistance calcule la consistance totale si le système de contraintes n'a que deux variables ;
  la k–B–Consistance fournit une procédure pour décider de la satisfiabilité d'un système de $k - 1$ variables.
- ◇ Relation entre $\Phi_{2B}(\mathrm{P})$ et $\Phi_{3B}(P_{decomp})$:
  - la relation $\Phi_{2B}(\mathrm{P}) \preceq \Phi_{3B}(P_{decomp})$ n'est pas vérifiée
  - la relation $\Phi_{3B}(P_{decomp}) \preceq \Phi_{2B}(P)$ n'est pas vérifiée

$\Phi_{2B}(\mathbf{P}) \not\preceq \mathbf{\Phi_{3B}}(\mathbf{P_{decomp}})$ (exemple)

$\mathcal{C} = \{x_1 + x_2 = 10; x_1 + x_1 - 2 \times x_2 = 0\}$, $D_{x_1} = D_{x_2} = [-10, 10]$

$decomp(x_1 + x_1 - 2 \times x_2 = 0) = \{x_1 + x_3 - 2 \times x_2 = 0, x_3 = x_1\}$

P est 2B–Consistant mais P_{decomp} n'est pas 3B–Consistant : lorsque x_1 est fixé à 10, $\Phi_{2B}(P_{D_{x_1} \leftarrow [10^-, 10]}) = P_\emptyset$ car D_{x_2} est réduit à $\emptyset$.

Le lien entre x_1 et x_3 est préservé et la 3B–Consistance reduit D_{x_2} à [5,5].

$\mathbf{\Phi_{3B}(P_{decomp})} \not\preceq \Phi_{2B}(\mathbf{P})$ (exemple)

$\mathcal{C} = \{x_1 * x_2 - x_1 + x_3 - x_1 + x_1 = 0\}$
$D_{x_1} = [-4, 3], D_{x_2} = [1, 2], D_{x_3} = [-1, 5]$
$decomp(c) = \{x_1 * x_2 - x_4 + x_3 - x_5 + x_6 = 0, x_1 = x_4 = x_5 = x_6\}$

c n'est pas 2B–Consistant car aucun couple de valeurs de D_{x_1} et de D_{x_2} ne vérifie la relation lorsque $x_3 = 5$.

$decomp(c)$ est 3B–Consistant.

La perte du lien entre les occurrences de x_1 interdit la réduction du domaine de x_3.

5 La Box–Consistance et Bound–Consistance

◇ La Box–Consistance est une approximation plus grossière de la consistance d'arc que la 2B–Consistance

◇ La Box–Consistance génère un système de fonctions univariables qui peut être résolu par la méthode de Newton

◇ La Box–Consistance n'amplifie pas le problème de localité des consistances partielles

Définition et propriétés de la Box–Consistance

Soit $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ un CSP et $c \in \mathcal{C}$ une contrainte k–aire définie sur les variables $(x_1, \ldots, x_k)$. c est Box–Consistant si, pour tout x_i dans $\{x_1, \ldots, x_k\}$ tel que $D_{x_i} = [a, b]$, les relations ci-dessous sont vérifiées :

1. $C(D_{x_1}, \ldots, D_{x_{i-1}}, [a, a^+), D_{x_{i+1}}, \ldots, D_{x_k})$,
2. $C(D_{x_1}, \ldots, D_{x_{i-1}}, (b^-, b], D_{x_{i+1}}, \ldots, D_{x_k})$.

Le filtrage par Box–Consistance de P est défini de manière similaire au filtrage par 2B–Consistance de P, et est noté $\Phi_{Box}(P)$.

Relation entre la Box–consistance et la 2B–consistance

$\Phi_{2B}(\mathrm{P}) \preceq \Phi_{Box}(\mathrm{P})$ et $\Phi_{2B}(\mathrm{P}) \equiv \Phi_{Box}(\mathrm{P})$ si aucune variable n'a d'occurrences multiples dans les contraintes de $\mathcal{C}$

Tout CSP qui est 2B–Consistant est aussi Box–Consistant, alors qu'un CSP peut être Box–Consistant sans être 2B–Consistant

Example :
$c\ : x_1 + x_2 - x_1 = 0, D_{x_1} = [-1, 1], D_{x_2} = [0, 1]$ n'est pas 2B–Consistant pour x_2 mais est Box–Consistant pour x_2 car
$([-1, 1] \oplus [0, 0^+] \ominus [-1, 1]) \cap [0, 0]$ et
$([-1, 1] \oplus [1^-, 1] \ominus [-1, 1]) \cap [0, 0]$ ne sont pas vides.

$$\Phi_{Box}(\mathbf{P}) \preceq \Phi_{2B}(\mathbf{P_{decomp}})$$

la 2B–Consistance sur le système décomposé est plus faible que la Box–Consistance sur le système initial

Preuve :

Pour le calcul des consistances locales, CSP P_{decomp} est une relaxation de P
$\rightarrow \Phi_{Box}(\mathrm{P}) \preceq \Phi_{Box}(P_{decomp})$.
Or du fait que P_{decomp} ne contient pas d'occurrences multiples de variables, on a : $\Phi_{Box}(P_{decomp}) \equiv \Phi_{2B}(P_{decomp})$
et donc $\Phi_{Box}(P) \preceq \Phi_{2B}(P_{decomp})$

Exemple

Soit $c : x_1 + x_2 - x_1 - x_1 = 0$ et $D_{x_1} = [-1, 1]$, $D_{x_2} = [0.5, 1]$

c n'est pas Box–Consistant car
$[-1, -1^+] \oplus [0.5, 1] \ominus [-1, -1^+] \ominus [-1, -1^+] \cap [0, 0]$ est vide

$decomp(c)$ est 2B–Consistant pour D_{x_1} and D_{x_2}

La Box—Consistance reste toutefois une consistance locale, et on peut montrer que $\Phi_{3B}(P_{decomp}) \preceq \Phi_{Box}(\mathrm{P})$.

Calcul de la Box–Consistance

La fonction $narrow(c, \mathcal{D})$ (algorithme générique `IN`) réduit les domaines des variables of c jusqu'à ce que c soit Box–Consistant :

- ◇ Pour chaque variable x de la contrainte c, une fonction univariable sur intervalles est générée en remplaçant toutes les variables sauf x par leur domaine

- ◇ Recherche du zéro le plus à gauche et la plus à droite de ces fonctions univariables sur intervalles qui sont de la forme :

$$C(D_{x_1}, .., D_{x_{i-1}}, x, D_{x_{i+1}}, ..., D_{x_k}) = \tilde{0}.$$

Fonction `LNAR` calcule le zéro le plus à gauche de F_x pour la variable x de domaine initial I_x

- `LNAR` réduit d'abord le domaine de l'intervalle I_x avec la fonction $NEWTON(F_x, I_x)$ (extension aux intervalles de la méthode de Newton)

- Si $NEWTON(F_x, I_x)$ ne peut pas réduire suffisamment le domaine I_x pour le rendre Box–Consistant, une division du domaine est effectuée pour s'assurer que la borne gauche de I_x est effectivement un zéro

- La fonction `SPLIT` divise l'intervalle I en deux intervalles I_1 et I_2 ; I_1 étant la partie gauche de l'intervalle initial

```
function LNAR (IN : F_x, I_x, return Interval)
    r ← right(I_x)
    if  0 ∉ F_x(I_x) then return ∅
        else  I_x ← NEWTON(F_x, I_x)
              if 0 ∈ F_x([left(I_x), left(I_x)^+]) then return [left(I_x), r]
                 else  SPLIT(I_x, I_1, I_2)
                       L_1 ← LNAR(F_x, I_1)
                       if L_1 ≠ ∅ then return [left(L_1), r]
                          else return [left(LNAR(F_x, I_2)), r]
                       endif
              endif
    endif
```

FIG. 3 – Function LNAR

Soit :$f : \mathcal{R} \rightarrow \mathcal{R}$.

D'après le théorème, on a pour toute valeur a entre u et v :

$$f(u) - f(v) = (u - v)f'(a)$$

Si v est un zéro de f on obtient : $v = u - \frac{f(u)}{f'(a)}$

D'après l'arithmétique des intervalles, si $a \in I$ alors $f(a) \in F(I)$

Si F est l'extension naturelle aux intervalles de f,

alors $v \in \tilde{u} - \frac{F(\tilde{u})}{F'(I)} = N(F, F', \tilde{u}, I)$

D'ou, v est un zéro de f si $v \in N^*(F, F', I)$ avec :

$N^*(F, F', I) = I_n (n \geq 1)$ où :

$$I_0 = I$$

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

$$I_n = I_{n-1}$$

FIG. 4 – Rappel du principe de l'opérateur itératif de Newton

Bound–Consistance : généralisation de la Box–Consistance

La Bound–Consistance applique le principe de la 3B–Consistance à la Box–Consistance : elle vérifie si la Box–Consistance peut réduire un domaine vide lorsque le domaine d'une variable est réduit à la valeur d'une de ses bornes.