

Contraintes sur les Domaines Continus

—Notes de cours (brouillon)—

Michel RUEHER

Décembre 2003

Contents

| | | |
|-----|--|----|
| 1 | Arithmétique des Intervalles | 4 |
| 1.1 | Notations | 4 |
| 1.2 | Bases du calcul d'intervalles | 6 |
| 2 | Propriétés | 11 |
| 2.1 | CSP "numérique" | 14 |
| 2.2 | Sémantique d'un CSP | 18 |
| 2.3 | Approximation d'un CSP | 19 |
| 2.4 | Algorithme standard de narrowing | 21 |
| 2.5 | Consistances partielles | 22 |
| 3 | 2-B-Consistance | 24 |
| 3.1 | Consistance d'arc versus 2-B-Consistance | 26 |

| | | |
|-----|--|----|
| 3.2 | filtrage par 2-B-Consistance | 29 |
| 3.3 | Calcul de la 2-B-Consistance | 30 |
| 3.4 | Algorithme de calcul de la 2-B-Consistance | 36 |
| 4 | 2-B(w)-Consistance | 38 |
| 4.1 | Arrêt prématuré de l'algorithme de propagation | 38 |
| 4.2 | Problèmes de la 2B(w)-Consistance | 42 |
| 5 | 3B-Consistance | 46 |
| 5.1 | 3B-Consistance | 47 |
| 6 | La Box-Consistance | 56 |
| 6.1 | Définition et propriétés de la Box-Consistance | 57 |
| 6.2 | Calcul de la Box-Consistance | 61 |
| 7 | Bound-Consistance | 65 |

1 Arithmétique des Intervalles

1.1 Notations

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

- ◇ $\mathcal{R}^\infty = \mathcal{R} \cup \{-\infty, +\infty\}$ est l'ensemble des réels étendu aux symboles des infinis.
- ◇ $\overline{\mathbb{F}}$ est un sous-ensemble fini de \mathcal{R}^∞ contenant en particulier $\{-\infty, +\infty\}$;
- ◇ a, b représentent des constantes dans $\overline{\mathbb{F}}$.
Concrètement, $\overline{\mathbb{F}}$ correspond à un ensemble de nombres flottants utilisés dans une implémentation machine.
- ◇ a^+ (resp. a^-) correspond au plus petit (resp. plus grand) nombre de $\overline{\mathbb{F}}$ strictement plus grand (resp. plus petit) que a .

1.2 Bases du calcul d'intervalles

Définition 1.1 (Intervalle). Un intervalle $[a, b]$ avec $a, b \in \overline{\mathbb{F}}$ est l'ensemble de nombres réels $\{r \in \mathcal{R} \mid a \leq r \leq b\}$.

Soit r un nombre réel. \tilde{r} correspond au plus petit (w.r.t. inclusion) intervalle de $\overline{\mathbb{F}}$ contenant r . \mathcal{I} correspond à l'ensemble de tous les intervalles et est partiellement ordonné par l'inclusion ensembliste. $\mathcal{U}(\mathcal{I})$ correspond à l'ensemble de toutes les unions d'intervalles.

Commentaire 1.1 (Incidence de la précision des calculs).

Le terme plus petit (w.r.t. inclusion) sous-ensemble doit être interprété en fonction de la précision des calculs lors des opérations en virgule flottante. Nous supposons ici que :

1. tous les résultats des calculs sont arrondis vers l'extérieur (i.e., vers $-\infty$ lors du calcul de la borne inférieure d'un intervalle et vers $+\infty$ lors du calcul de la borne supérieure d'un intervalle) ;
2. l'erreur maximale lors du calcul d'une borne de l'intervalle associé à une variable du système de contraintes initial est toujours inférieure à un flottant.

La deuxième hypothèse peut nécessiter de recourir à des "grands flottants". Elle est toutefois requise pour pouvoir garantir que les algorithmes de filtrage basés sur une consistance partielle calculent un point fixe unique

Définition 1.2 (Extension ensembliste). Soit S un sous ensemble de \mathcal{R} . l'approximation de S —notée $\square S$ — est le plus petit intervalle I tel que $S \subseteq I$.

Définition 1.3 (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 :

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

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

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

Par exemple, la relation sur les intervalles \doteq définie comme $I_1 \doteq I_2 \Leftrightarrow (I_1 \cap I_2) \neq \emptyset$ est une extension aux intervalles de la relation d'égalité sur les réels.

Définition 1.4 (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)

Les extensions optimales aux intervalles ont été introduites par Moore pour les opérations de base sur les intervalles. Par exemple, soit \odot un opérateur binaire parmi $\{+, -, *, /\}$ et $[a, b] \odot [c, d] = \{x \odot y \text{ tel que } a \leq x \leq b \text{ et } c \leq y \leq d\}$ alors l'extension optimale aux intervalles pour les autres opérations de base est définie par :

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

es extensions optimales aux intervalles peuvent être définies pour quasiment toutes les fonctions élémentaires . Il est important de remarquer que la forme syntaxique d'une fonction a une très forte influence sur la précision de l'extension naturelle aux intervalles qui lui sera associée.

Exemple 1.1. Soit $f(x) = x + x \times y + 3$ une fonction sur les réels. Son extension naturelle aux intervalles est définie par $X \oplus X \otimes Y \oplus \tilde{3}$.

Exemple 1.2. Soit les fonctions sur les réels définies par :

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

L'extension naturelle aux intervalles nous donne le résultat suivant :

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

alors que la valeur de l'extension optimale aux intervalles de ces deux fonctions sur l'intervalle $[0, 5]$ correspond à l'intervalle $[-0.25, 20]$: le domaine de variation de f_1 et f_2 est $[-0.25, 20]$ lorsque x varie entre $[0, 5]$ (la dérivée de ces deux fonctions s'annulent pour $x = 0.5$).

2 Propriétés fondamentales de l'arithmétique des intervalles

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

Proposition 1. Soit $f : \mathcal{R}^n \rightarrow \mathcal{R}$ l'extension naturelle aux intervalles de $f : \mathcal{R}^n \rightarrow \mathcal{R}$ et soit $f_{sol} = \square\{f(r_1, \dots, r_n) \mid r_1 \in I_1, \dots, r_n \in I_n\}$.

Si chaque r_i a une seule occurrence dans f alors

$f_{sol} = F(I_1, \dots, I_n)$ sinon $f_{sol} \subseteq F(I_1, \dots, I_n)$.

Ce résultat peut aisément être étendu aux relations k-aires sur \mathcal{R}^n .

Proposition 2. Soit $C : \mathcal{I}^n \rightarrow \mathcal{Bool}$ l'extension naturelle aux intervalles d'une équation $c : \mathcal{R}^n \rightarrow \mathcal{Bool}$. Si chaque x_i a une seule occurrence dans c , alors :

$$C(I_1, \dots, I_n) \Leftrightarrow (\exists x_1 \in D_1, \dots, \exists x_n \in D_n \mid c(x_1, \dots, x_n)).$$

Exemple 2.1. Voici quelques exemples qui illustrent les limites de la sous-distributivité :

$$\begin{aligned} \text{a)} \quad & [1, 2] * ([1, 2] - [1, 2]) = [-2, 2] \\ & [1, 2] * [1, 2] - [1, 2] * [1, 2] = [-3, 3] \end{aligned}$$

$$\begin{aligned} \text{b)} \quad & f_1 : \mathcal{R} \rightarrow \mathcal{R} \quad F_1 : I \rightarrow I \\ & x \rightarrow x - x \quad i \rightarrow i - i \end{aligned}$$

$$\text{Si } I_0 = [10, 20] \quad F_1(I_0) = [-10, 10]$$

2.1 Système de contraintes sur les intervalles

Définition 2.1. Un CSP sur les domaines continus est un triplet :

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

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

Exemple 2.2. Voici quelques exemples de CSP sur les domaines continus :

- a) $P_0 = (\mathcal{V}, \{D_U, D_R, D_I\}, \{U = R \times I\})$
avec $D_R = [10.5, 11], D_I = [1, 2], D_U = (-\infty, +\infty)$
- b) $P_1 = (V, D_1, C) P_2 = (V, D_2, C) P_3 = (V, D_3, C)$

avec :

$$V = \{x, y\}$$

$$D_1 = \{D_x = [0, 100], D_y = [0, 100]\}$$

$$D_2 = \{D_x =]0, 2], D_y = [0, 2[\}$$

$$D_3 = \{D_x = [0.5, 1], D_y = [1, 1.5]\}$$

$$C = \{x + y = 2, y \leq x + 1, y \geq 1 + \ln x\}$$

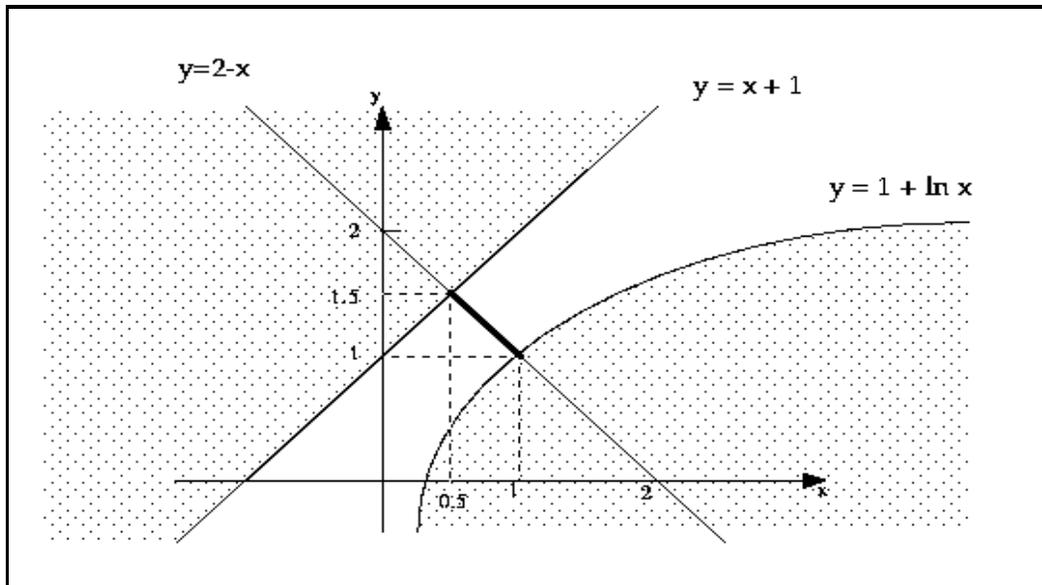


Figure 1: Contraintes des CSP P_1 , P_2 et P_3

Les CSP P_1 , P_2 et P_3 ont le même ensemble de solutions (cf. fig. 1).

Relations entre CSP

- ◇ P_\emptyset représente la classe des CSP vides, i.e., l'ensemble des CSP dont au moins un domaine est vide.
- ◇ $\mathcal{D}' \subseteq \mathcal{D}$ signifie que $D'_{x_i} \subseteq D_{x_i}$ pour tout $i \in 1..n$.
- ◇ un CSP $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ est plus petit qu'un CSP $P' = (\mathcal{X}, \mathcal{D}', \mathcal{C})$ si $\mathcal{D}' \subseteq \mathcal{D}$
On note $P \prec P'$ cette relation d'ordre.
- ◇ Par convention P_\emptyset est le plus petit des CSP.

2.2 Sémantique d'un CSP

Soit le CSP $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ avec $\mathcal{X} = \{x_1, \dots, x_n\}$,
 $\mathcal{D} = \{D_1, \dots, D_n\}$ et $\mathcal{C} = \{c_1, \dots, c_m\}$ où chaque contrainte est
d'arité n . L'ensemble des solutions du CSP P est défini par la
relation $\bigcap_{i=1}^m c_i \cap (D_{x_{j_1}} \times \dots \times D_{x_{j_k}})$.

2.3 Approximation d'un CSP

Comme il est en général impossible de calculer toutes les solutions d'un CSP, on cherche à calculer une approximation de cet ensemble de solutions.

Définition 2.2 (domaine d'approximation). Un domaine d'approximation A sur \mathcal{R} est un sous-ensemble des parties de \mathcal{R} clos par intersection, muni d'une relation d'ordre bien fondée, et tel que $\mathcal{R} \in A$.

\mathcal{I} correspond à l'ensemble des intervalles dont les bornes appartiennent à $\overline{\mathcal{I}}$, un sous ensemble fini de \mathcal{R} . \mathcal{I} est partiellement ordonné par l'inclusion ensembliste et constitue donc un domaine d'approximation de \mathcal{R} . Cet ensemble est un treillis complet dont la borne inférieure est définie par l'intersection ensembliste et la borne supérieure par l'union ensembliste.

Définition 2.3 (opérateur de narrowing). 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. $N(u) \subset u$,
2. $u \cap \rho \subset N(u)$,
3. $u \subset v \Rightarrow N(u) \subset N(v)$

La première propriété correspond à la contractance, la deuxième à la correction et la troisième à la monotonie.

2.4 Algorithme standard de narrowing

Un algorithme standard d'approximation d'un CSP est défini dans la figure 2. Cet algorithme est dérivé de l'algorithme AC3 utilisé pour calculer la consistance d'arc sur les domaines finis.

L'approximation qui sera effectivement calculée est définie par la consistance locale associée à l'opérateur de narrowing.

```

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

```

Figure 2: Algorithme standard d'approximation d'un CSP

2.5 Consistances partielles

Les consistances partielles utilisées au niveau des CSP sur les domaines continus se divisent en deux grandes catégories :

- ◇ les consistances strictement locales, i.e., qui ne travaillent que sur une seule contrainte
- ◇ les consistances partielles qui ne sont pas strictement locales, i.e., qui vérifient certaines propriétés sur l'ensemble (ou sur un sous-ensemble) du système de contraintes locales

3 2-B-Consistance

La 2-B-Consistance est une approximation de la consistance d'arc et est donc une consistance strictement locale.

La contrainte c est 2B-Consistante si pour toute variable x de domaine $D_x = [a, b]$ 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 .

Ou encore 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 $f(x)$ dans l'espace délimité par $D_{x_1} \times \dots \times D_{x_n}$.

Définition 3.1 (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, \dots, x_k) . La contrainte c est 2B-Consistante ssi :

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

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

3.1 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 cette même condition aux seules bornes de l'intervalle qui contient les domaines.

Exemple 3.1. $P = \{\{x, y\}, \{D_x, D_y\}, \{x = y^2\}\}$ avec

$$D_x = [1, 4], \quad D_y = [-2, +2]$$

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 .

La 2-B-Consistance ne garantit pas l'existence d'une solution.

Exemple 3.2. $P' = \{\{x, y\}, \{D_x, D_y\}, \{x^2 = 4, y^2 = 9, x * y = 1\}\}$
avec $D_x =] - 2, 2[$, $D_y = [-3, +3]$

P' est 2-B-Consistant, mais n'a pas de solution.

La propriété suivante explicite la relation entre la 2B-Consistance et l'extension aux intervalles d'une contrainte.

Proposition 3. 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, \dots, x_k) . c est 2-B-Consistant ssi pour tout x_i de $\{x_1, \dots, x_k\}$ tel que $D_{x_i} = [a, b]$ les relations ci-dessous sont vérifiées :

$$\diamond C(D_{x_1}, \dots, D_{x_{i-1}}, [a, a^+), D_{x_{i+1}}, \dots, D_{x_k}),$$

$$\diamond C(D_{x_1}, \dots, D_{x_{i-1}}, (b^-, b], D_{x_{i+1}}, \dots, D_{x_k}).$$

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

3.2 filtrage par 2-B-Consistance

Définition 3.2 (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 $P' = \Phi_{2B}(P)$. Ainsi, dans l'exemple 1 : $P_2 = \Phi_{2B}(P_1)$.

3.3 Calcul de la 2-B-Consistance

Le calcul de la 2-B-Consistance repose sur l'approximation des fonctions de projection utilisées pour réduire le domaine des variables dans la fonction `narrowing`(\mathcal{D}, x, C) de IN-1

Soit c une contrainte k -aire sur (x_1, \dots, x_k) , et

$\langle I_1, \dots, I_k \rangle \in \mathcal{I}^k$ un ensemble de domaines, $\pi_i(c, I_1 \times \dots \times I_k)$ dénote la projection sur x_i de l'ensemble des solutions de c dans l'espace délimité par $I_1 \times \dots \times I_k$.

Définition 3.3 (projection d'une contrainte).

$\pi_i(c, I_1 \times \dots \times I_k) : (\mathcal{C}, \mathcal{I}^k) \rightarrow \mathcal{U}(\mathcal{I})$ est la projection de c sur x_i ssi :

$$\pi_i(c, I_1 \times \dots \times I_k) = \{v_i \in I_i \mid \exists \langle v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_k \rangle \in I_1 \times \dots \times I_{i-1} \times I_{i+1}, \dots \times I_k \text{ tel que } c(v_1, \dots, v_i, \dots, v_k) \text{ est vrai} \}$$

La projection d'une contrainte sur une variable n'est en général pas un intervalle mais une union d'intervalles. Pour éviter de devoir gérer les unions d'intervalles on approxime cette projection par un intervalle.

Définition 3.4 (approximation de la projection).

$AP_i(c, I_1 \times \cdots \times I_k) : (\mathcal{C}, \mathcal{I}^k) \rightarrow \mathcal{I}$ est une approximation de $\pi_i(c, I_1 \times \cdots \times I_k)$ ssi

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

En d'autres termes, $AP_i(c, I_1 \times \cdots \times I_k)$ est le plus petit intervalle contenant la projection $\pi_i(c, I_1 \times \cdots \times I_k)$.

Il résulte de ces définitions qu'une contrainte c est 2B-Consistante sur $\langle I_1, \dots, I_k \rangle$ si pour tout i dans $\{1, \dots, k\}$, $I_i = AP_i(c, I_1 \times \cdots \times I_k)$.

- ◇ Le calcul de AP_i est immédiat pour les fonctions monotones : il suffit d'évaluer la fonction de projection pour les bornes de l'intervalle associé à la variable x_i .
- ◇ un des problèmes majeurs de la 2-B-Consistance est que AP_i ne peut pas être calculée directement dans le cas général où il est difficile de définir les fonctions *Min* et *Max*
Si une variable a des occurrences multiples. La solution consiste alors à décomposer les contraintes en fonctions primitives pour lesquelles il est facile d'identifier les parties monotones.

Exemple 3.3 (évaluation des fonctions de projection). Voici deux 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'où : $D'_y = ([0, 6] - [0, 16]) \cap [5, 9] = [5, 6]$.

On remarquera que pour calculer 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 $-\sqrt{4}$.

- ◇ 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.
- ◇ une propriété locale comme la 2B-Consistance, 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 et le filtrage par 2B-Consistance de $decomp(\mathcal{C})$ sera plus faible que celui de \mathcal{C} .

Exemple 3.4 (décomposition du système de contraintes).

Soit $c : x_1 + x_2 - x_1 = 0$ une contrainte $D_{x_1} = [-1, 1]$, $D_{x_2} = [0, 1]$ les domaines de x_1 et x_2 . Puisque x_1 apparait deux fois dans c , sa deuxième occurrence sera remplacée par une nouvelle variable x_3 .

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

Dans ce nouveau système de contraintes, chaque fonction de projection peut être évaluée aisément à l'aide des opérations du calcul d'intervalle. Par exemple :

$AP_1(x_1 + x_2 - x_3 = 0, D_{x_1}, D_{x_2}, D_{x_3})$ est $D_{x_1} \cap (D_{x_3} \ominus D_{x_2})$.

Toutefois la décomposition accentue les limites due à la localité de la 2-B-Consistance : la contrainte initiale n'est pas

2-B-Consistante (aucune valeur x_1 ne permet de satisfaire la contrainte lorsque $x_2 = 1$) alors que le système décomposé est

2-B-Consistant (les valeurs $x_1 = -1$ et $x_3 = 0$ satisfont

$x_1 + x_2 - x_3 = 0$ lorsque $x_2 = 1$).

3.4 Algorithme de calcul de la 2-B-Consistance

L'algorithme de calcul de la 2-B-Consistance résulte directement de l'instanciation de la fonction `narrowing` par le calcul des fonctions extremum (cf. figure 3) dans l'algorithme IN-1.

```
1 function narrow ( $\mathcal{D}, x_i, C_j$ ) : set of domains
2    $m \leftarrow \text{Min}_{x_i}(C, D_{x_i})$ 
3    $M \leftarrow \text{Max}_{x_i}(C, D_{x_i})$ 
4   return  $\mathcal{D}[D_{x_i} \leftarrow [m, M]]$ 
```

Figure 3: Fonction `narrowing` pour le calcul de la 2-B-Consistance

Exemple 3.5 (filtrage par 2-B-Consistance). Soit

$$c_1 : x + y = 1, c_2 : y = 2 * z, D_x = [-4, 5], D_y = [3, +\infty), D_z = [1, 2].$$

La file des couples $\langle \text{contrainte}, \text{variable} \rangle$ à traiter est

$$Q = \{a : \langle c_1, x \rangle, b : \langle c_1, y \rangle, c : \langle c_2, y \rangle, d : \langle c_2, z \rangle\}$$

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 , $D_y \leftarrow [3, 4]$.

$$Q = \{d : \langle c_2, z \rangle, a : \langle c_1, x \rangle\}$$

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 .

6. $Q = \emptyset$

4 2-B(w)-Consistance

4.1 Arrêt prématuré de l'algorithme de propagation

Lors de convergences asymptotiques il est en général préférable d'arrêter la propagation avant d'atteindre le point fixe. En effet, il n'est pas réaliste dans ce cas de vouloir réduire tous les intervalles jusqu'à ce qu'aucun élément de \overline{IF} (e.g., aucun flottant) ne puisse être enlevé.

Exemple 4.1. 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 ...

On introduit de ce fait une notion de “largeur” de borne (cf. figure 4) et on se contente de rechercher une solution dans cette borne large. On appelle $2B(w)$ -Consistance la consistance obtenue lors d’un tel arrêt prématuré de l’algorithme de filtrage.



Figure 4: illustration de la notion de borne “large”

Formellement, la 2B(w)–Consistance peut être définie de la manière suivante :

Définition 4.1. 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 $c(x, x_1, \dots, x_k)$ de \mathcal{C} si les relations suivantes sont vérifiées :

- 1) $\exists v \in [a, a^{+w}), \exists v_1, \dots, v_k \in D_{x_1} \times \dots \times D_{x_k} \mid c(v, v_1, \dots, v_k)$
 - 2) $\exists v' \in (b^{-w}, b], \exists v'_1, \dots, v'_k \in D_{x_1} \times \dots \times D_{x_k} \mid c(v', v'_1, \dots, v'_k)$
- où a^{+w} (resp. a^{-w}) dénote le $(w + 1)^{ieme}$ floattant après (resp. avant) a .

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

L’algorithme de filtrage par 2B(w)–Consistance s’obtient en modifiant la fonction Narrow pour que la réduction du domaine ne soit effectuée que lorsqu’elle est supérieure à w .

4.2 Problèmes de la 2B(w)–Consistance

- ◇ Le fait de ne plus réduire le plus possible les intervalles à chaque étape fait perdre la notion de point fixe : le filtrage par 2B(w)–Consistance dépend de l'ordre d'évaluation des fonctions de projection.
- ◇ Soit $D_x = [a, b]$ le domaine de x dans un CSP P , $[a_1, b_1]$ le domaine de x après filtrage par 2B(w)–Consistance de P , et $[a_2, b_2]$ le domaine de x après filtrage par 2B–Consistance de P . Il n'y a pas de rapport direct entre la valeur de w et la distance $|a_1 - a_2|$;
- ◇ Le mode de décomposition d'un système de contraintes peut avoir une influence sur l'ordre d'évaluation des fonctions de projection et peut de ce fait 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 4.2.

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

$$D_x = [-10, 2], D_y = [-2, 2], D_z = [-1.5, 2] \text{ et } w = 1.$$

◇ Ordre 1 : $\langle c_1, c_2, c_3 \rangle$

D'où $Q = \{ \langle c_1, x \rangle, \langle c_1, y \rangle, \langle c_2, x \rangle, \langle c_2, z \rangle, \langle c_3, x \rangle$

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

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

$$D_x = [-10, 2], D_y = [-2, 2], D_z = [-1.5, 2] \text{ et } w = 1.$$

◇ Ordre 2 : $\langle c_2, c_1, c_3 \rangle$

D'où $Q = \{\langle c_2, x \rangle, \langle c_2, z \rangle, \langle c_1, x \rangle, \langle c_1, y \rangle, \langle c_3, x \rangle$

1. Sélection de $\langle c_2, x \rangle$, $D_x \leftarrow [-1.5, 2]$.

Aucun élément n'est ajouté dans Q .

2. Sélection de $\langle c_2, z \rangle, \langle c_1, x \rangle, \langle c_1, y \rangle$ ne permettent d'effectuer aucune réduction.

3. Sélection de $\langle c_3, x \rangle$, $D_x \leftarrow [2, 2]$

$\langle c_1, y \rangle$ et $\langle c_2, z \rangle$ sont ajoutés dans Q .

4. Sélection de $\langle c_1, y \rangle$, $D_y \leftarrow [2, 2]$

5. Sélection de $\langle c_2, z \rangle$, $D_z \leftarrow [2, 2]$

Remarques :

Soit $D_x = [a, b]$ le domaine de x dans un CSP P , $[a_1, b_1]$ le domaine de x après filtrage par $2B(w_1)$ -Consistance de P , et $[a_2, b_2]$ le domaine de x après filtrage par $2B(w_2)$ -Consistance de P . $w_1 < w_2$ n'implique pas $[a_1, b_1] \subset [a_2, b_2]$.

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 à la fonction de projection d'une contrainte c_k d'effectuer une réduction plus significative sur le domaine de cette même variable x .

Exemple 4.3. Soit $f_1 : y \leftarrow 0.71 \times x$ $f_2 : y \leftarrow 0.6 \times x + z$

$D_x = [0, 10]$ $D_y = [-10, 20]$ $D_z = [0, 0.9]$

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 $[0.8, 6.9]$

5 3B–Consistance

La 2B–Consistance est uniquement une consistance partielle et comme la consistance d’arc elle est souvent trop faible pour permettre une approximation fine du domaine des variables. De même que des consistances plus fortes, basées sur l’arc consistance (e.g., consistance de chemin), ont été introduites dans les domaines finis, des consistances plus fortes ont été proposées “au dessus” de la 2B–Consistance.

5.1 3B-Consistance

Le principe de la 3B-Consistance est de vérifier si le système de contraintes ne devient pas trivialement inconsistant (i.e., la 2B-Consistance permet de générer des domaines vides) lorsqu'une variable est instanciée par la valeur d'une des bornes de l'intervalle qui lui est associée.

En d'autres termes, la 3B-Consistance cherche à vérifier si une borne peut effectivement être solution.

Définition 5.1 (3B-Consistance). Soit $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ un CSP et x une variable de \mathcal{X} de domaine $[a, b]$. Soit :

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

Exemple 5.1. Dans l'exemple 2.2, P_1 et P_2 ne sont pas 3-B-Consistants mais P_3 est 3-B-Consistant :

$$P_3 = \Phi_{3B}(P_1) = \Phi_{3B}(P_2).$$

Tout CSP qui est 3B-Consistant est aussi 2B-Consistant . La généralisation de la 3-B-Consistance à la k-B-Consistance est immédiate .

Définition 5.2 (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. $\forall P'' = (\mathcal{V}, \mathcal{D}'', \mathcal{C}), \{P'' \equiv P \wedge (P'' \text{ est } 3\text{-B-Consistant}) \wedge P'' \preceq P \wedge P' \preceq P''\} \Rightarrow D'' = D'$.

```

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
    %  $S$  : Vecteur des  $\Delta$  utilisés pour la réduction des domaines
    %  $n$  : 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  $i = 1 \dots n$  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           $\mathcal{D} \leftarrow 3B\text{-fixpoint}(P, S)$ 
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   $PF \leftarrow false$  : booléen qui indique si le point fixe est atteint
2  while  $\neg PF$  do
3     $PF \leftarrow true$ 
4    for  $i = 1 \dots n$  do : cardinal de  $\mathcal{V}$ 
5       $P_{borne\_inf} = (\mathcal{V}, \{D_{x_1}, \dots, D_{x_{i-1}}, [a, a + S_{x_i}], D_{x_{i+1}}, \dots, D_{x_n}\}, \mathcal{C})$ 
6      if 2B-filtering( $P_{borne\_inf}$ ) =  $P_\emptyset$  then
7         $PF \leftarrow false$ 
8         $D_{x_i} \leftarrow D_{x_i} \setminus [a, a + S_{x_i}]$ 
9         $\mathcal{D} \leftarrow 2B\text{-filtering}(P)$ 
10      $P_{borne\_sup} = (\mathcal{V}, \{D_{x_1}, \dots, D_{x_{i-1}}, [b - S_{x_i}, b], D_{x_{i+1}}, \dots, D_{x_n}\}, \mathcal{C})$ 
11     if 2B-filtering( $P_{borne\_sup}$ ) =  $P_\emptyset$  then
12        $PF \leftarrow false$ 
13        $D_{x_i} \leftarrow D_{x_i} \setminus [b - S_{x_i}, b]$ 
14        $\mathcal{D} \leftarrow 2B\text{-filtering}(P)$ 
15  return  $\mathcal{D}$ 

```

Algorithme = une suite de tentatives de refutation

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

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

La figure 5 présente les premières étapes d'une réduction d'un intervalle $[a, b]$ à partir de la borne gauche : $\Phi_{3B}(P_{D_x \leftarrow [a, a']}) = \emptyset$, $\Phi_{3B}(P_{D_x \leftarrow [a', c]}) \neq \emptyset$, $\Phi_{3B}(P_{D_x \leftarrow [a', d]}) \neq \emptyset$, $\Phi_{3B}(P_{D_x \leftarrow [a', e]}) = \emptyset, \dots$

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.

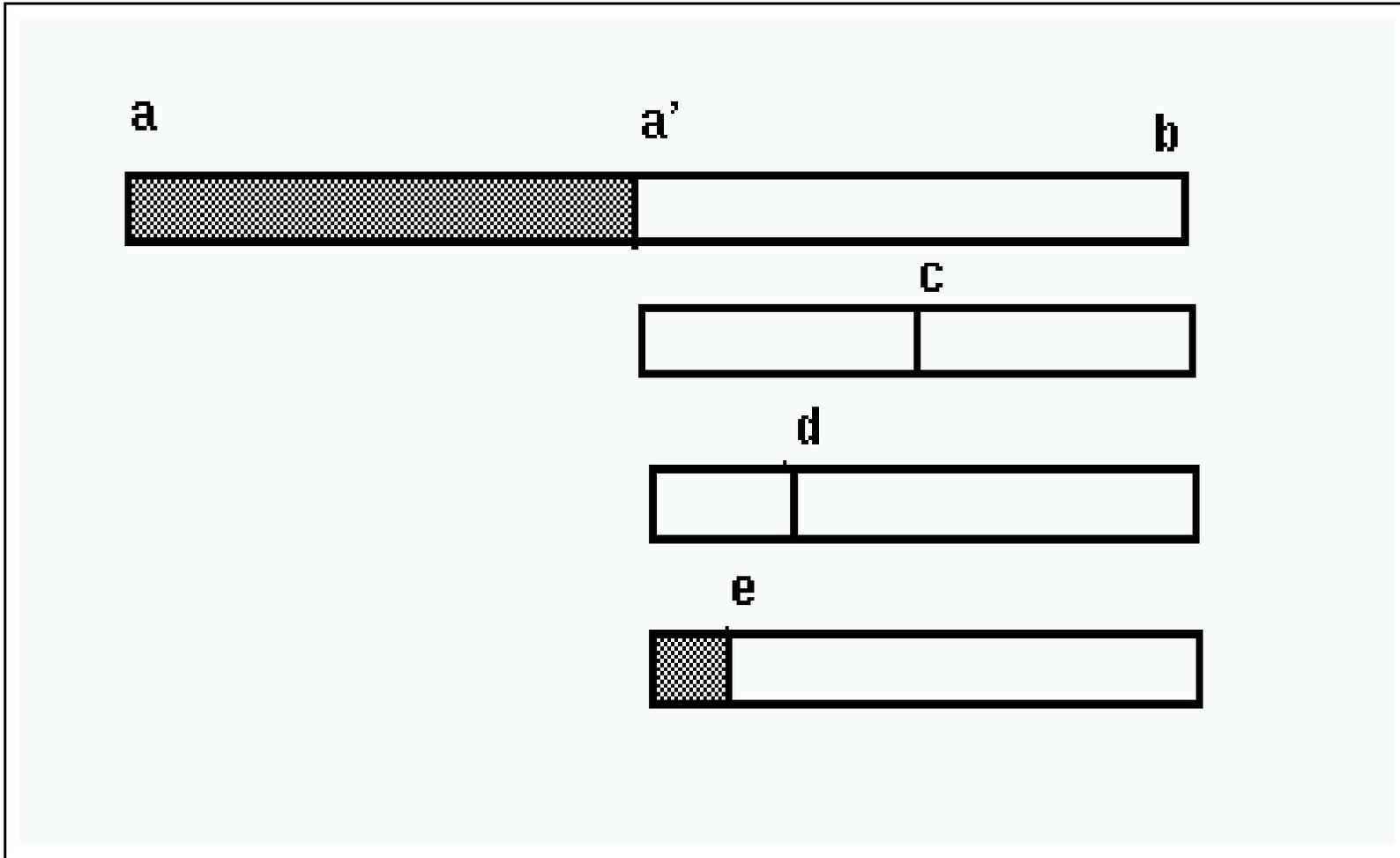


Figure 5: Exemple de réduction de la borne gauche

Exemple 5.2. 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 :

Il est facile de voir que la 3-B-Consistance calcule la consistance totale si le système de contraintes n'a que deux variables. De manière générale, on peut montrer que la k -B-Consistance fournit une procédure pour décider de la satisfiabilité d'un système de $k - 1$ variables.

Une question intéressante est celle de la relation entre $\Phi_{2B}(P)$ et $\Phi_{3B}(P_{decomp})$: les exemples 5.3 et 5.4 montrent respectivement que ni la relation $\Phi_{2B}(P) \preceq \Phi_{3B}(P_{decomp})$, ni la relation $\Phi_{3B}(P_{decomp}) \preceq \Phi_{2B}(P)$ n'est vérifié.

Exemple 5.3. Soit P le CSP défini par

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

$decomp(x_1 + x_1 - 2 \times x_2 = 0) = \{x_1 + x_3 - 2 \times x_2 = 0, x_3 = x_1\}$. P est 2B-Consistant mais P_{decomp} n'est pas 3B-Consistant : En effet, 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 . Dans ce cas le lien entre x_1 et x_3 est préservé et la 3B-Consistance réduit D_{x_2} à $[5, 5]$.

Exemple 5.4. Soit c la contrainte $x_1 * x_2 - x_1 + x_3 - x_1 + x_1 = 0$; $D_{x_1} = [-4, 3]$, $D_{x_2} = [1, 2]$ et $D_{x_3} = [-1, 5]$ les domaines de ses variables.

$decomp(c) = \{x_1 * x_2 - x_4 + x_3 - x_5 + x_6 = 0, x_1 = x_4 = x_5 = x_6\}$. 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 $x_3 = 5$.

Toutefois, $decomp(c)$ est 3B-Consistant. La perte du lien entre les deux occurrences de x_1 interdit la réduction du domaine de x_3 .

6 La Box-Consistance

- ◇ La Box-Consistance est une approximation plus grossière de la consistance d'arc que la 2B-Consistance.
- ◇ La Box-Consistance peut être définie en remplaçant dans la définition de la 2B-Consistance toutes les variables quantifiées existentiellement (sauf une) par l'intervalle qui correspond à leur domaine.
- ◇ 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

6.1 Définition et propriétés de la Box-Consistance

Définition 6.1 (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, \dots, x_k) . c est Box-Consistant si, pour tout x_i dans $\{x_1, \dots, x_k\}$ tel que $D_{x_i} = [a, b]$, les relations ci-dessous sont vérifiées :

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

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

Proposition 4. $\Phi_{2B}(\mathcal{P}) \preceq \Phi_{Box}(\mathcal{P})$ et $\Phi_{2B}(\mathcal{P}) \equiv \Phi_{Box}(\mathcal{P})$ 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 (cf. exemple 6.1).

Exemple 6.1. L'exemple 3.4 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.

Toutefois, la décomposition du système de contraintes amplifie les limites dues au caractère local de la 2B-Consistance. Il en résulte que la 2B-Consistance sur le système décomposé est plus faible que la Box-Consistance sur le système initial :

Proposition 5. $\Phi_{Box}(P) \preceq \Phi_{2B}(P_{decomp})$

Preuve : Pour le calcul des consistances locales, CSP P_{decomp} est une relaxation de P . D'où : $\Phi_{Box}(P) \preceq \Phi_{Box}(P_{decomp})$. 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 $\Phi_{Box}(P) \preceq \Phi_{2B}(P_{decomp}) \diamond$

Exemple 6.2. Soit c la contrainte $x_1 + x_2 - x_1 - x_1 = 0$,
 $D_{x_1} = [-1, 1]$ et $D_{x_2} = [0.5, 1]$ les domaines des variables de c . c
n'est pas Box-Consistant car
 $[-1, -1^+] \oplus [0.5, 1] \ominus [-1, -1^+] \ominus [-1, -1^+] \cap [0, 0]$ est vide. Par
contre, $decomp(c)$ est 2B-Consistant pour D_{x_1} and D_{x_2} .

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

6.2 Calcul de la Box-Consistance

L'algorithme proposé dans pour effectuer un filtrage par Box-Consistance est basé sur l'algorithme générique IN (cf. figure 2) utilisé pour la 2B-Consistance ^a. La fonction $narrow(c, \mathcal{D})$ réduit les domaines des variables de c jusqu'à ce que c soit Box-Consistant.

Schématiquement, 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. Le processus consiste alors à trouver le zéro le plus à gauche et la plus à droite dans toutes ces fonctions univariables sur intervalles qui sont de la forme :

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

La fonction LNAR calcule le zéro le plus à gauche de F_x pour la

^aDans *Numerica*, une heuristique de “domain splitting” est combinée avec cet algorithme de filtrage pour trouver toutes les solutions.

variable x de domaine initial I_x :

- ◇ LNAR réduit d'abord le domaine de l'intervalle I_x avec la fonction $NEWTON(F_x, I_x)$ qui correspond à l'extension aux intervalles de la méthode de Newton
- ◇ Si la fonction $NEWTON(F_x, I_x)$ 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.

Ce processus de division permet d'éviter la recherche d'une "safe starting box" pour Newton (même si F_x n'est pas différentiable, la fonction LNAR va trouver le zéro le plus à gauche grâce au processus de division).

```

function LNAR (IN :  $F_x, I_x$ , return Interval)
  r  $\leftarrow$  right( $I_x$ )
  if  $0 \notin F_x(I_x)$  then return  $\emptyset$ 
  else  $I_x \leftarrow NEWTON(F_x, I_x)$ 
      if  $0 \in F_x([left(I), left(I)^+])$  then return  $[left(I), r]$ 
      else  $SPLIT(I_x, I_1, I_2)$ 
           $L_1 \leftarrow LNAR(F_x, I_1)$ 
          if  $L_1 \neq \emptyset$  then return  $[left(L_1), r]$ 
          else return  $[left(LNAR(F_x, I_2)), r]$ 
      endif
  endif
endif

```

Figure 6: 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 $f(a) \in F(I)$. De plus si F est l'extension naturelle aux intervalles de f , alors $v \in \tilde{u} - \frac{F(\tilde{u})}{F'(I)} = N(F, F', \tilde{u}, I)$. Il en résulte que v est un zéro de f si $v \in N^*(F, F', I)$ avec :

$N^*(F, F', I) = I_n (n \geq 1)$ où :

$$I_0 = I$$

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

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

Figure 7: Rappel du principe de l'opérateur itératif de Newton

7 Bound-Consistance

Comme la 2B-Consistance, la Box-Consistance a été généralisée pour limiter les effets de localité. Schématiquement, la Bound-Consistance applique le principe de la 3B-Consistance à la Box-Consistance : elle vérifie si la Box-Consistance peut être appliquée lorsque le domaine d'une variable est réduit à la valeur d'une de ses bornes.

Définition 7.1 (Bound–Consistance). Soit $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ un CSP et $c \in \mathcal{C}$ une contrainte d'arité k définie sur les variables (x_1, \dots, x_k) . c est Bound–Consistant si pour tout $x_i \in (x_1, \dots, x_k)$ avec

$D_{x_i} = [a, b]$, les propriétés suivantes sont vérifiées :

1. $\Phi_{Box}(C(D_{x_1}, \dots, D_{x_{i-1}}, [a, a^+), D_{x_{i+1}}, \dots, D_{x_k})) \neq P_\emptyset$,
2. $\Phi_{Box}(C(D_{x_1}, \dots, D_{x_{i-1}}, (b^-, b], D_{x_{i+1}}, \dots, D_{x_k})) \neq P_\emptyset$.

Comme $\Phi_{Box}(P) \preceq \Phi_{2B}(P_{decomp})$ on a :

$$\Phi_{Bound}(P) \preceq \Phi_{3B}(P_{decomp}).$$