A Note on Partial Consistencies over Continuous Domains

Hélène Collavizza, François Delobel, and Michel Rueher

Université de Nice–Sophia-Antipolis, I3S
ESSI, 930, route des Colles - B.P. 145
06903 Sophia-Antipolis, France
{helen,delobel,rueher}@essi.fr

Abstract. This paper investigates the relations among different partial consistencies which have been proposed for pruning the domains of the variables in constraint systems over the real numbers. We establish several properties of the filtering achieved by the algorithms based upon these partial consistencies. Especially, we prove that :
1) 2B–Consistency (or Hull consistency) algorithms actually yield a weaker pruning than Box-consistency;
2) 3B–Consistency algorithms perform a stronger pruning than Box-consistency.
This paper also provides an analysis of both the capabilities and the inherent limits of the filtering algorithms which achieve these partial consistencies.

1 Introduction

Partial consistencies are the cornerstones for solving non linear constraints over the real numbers [5, 21, 3, 14, 13, 1, 26].

A partial consistency is a *local* property which is enforced in order to prune the sets of possible values of a variable before searching for isolated solutions. *Arc–consistency* is a partial consistency which has widely been used in constraint solvers over finite domains [17, 25]. However, arc-consistency cannot be enforced when working with real numbers or floating point numbers. Thus, specific local consistencies have been proposed in order to prune intervals of real numbers [6, 11, 14, 1]. This paper investigates the relation between two of them : *2B–Consistency* and *Box–Consistency.* 2B–Consistency (or hull consistency) [1, 2, 5, 13, 14] is an approximation of arc–consistency which only requires the checking of arc–consistency property for each bound of the intervals; Box–Consistency [1, 26] is a coarser approximation of arc–consistency than 2B–Consistency. Roughly speaking, Box–Consistency consists of replacing all existentially quantified variables but one with their intervals in the definition of 2B–Consistency.

To limit the effects of a strictly local processing, higher order extensions of these two consistencies have been introduced :

– *3B–Consistency* [14] is an approximation of path consistency, a higher order extension of arc-consistency [17, 25]. Roughly speaking, 3B–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;
– *Bound-consistency* [26] applies the principle of 3B–Consistency to Box–Consistency : Bound-consistency checks whether Box–Consistency can be enforced when the domain of a variable is reduced to the value of one of its bounds in the whole system.

It has been stated in [16] without proof that :

– 2B–Consistency algorithms actually achieves a weaker filtering than Box–Consistency, especially when a variable occurs more than once in some constraint. This is due to the fact that 2B–Consistency algorithms require a decomposition of the constraints with multiple occurrences of the same variable;
– The filtering achieved by Box–Consistency algorithms is weaker than that computed by 3B–Consistency algorithms.

The main result of this paper is a proof of the above properties. If $\Phi_{cstc}(P)$ is the closure of a constraint system P computed by an algorithm ensuring consistency *cstc*, and P_{decomp} is a decomposition of P allowing 2B-consistency filtering, we then prove that the following relations hold :

$$\Phi_{Bound}(P) \preceq \Phi_{3B}(P_{decomp}) \preceq \Phi_{box}(P) \preceq \Phi_{2B}(P_{decomp})$$

This paper also provides an analysis of both the capabilities and the limits of the filtering algorithms which achieve these partial consistencies. We pay special attention to their ability to handle the so-called dependency problem[1].

Layout of the paper

Section 2 reviews some basic concepts of interval analysis required in the rest of the paper. Section 3 is devoted to the analysis of 2B–Consistency. Features and properties of Box–Consistency are the focus of Section 4. 3B–Consistency and Bound–Consistency are introduced in section 5. Section 6 mentions efficiency and precision issues.

2 Interval constraint solving

This section recalls some basics of interval analysis [1, 2, 11] and formally defines a constraint system over intervals of real numbers.

[1] The so-called *dependency problem* [9] is a fundamental problem in interval analysis : when a given variable occurs more than once in an interval computation it is treated as a different variable. For instance, $X \otimes SIN(X)$ is the same as $X \otimes SIN(Y)$ with Y equal to but independent of X. Suppose $X = [-5, 5]$ and $\otimes$ and SIN are respectively interval extensions of the multiplication over the reals and function *sinus*, then the value of $X \otimes SIN(X)$ is not $[-4.8, 1.9]$ but $[-5, 5]$. As shown by this example, the dependency problem often entails a widening of the computed intervals.

2.1 Notations

Throughout this paper, the following notations, possibly subscripted, are used :

- x, y, z denote variables over the reals; X, Y, Z denote variables over the intervals;
- $\mathcal{R}^{\infty} = \mathcal{R} \cup \{\Leftrightarrow\infty, +\infty\}$ denotes the set of real numbers augmented with the two infinity symbols. $\overline{\mathbb{F}}$ denotes a finite subset of $\mathcal{R}^{\infty}$ containing $\{\Leftrightarrow\infty, +\infty\}$[2];
- u, v, r denote constants in $\mathcal{R}$; a, b denote constants in $\overline{\mathbb{F}}$; a^{+} (resp. a^{-}) corresponds to the smallest (resp. largest) number of $\overline{\mathbb{F}}$ strictly greater (resp. smaller) than a;
- f, g denote functions over the reals; F,G denote functions over the intervals;
- c denotes a constraint over the reals, C denotes a relation over the intervals; $Var(c)$ denotes the variables occuring in c;
- $\Phi_{cstc}(P)$ is the closure of P by consistency $cstc$ (where $cstc$ is $2B$, Box, $3B$, $Bound$).

2.2 Interval analysis

Definition 1 (Interval). *An interval $[a, b]$ with $a, b \in \overline{\mathbb{F}}$ is the set of real numbers $\{r \in \mathcal{R} \mid a \leq r \leq b\}$.*

Let r be a real number. $\tilde{r}$ denotes the smallest[3] (w.r.t. inclusion) interval of $\overline{\mathbb{F}}$ containing r. $\mathcal{I}$ denotes the set of intervals and is ordered by set inclusion. $\mathcal{U}(\mathcal{I})$ denotes the set of unions of intervals.

Definition 2 (Set Extension). *Let S be a subset of $\mathcal{R}$. The approximation of S —denoted $\Box S$— is the smallest interval I such that $S \subseteq I$.*

Definition 3 (Interval Extension [19, 9]).
• *An interval function $F : \mathcal{I}^n \to \mathcal{I}$ is an interval extension of function $f : \mathcal{R}^n \to \mathcal{R}$ iff :*

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

• *An interval relation $C : \mathcal{I}^n \to \mathcal{B}ool$ is an interval extension of relation $c : \mathcal{R}^n \to \mathcal{B}ool$ iff :*

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

2 Practically speaking, $\overline{\mathbb{F}}$ corresponds to the set of floating-point numbers used in the implementation of non linear constraint solvers.

3 The term smallest subset (w.r.t. inclusion) must be understood here according to the precision of floating-point operations. In the rest of the paper, we consider —as in [13, 1]— that results of floating-point operations are outward-rounded to preserve the correctness of the computation. However, we assume that the largest computing error when computing a bound of a variable of the initial constraint system is always smaller than one float. This hypothesis may require the use of big floats [4] when computing intermediate results. Consequences of the relaxation of this hypothesis are examined in details in section 6.

For instance, the interval relation $\doteq$ defined as $I_1 \doteq I_2 \Leftrightarrow (I_1 \cap I_2) \neq \emptyset$ is an interval extension of the equality relation on real numbers.

Definition 4 (Natural Interval Extension [19, 20]). *An interval function $F\ :\mathcal{I}^n \rightarrow \mathcal{I}$ is the natural interval extension of $f\ :\mathcal{R}^n \rightarrow \mathcal{R}$ if F is the interval extension of f obtained by replacing in f each constant k with its natural interval extension $\tilde{k}$, each variable with an interval variable and each arithmetic operation with its optimal interval extension [19].*

Optimal interval extensions have been introduced by [19] for the basic interval operations. For instance, let $\odot$ be an operator in $\{+, \Leftrightarrow, \times, /\}$, and $[a,b]\odot[c,d] = \{x \odot y$ such that $a \leq x \leq b$ and $c \leq y \leq d\}$ then the optimal interval extensions for these four operations are :

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

In the rest of this paper $\oplus, \ominus, \otimes, \oslash$ denote the optimal interval extensions of $+, \Leftrightarrow, \times, /$.

Optimal interval extensions can be defined in a similar way for *power* and almost all other elementary functions and relations [20].

Example 1. Let $f(x) = x + x^2 + 3$ be a function over the reals. Its natural interval extension is defined by $X \oplus X \nabla 2 \oplus \tilde{3}$ where ∇ is the optimal interval extensions of *power*.

We now recall a fundamental result of interval analysis with many consequences on efficiency and precision of interval constraint solving methods.

Proposition 1. *[19] Let $F\ :\mathcal{I}^n \rightarrow \mathcal{I}$ be the natural interval extension of $f\ :\mathcal{R}^n \rightarrow \mathcal{R}$ and let $f_{sol} = \Box\{f(v_1,\ldots,v_n) \mid v_1 \in I_1,\ldots,v_n \in I_n\}$. If each x_i occurs only once in f then $f_{sol} = F(I_1,\ldots,I_n)$ else $f_{sol} \subseteq F(I_1,\ldots,I_n)$.*

This result can be extended to k-ary relations over $\mathcal{R}^n$:

Proposition 2. *Let $C\ :\mathcal{I}^n \rightarrow \mathcal{B}ool$ be the natural extension of an equation $c\ :\mathcal{R}^n \rightarrow \mathcal{B}ool$ then, if each x_i occurs only once in c, then :*

$$C(I_1,\ldots,I_n) \Leftrightarrow (\exists v_1 \in I_1,\ldots,\exists v_n \in I_n \mid c(v_1,\ldots,v_n)).$$

Proof : We consider here constraints of the form $c\ : f(x_1,\ldots,x_n) = 0$. Thus, c holds if and only if the relation $(\exists v_1 \in I_{x_1},\ldots,\exists v_n \in I_{x_n} \mid f(v_1,\ldots,v_n) = 0)$ holds. If no variable has multiple occurrences in c, then, by proposition 1, we have : $(\exists v_1 \in I_{x_1},\ldots,\exists v_n \in I_{x_n} \mid f(v_1,\ldots,v_n) = 0) \Leftrightarrow F(X_1,\ldots,X_n) \doteq [0,0]$, where $F(X_1,\ldots,X_n)$ is the natural interval extension of $f(x_1,\ldots,x_n)$ according to definition 4. So, property 2 holds since $F(X_1, \ldots, X_n) \doteq [0,0]$ is the interval extension of $f(x_1, \ldots, x_n) = 0$ $\diamond$

It follows that only the sub-distributive law holds in interval analysis[4]:

$$I \otimes (J \oplus K) \subseteq I \otimes J \oplus I \otimes K \qquad .$$

2.3 Interval constraint system

A k-ary constraint c is a relation over the reals. C denotes its natural interval extension.

Definition 5 (CSP).
A CSP [17] is a triple $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ where $\mathcal{X} = \{x_1, \ldots, x_n\}$ denotes a set of variables, $\mathcal{D} = \{D_{x_1}, \ldots, D_{x_n}\}$ denotes a set of domains, D_{x_i} being the interval containing all acceptable values for x_i, and $\mathcal{C} = \{c_1, \ldots, c_m\}$ denotes a set of constraints.

$P_\emptyset$ denotes an empty CSP, i.e., a CSP with at least one empty domain. $\mathcal{D}' \subseteq \mathcal{D}$ means $D'_{x_i} \subseteq D_{x_i}$ for all $i \in 1..n$. We define a CSP $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ to be smaller than a CSP $P' = (\mathcal{X}, \mathcal{D}', \mathcal{C})$ if $\mathcal{D}' \subseteq \mathcal{D}$. We note $P \preceq P'$ this relation. By convention $P_\emptyset$ is the smallest CSP.

3 2B–Consistency

Most of the CLP systems over intervals (e.g., [21, 22, 2, 23]) compute an approximation of arc-consistency [17] called 2B–Consistency (or Hull consistency). In this section, we give the definition of 2B–consistency and explain why its computation requires a relaxation of the constraint system.

3.1 Definitions

2B–Consistency [14] states a local property on the bounds of the domains of a variable at a single constraint level. Roughly speaking, a constraint c is 2B–Consistent if for any variable x there exist values in the domains of all other variables which satisfy c when x is fixed to any bound of D_x.

Definition 6 (2B–Consistency). *Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a CSP and $c \in \mathcal{C}$ a k-ary constraint over the variables $(x_1, \ldots, x_k)$. c is 2B–Consistent iff :*
$\forall i, D_{x_i} = \Box\{v_i \in D_{x_i} \mid \exists v_1 \in D_{x_1}, \ldots, \exists v_{i-1} \in D_{x_i-1}, \exists v_{i+1} \in D_{x_i+1}, \ldots, \exists v_k \in D_{x_k}$ *such that* $c(v_1, \ldots, v_{i-1}, v_i, v_{i+1} \ldots, v_k)$ *holds* $\}$.
A CSP is 2B–Consistent iff all its constraints are 2B–Consistent.

2B–Consistency is weaker than arc–consistency.

Example 2. Let $P_1 = (\{x_1, x_2\}, \{D_{x_1} = [1,4],\ D_{x_2} = [\Leftrightarrow 2, 2]\}, \{x_1 = x_2^2\})$ be a CSP. P_1 is 2B–Consistent but not arc–Consistent since there is no value in D_{x_1} which satisfies the constraint when $x_2 = 0$.

[4] Of course, commutative and associative laws are preserved.

Proposition 3. *Let $(\mathcal{X},\mathcal{D},\mathcal{C})$ be a CSP such that no variable occurs more than once in any constraint of $\mathcal{C}$. Let $c \in \mathcal{C}$ be a k-ary constraint over the variables $(x_1,\ldots,x_k)$. c is 2B–Consistent iff for all x_i in $\{x_1,\ldots,x_k\}$ such that $D_{x_i} = [a,b]$ the following relations hold :*

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

where $[a,a^+)$ and $(b^-,b]$ denote half–open intervals.

Proof : Assume that both $C(D_{x_1},\ldots,D_{x_{i-1}},[a,a^+),D_{x_{i+1}},\ldots,D_{x_k})$ and $C(D_{x_1},\ldots,D_{x_{i-1}},(b^-,b],D_{x_{i+1}},\ldots,D_{x_k})$ hold. By proposition 2 we have :

1. $\exists x_1 \in D_{x_1},\ldots,\exists x_{i-1} \in D_{x_i-1},\exists x_i \in [a,a^+),\exists x_{i+1} \in D_{x_i+1},\ldots,\exists x_k \in D_{x_k}$ such that $c(x_1,\ldots,x_{i-1},x_i,x_{i+1},\ldots,x_k)$ holds, and
2. $\exists x_1 \in D_{x_1},\ldots,\exists x_{i-1} \in D_{x_i-1},\exists x_i \in (b^-,b],\exists x_{i+1} \in D_{x_i+1},\ldots,\exists x_k \in D_{x_k}$ such that $C(x_1,\ldots,x_{i-1},x_i,x_{i+1},\ldots,x_k)$ holds.

Thus, $D_{x_i} = \Box\{x_i \in D_{x_i} \mid \exists v_1 \in D_{x_1},\ldots,\exists v_{i-1} \in D_{x_i-1},\exists v_{i+1} \in D_{x_i+1},\ldots,$ $\exists v_k \in D_{x_k}$ such that $c(v_1,\ldots,x_i,\ldots,v_k)$ holds $\}$.
The counterpart results from the definition of 2B–Consistency.◇

Definition 7 (Closure by 2B–Consistency). *[14] Closure by 2B-Consistency of a CSP $P = (\mathcal{X},\mathcal{D},\mathcal{C})$ is the CSP $P' = (\mathcal{X},\mathcal{D}',\mathcal{C})$ such that :*

– *P and P' have the same solutions;*
– *P' is 2B–Consistent;*
– *$\mathcal{D}' \subseteq \mathcal{D}$ and domains in $\mathcal{D}'$ are the largest ones for which P' is 2B–Consistent.*

Closure by 2B–Consistency of a CSP always exists and is unique [15].

3.2 Computing 2B–Consistency

2B–Consistency is enforced by narrowing the domains of the variables. Using the above notations, the scheme of the standard interval narrowing algorithm — derived from AC3– can be written down as in figure 1. `IN` implements the computation of the closure by 2B-consistency of a CSP $P = (\mathcal{X},\mathcal{D},\mathcal{C})$. $narrow(c,\mathcal{D})$ is a function which prunes the domains of variables $Var(c)$ until c is 2B–consistent.

The approximation of the projection functions is the basic tool for narrowing domains in $narrow(c,\mathcal{D})$.

Let c be a k-ary constraint over $(x_1,\ldots,x_k)$, and $< I_1,\ldots,I_k >\in \mathcal{I}^k$: for each i in $1..k$, $\pi_i(c, I_1 \times \ldots \times I_k)$ denotes the projection over x_i of the solutions of c in the part of the space delimited by $I_1 \times \ldots \times I_k$.

Definition 8 (projection of a constraint).
$\pi_i(c, I_1 \times \ldots \times I_k)$: $(\mathcal{C},\mathcal{I}^k) \to \mathcal{U}(\mathcal{I})$ is the projection of c on x_i iff :
$\pi_i(c, I_1 \times \ldots \times I_k) = \{v_i \in I_i \mid \exists < v_1,\ldots,v_{i-1},v_{i+1},\ldots,v_k >\in I_1 \times \ldots \times I_{i-1} \times I_{i+1},\ldots \times I_k$ such that $c(v_1,\ldots,v_i,\ldots,v_k)$ holds$\}$

```
IN(in $\mathcal{C}$, inout $\vec{\mathcal{D}}$)
    Queue $\leftarrow \mathcal{C}$ ;
    while Queue $\neq \emptyset$
        $c \leftarrow$ POP Queue;
        $\mathcal{D}' \leftarrow$ narrow$(c, \mathcal{D})$;
        if $\mathcal{D}' \neq \mathcal{D}$ then $\mathcal{D} \leftarrow \mathcal{D}'$;
                        Queue $\leftarrow$Queue $\cup \{c' \in \mathcal{C} \mid Var(c) \cup Var(c') \neq \emptyset\}$
        endif
    endwhile
```

Fig. 1. Algorithm `IN`

Definition 9 (approximation of the projection).
$AP_i(c, I_1 \times \ldots \times I_k) : (\mathcal{C}, \mathcal{I}^k) \rightarrow \mathcal{I}$ is an approximation of $\pi_i(c, I_1 \times \ldots \times I_k)$ iff $AP_i(c, I_1 \times \ldots \times I_k) = \Box\, \pi_i(c, I_1 \times \ldots \times I_k) = [Min\ \pi_i(c, I_1 \times \ldots \times I_k), Max\ \pi_i(c, I_1 \times \ldots \times I_k)]$.
In other words $AP_i(c, I_1 \times \ldots \times I_k)$ is the smallest interval encompassing projection $\pi_i(c, I_1 \times \ldots \times I_k)$.

The following proposition trivially holds :

Proposition 4. *Constraint c is 2B–Consistent on $< I_1, \ldots, I_k >$ iff for all i in $\{1, \ldots, k\}$, $I_i = AP_i(c, I_1 \times \ldots \times I_k)$.*

In the general case, AP_i cannot be computed directly because it is difficult to define functions Min and Max, especially when c is not monotonic. For instance, if variable x has multiple occurrences in c, defining these functions would require x to be isolated[5]. Since such a symbolic transformation is not always possible, this problem is usually solved by decomposing the constraint system into a set of basic constraints for which the AP_i can easily be computed [16]. Basic constraints are generated syntactically by introducing new variables.

Definition 10 (decomposition of a constraint system).
Let $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a CSP and $c \in \mathcal{C}$ a constraint. We define $\mathcal{M}_c \subseteq \mathcal{X}$ as the set of variables having multiple occurrences in c. decomp(c) is the set of constraints obtained by substituting in c each occurrence of variables $x \in \mathcal{M}_c$ by a new variable y with domain $D_y = D_x$ and by adding a constraint $x = y$. $New_{(x,c)}$ is the set of new variables introduced to remove multiple occurrences of variable x in c, $\mathcal{X}_{New} = \bigcup\{New_{(x,c)} \mid x \in \mathcal{X}$ and $c \in \mathcal{C}\}$. P_{decomp} is the CSP $(\mathcal{X}', \mathcal{D}', \mathcal{C}')$ where $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}_{New}$, $\mathcal{D}' = \mathcal{D} \cup \{D_y \mid y \in \mathcal{X}_{New}\}$ and $\mathcal{C}' = \{decomp(c) \mid c \in \mathcal{C}\}$.

[5] B. Faltings [6] has recently introduced a new method for computing the projection without defining projection function. However, this method requires a complex analysis of constraints in order to find extrema.

Decomposition does not change the semantics of the constraint system : P and P_{decomp} have the same solutions since P_{decomp} just results from a rewriting[6] of P. However, a *local* consistency like Arc–Consistency is not preserved by such a rewriting. Thus, P_{decomp} is a *relaxation* of P when computing an approximation of Arc–Consistency.

Example 3 (decomposition of the constraint system). Let $c\ : x_1 + x_2 \Leftrightarrow x_1 = 0$ be a constraint and $D_{x_1} = [\Leftrightarrow 1, 1]$, $D_{x_2} = [0, 1]$ the domains of x_1 and x_2. Since x_1 appears twice in c, its second occurrence will be replaced with a new variable x_3 : $decomp(c) = \{x_1 + x_2 \Leftrightarrow x_3 = 0, x_1 = x_3\}$.

In this new constraint system, each projection can easily be computed with interval arithmetic. For instance, $AP_1(x_1 + x_2 \Leftrightarrow x_3 = 0, D_{x_1}, D_{x_2}, D_{x_3})$ is $D_{x_1} \cap (D_{x_3} \ominus D_{x_2})$.

However, this decomposition increases the locality problem : the first constraint is checked independently of the second one and so x_1 and x_3 can take distinct values. More specifically, the initial constraint c is not 2B–Consistent since there is no value of x_1 which satisfies c when $x_2 = 1$. On the contrary, *decomp*(c) is 2B–Consistent since the values $x_1 = \Leftrightarrow 1$ and $x_3 = 0$ satisfy $x_1 + x_2 \Leftrightarrow x_3 = 0$ when $x_2 = 1$. On the initial constraint, 2B–Consistency reduces D_{x_2} to [0,0] while it yields $D_{x_1} = [\Leftrightarrow 1, 1], D_{x_2} = [0, 1]$ for *decomp*(c).

Remark 1. Example 3 like almost all other examples in this paper trivially can be simplified. However, the reader can more easily check partial consistencies on such examples than on non–linear constraints where the same problems occur.

4 Box–Consistency

Box–Consistency [1, 26] is a coarser approximation of arc-consistency than 2B–Consistency. It mainly consists of replacing every existentially quantified variable but one with its interval in the definition of the 2B–Consistency. Thus, Box–Consistency generates a system of univariate functions which can be tackled by numerical methods such as Newton. Contrary to 2B–Consistency, Box–Consistency does not require any constraint decomposition and thus does not amplify the locality problem. Moreover, Box–Consistency can tackle some dependency problems when each constraint of a CSP contains only one variable which has multiple occurrences.

4.1 Definition and properties of Box–Consistency

Definition 11 (Box–Consistency). *Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a CSP and $c \in \mathcal{C}$ a k-ary constraint over the variables $(x_1, \ldots, x_k)$. c is Box–Consistent if, for all x_i*

[6] In practice, c is decomposed into binary and ternary constraints for which projection functions are straightforward to compute. Since there are no multiple occurrences in *decomp*(c) and interval calculus is associative, this binary and ternary constraint system has the same solutions as P_{decomp}.

in $\{x_1, \ldots, x_k\}$ *such that* $D_{x_i} = [a, b]$*, the following relations hold :*
1. $C(D_{x_1}, \ldots, D_{x_{i-1}}, [a, a^+), D_{x_{i+1}}, \ldots, D_{x_k})$*,*
2. $C(D_{x_1}, \ldots, D_{x_{i-1}}, (b^-, b], D_{x_{i+1}}, \ldots, D_{x_k})$*.*

Closure by Box–Consistency of P is defined similarly to closure by 2B–Consistency of P, and is denoted by $\Phi_{Box}(P)$.

Proposition 5. $\Phi_{2B}(P) \preceq \Phi_{Box}(P)$ *and* $\Phi_{2B}(P) \equiv \Phi_{Box}(P)$ *when no variable occurs more than once in the constraints of* $\mathcal{C}$*.*

Proof : From the definitions of 2B–Consistency, Box–Consistency and interval extension of a relation, it results that $\Phi_{2B}(P) \preceq \Phi_{Box}(P)$. By proposition 2 the equivalence holds when no variable occurs more than once in the constraints of $\mathcal{C}$. $\diamond$

It follows that any CSP which is 2B–Consistent is also Box–Consistent. On the contrary a CSP which is Box–Consistent may not be 2B–Consistent (see example 4).

Example 4. Example 3 is not 2B–Consistent for x_2 but it is Box–Consistent for x_2 since $([⇔1, 1] \oplus [0, 0^+] \ominus [⇔1, 1]) \cap [0, 0]$ and $([⇔1, 1] \oplus [1^-, 1] \ominus [⇔1, 1]) \cap [0, 0]$ are non-empty.

The decomposition of a constraint system amplifies the limit due to the local scope of 2B–Consistency. As a consequence, 2B–Consistency on the decomposed system yields a weaker filtering than Box–Consistency on the initial system :

Proposition 6. $\Phi_{Box}(P) \preceq \Phi_{2B}(P_{decomp})$

Proof : The different occurrences of the same variable are connected by the existential quantifier as stated in the definition of the 2B–Consistency. However, the decomposition step breaks down the links among these different occurrences and generates a CSP P_{decomp} which is a relaxation of P for the computation of a local consistency. It follows that Φ_{Box}(P) $\preceq \Phi_{Box}(P_{decomp})$. By proposition 5 we have : $\Phi_{Box}(P_{decomp}) \equiv \Phi_{2B}(P_{decomp})$, and thus $\Phi_{Box}(P) \preceq \Phi_{2B}(P_{decomp})\diamond$

Example 5. Let c be the constraint $x_1 + x_2 ⇔ x_1 ⇔ x_1 = 0$ and $D_{x_1} = [⇔1, 1]$ and $D_{x_2} = [0.5, 1]$ the domains of its variables. c is not Box–Consistent since $[⇔1, ⇔1^+] \oplus [0.5, 1] \ominus [⇔1, ⇔1^+] \ominus [⇔1, ⇔1^+] \cap [0, 0]$ is empty. But *decomp*(c) is 2B–Consistent for D_{x_1} and D_{x_2}.

Box–Consistency can tackle some dependency problems in a constraint c which contains only one variable occuring more than once. More precisely, Box–Consistency enables to reduce domain D_x if variable x occurs more than once in c and if D_x contains inconsistent values. For instance, in example 5, filtering by Box–consistency reduces the domain of x_1 because value ⇔1 of D_{x_1} has no

support in domain D_{x_2}.
However, Box–Consistency may fail to handle the dependency problem when the inconsistent values of constraint c are in the domain of variable x_i while a variable x_j $(j \neq i)$ occurs more than once in c. For instance, in example 3, value 1 of D_{x_2} has no support in domain D_{x_1} but Box–Consistency fails to detect the inconsistency because $[\Leftrightarrow 1, 1] \oplus [1^-, 1] \ominus [\Leftrightarrow 1, 1] \cap [0, 0]$ holds.

4.2 Computing Box–Consistency

The Box–Consistency filtering algorithm proposed in [1, 26, 27] is based on an iterative narrowing operation using the interval extension of the Newton method. Computing Box–Consistency follows the generic algorithm `IN` (see figure 1) used for computing 2B–Consistency[7]. The function $narrow(c, \mathcal{D})$ prunes the domains of the variables of c until c is Box–consistent. Roughly speaking, for each variable x of constraint c, an interval univariate function F_x is generated from c by replacing all variables but x with their intervals. The narrowing process consists in finding the leftmost and rightmost zeros of F_x. Figure 2 shows function `LNAR` which computes the leftmost zero of F_x for initial domain I_x of variable x (this procedure is given in [27]).

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

Fig. 2. Function `LNAR`

Function `LNAR` first prunes interval I_x with function `NEWTON` which is an interval version of the classical Newton method. However, depending on the value of I_x, Newton may not reduce I_x enough to make I_x Box–Consistent. So, a split step is applied in order to ensure that the left bound of I_x is actually a zero. Function `SPLIT` divides interval I in two intervals I_1 and I_2, I_1 being the left part of the interval. The splitting process avoids the problem of finding

[7] In [26], a branch process is combined with this filtering algorithm in order to find all the isolated solutions

a safe starting box for Newton (see [12]). As mentioned in [27], even if F_x is not differentiable, the function `LNAR` may find the leftmost zero thanks to the splitting process (in this case, the call to function `NEWTON` is just ignored). Notice that Box–consistency can be computed in such a way because it is defined on interval constraints whereas the existential quantifiers in the definition of 2B–consistency require the use of projection functions.

5 3B–Consistency and Bound–Consistency

2B–Consistency is only a partial consistency which is often too weak for computing an accurate approximation of the set of solutions of a CSP. In the same way that arc-consistency has been generalized to higher consistencies (e.g., path consistency [17]), 2B–Consistency and Box–Consistency can be generalized to higher order consistencies [14].

5.1 3B–Consistency

Definition 12 (3B–Consistency). *[14] Let $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a CSP and x a variable of $\mathcal{X}$ with domain $[a, b]$. Let also be :*

- *$P_{D_x^1 \leftarrow [a,a^+)}$ the CSP derived from P by substituting D_x in $\mathcal{D}$ by $D_x^1 = [a, a^+)$;*
- *$P_{D_x^2 \leftarrow (b^-,b]}$ the CSP derived from P by substituting D_x in $\mathcal{D}$ by $D_x^2 = (b^-, b]$.*

D_x is 3B–Consistent iff $\Phi_{2B}(P_{D_x^1}) \neq P_\emptyset$ and $\Phi_{2B}(P_{D_x^2}) \neq P_\emptyset$.
A CSP is 3B–Consistent iff all its domains are 3B–Consistent.

It results from this definition that any CSP which is 3B–Consistent is also 2B–Consistent ([14]). The generalization of the 3B–Consistency to kB–Consistency is straightforward and is given in [14, 16]. Closure by kB–Consistency of P is defined in a similar way to closure by 2B–Consistency of P, and is denoted by $\Phi_{kB}(P)$.

Proposition 7. *Let $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a CSP. If P_{decomp} is 3B–Consistent then P is Box–Consistent.*

Proof : Since Box-consistency is a local consistency we just need to show that the property holds for a single constraint.
Assume c is a constraint over $(x_1, ..., x_k)$, x is one of the variables occuring more than once in c, $D_x = [a, b]$ and $New_{(x,c)} = (x_{k+1}, \dots, x_{k+m})$ is the set of variables introduced for replacing the multiple occurrences of x in c. Suppose that P_{decomp} is 3B–Consistent for D_x .
Consider P_1 the CSP derived from P_{decomp} by reducing domain D_x to $[a, a^+)$. P_1 is 2B–Consistent for D_x and thus the domain of all variables in $new_{(x,c)}$ is reduced to $[a, a^+)$; this is due to the equality constraints added when introducing new variables. From proposition 3, it results that the following relation holds:
$C'(D_{x_1}, \dots, D_{x_{i-1}}, [a, a^+), D_{x_{i+1}}, \dots, D_{x_k}, [a, a^+), \dots, [a, a^+), D_{x_{k+m}}, \dots, D_{x_n})$
C' is the very same syntactical expression as C up to variable renaming.

$(D_{x_{k+m}}, \ldots, D_{x_n})$ are the domains of the variables introduced for replacing the multiple occurrences of $\mathcal{M}_c \setminus \{x\}$. As the natural interval extension of a constraint is defined over the intervals corresponding to the domains of the variables, relation $C(D_{x_1}, \ldots, D_{x_{i-1}}, [a, a^+), D_{x_{i+1}}, \ldots, D_{x_k})$ holds too.

The same reasoning can be applied when x is replaced with its upper bound $(b^-, b]$. So we conclude that D_x is also Box–Consistent. $\diamond$

Example 6. [8] Let $\mathcal{C} = \{x_1 + x_2 = 100, x_1 \Leftrightarrow x_2 = 0\}$ and $\mathcal{D} = \{[0, 100], [0, 100]\}$ be the constraints and domains of a given CSP P. $\Phi_{3B}(P_{decomp})$ reduces the domains of x_1 and x_2 to the interval [50,50] whereas Φ_{Box}(P) does not achieve any pruning (P is Box–Consistent).

The following proposition is a direct consequence of proposition 7 :

Proposition 8. *$\Phi_{3B}(P_{decomp}) \preceq \Phi_{Box}(P)$.*

Thus, 3B–Consistency allows to tackle at least the same dependency problems as Box–consistency. However, 3B–Consistency is not effective enough to tackle the dependency problem in general (see example 7).

Example 7. Let c_0 be the constraint $x_1 * x_2 \Leftrightarrow x_1 + x_3 \Leftrightarrow x_1 + x_1 = 0$ and $D_{x_1} = [\Leftrightarrow 4, 3], D_{x_2} = [1, 2]$ and $D_{x_3} = [\Leftrightarrow 1, 5]$ the domains of its variables. *decomp*$(c) = \{x_1 * x_2 \Leftrightarrow x_4 + x_3 \Leftrightarrow x_5 + x_6 = 0, x_1 = x_4 = x_5 = x_6\}$. c is not 2B–Consistent since there are no values in D_{x_1} and D_{x_2} which verify the relation when $x_3 = 5$. However, *decomp*(c) is 3B–Consistent. Indeed, the loss of the link between the two occurrences of x_1 prevents the pruning of x_3.

A question which naturally arises is that of the relation which holds between Φ_{2B}(P) and $\Phi_{3B}(P_{decomp})$: example 8 shows that Φ_{2B}(P) $\preceq \Phi_{3B}(P_{decomp})$ does not hold and example 7 shows that $\Phi_{3B}(P_{decomp}) \preceq \Phi_{2B}(P)$ does not hold, even if only one variable occurs more than once in each constraint of P. It follows that no order relation between $\Phi_{3B}(P_{decomp})$ and Φ_{2B}(P) can be exhibited.

Example 8. Let P be a CSP defined by $\mathcal{C} = \{x_1 + x_2 = 10; x_1 + x_1 \Leftrightarrow 2 \times x_2 = 0\}$, $D_{x_1} = D_{x_2} = [\Leftrightarrow 10, 10]$. *decomp*$(x_1 + x_1 \Leftrightarrow 2 \times x_2 = 0) = \{x_1 + x_3 \Leftrightarrow 2 \times x_2 = 0, x_3 = x_1\}$. P is 2B–Consistent but P_{decomp} is not 3B–Consistent : Indeed, when x_1 is fixed to 10, $\Phi_{2B}(P_{D_{x_1} \leftarrow [10^-, 10]}) = P_\emptyset$ since D_{x_2} is reduced to $\emptyset$. In this case, the link between x_1 and x_3 is preserved and 3B–Consistency reduces D_{x_2} to [5,5].

[8] One may also notice that :

– Neither the initial constraint nor the decomposed system in example 3 are 3B–Consistent but both of them are Box–Consistent;
– Constraint c in example 7 is not 2B–Consistent but it is Box–Consistent

5.2 Bound–Consistency

Bound-consistency was suggested in [16] and was formally defined in [26]. Informally speaking, Bound-consistency applies the principle of 3B–Consistency to Box–Consistency : it checks whether Box–Consistency can be enforced when the domain of a variable is reduced to the value of one of its bounds in the whole system.

Definition 13 (Bound–Consistency). *Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a CSP and $c \in \mathcal{C}$ a k-ary constraint over the variables $(x_1, \ldots, x_k)$. c is Bound–Consistent if for all $x_i \in (x_1, \ldots, x_k)$ such that $D_{x_i} = [a, b]$, the following relations hold :*
1. $\Phi_{Box}(C(D_{x_1}, \ldots, D_{x_{i-1}}, [a, a^+), D_{x_{i+1}}, \ldots, D_{x_k})) \neq P_\emptyset$,
2. $\Phi_{Box}(C(D_{x_1}, \ldots, D_{x_{i-1}}, (b^-, b], D_{x_{i+1}}, \ldots, D_{x_k})) \neq P_\emptyset$.

Since $\Phi_{Box}(\mathrm{P}) \preceq \Phi_{2B}(P_{decomp})$ it is trivial to show that $\Phi_{Bound}(P) \preceq \Phi_{3B}(P_{decomp})$. Bound–Consistency achieves the same pruning as 3B–Consistency when applied to examples 6 and 3.

6 Discussion

This paper has investigated the relations among 2B–Consistency, 3B–Consistency, Box–Consistency and Bound–Consistency. The advantage of Box–Consistency is due to the fact that it generates univariate functions which can be tackled by numerical methods such as Newton, and which do not require any constraint decomposition. On the other hand, 2B–Consistency algorithms require a decomposition of the constraints with multiple occurrences of the same variable. This decomposition increases the limitations due to the local nature of 2B–Consistency

As expected, higher consistencies — e.g., 3B–Consistency and Bound–Consistency — can reduce the drawbacks due to the local scope of the inconsistency detection.

Experimental results of `Numerica` and `Newton` are very impressive [1, 27]. However, other experimental results [8] show that Box–Consistency is not always better than 2B–Consistency, and that combining these two kinds of partial consistencies is clearly a promising approach. It is nevertheless important to notice that the precision of the computations is a very critical issue when one is comparing different filtering algorithms.

Up until now, we assumed that the largest computing error when computing a bound of a variable of the initial system is always smaller than one float. However, as this hypothesis may entail a significant computing cost overhead, it is relaxed in most implementations. Moreover, for efficiency reasons kB–Consistency and Box–Consistency algorithms usually stop the propagation process before normal termination : when the restriction of a domain is less than ϵ —relative or absolute— no pruning is achieved. kB(w)–Consistency and Box(w)–Consistency have been introduced to characterize such filtering algorithms [14, 8]

Formally, 2B(w)–Consistency can be defined in the following way :

Definition 14. *Let $(\mathcal{X},\mathcal{D},\mathcal{C})$ be a CSP, $x \in \mathcal{X}$, $D_x = [a,b]$, w a positive integer. D_x is 2–B(w)–consistent if for all $C(x,x_1,\ldots,x_k)$ in $\mathcal{C}$, the following relations hold :*
1) $\exists v \in [a, a^{+w}]$, $\exists v_1,\ldots,v_k \in D_{x_1} \times \ldots \times D_{x_k} \mid c(v,v_1,\ldots,v_k)$
2) $\exists v' \in [b^{-w}, b]$, $\exists v'_1,\ldots,v'_k \in D_{x_1} \times \ldots \times D_{x_k} \mid c(v',v'_1,\ldots,v'_k)$
where a^{+w} (resp.a^{-w}) denotes the w^{th} float after (resp. before) a.

Definition of kB(w)–Consistency and Box(w)–Consistency is straightforward.

Of course, if the result of any elementary arithmetic operation which is not a float is rounded to the previous (or next) float, then there is no guarantee on the unicity of the result; the fixed-point computed by filtering algorithms will depend on the order of evaluation of the terms in an expression, or even on the order of evaluation of constraints when a weaker precision than one float is used to stop propagation (e.g. w in [14]). So, it becomes hard to set the properties of the intervals computed by filtering algorithms.

In other words, the major problem when comparing kB(w)–Consistency and Box(w)–Consistency comes from the fact that the confluency of the filtering algorithms is lost : the final values of the domains depend on the propagation strategy. Thus, no relation can be established : neither among the different kB(w)–Consistencies themselves, nor between kB(w)–Consistency and Box(w)–Consistency.

Acknowledgements

Thanks to Olivier Lhomme, Christian Bliek and Bertrand Neveu for their careful reading and helpful comments on earlier drafts of this paper. Thanks also to Frédéric Goualard, and Laurent Granvilliers for interesting discussions and email correspondence.

References

1. F. Benhamou, D. Mc Allester, and P. Van Hentenryck. CLP(Intervals) Revisited. in *Proc. Logic Programming : Proceedings of the 1994 International Symposium*, MIT Press, (1994).
2. F. Benhamou and W. Older. Applying interval arithmetic to real, integer and boolean constraints. *Journal of Logic Programming*, (1997).
3. C. Bliek. Computer Methods for Design Automation. PhD thesis, Massachusetts Institute of Technology, 1992.
4. R.P. Brent. A FORTRAN multiple-precision arithmetic package *ACM Trans. on Math. Software*, 4, no 1, 57-70, 1978.
5. J.C. Cleary. Logical arithmetic. *Future Computing Systems*, 2(2) :125–149, 1987.
6. B. Faltings. Arc–consistency for continuous variables. *Artificial Intelligence*, vol. 65, pp. 363–376, 1994.
7. E. C. Freuder. Synthesizing constraint expressions. *Communications of the ACM*, 21 :958–966, November 1978.

8. L. Granvilliers. On the combination of Box–consistency and Hull-consistency. Workshop " Non binary constraints", ECAI-98, Brighton, 23-28 August 1998.
9. E. Hansen. Global optimization using interval analysis. *Marcel Dekker*, NY, 1992.
10. Hoon Hong and Volker Stahl. Safe starting regions by fixed points and tightening. *Computing*, 53 :323–335, 1994.
11. E. Hyvőnen. Constraint reasoning based on interval arithmetic : the tolerance propagation approach. *Artificial Intelligence*, vol. 58, pp. 71–112, 1992.
12. H. Hong, V. Stahl . *Safe Starting Regions by Fixed Points and Tightening.* Computing, vol. 53, pp 323-335, 1994.
13. J. H. M. Lee and M. H. van Emden. Interval computation as deduction in CHIP. *Journal of Logic Programming*, 16 :3–4, pp.255–276, 1993.
14. O. Lhomme. Consistency techniques for numeric CSPs. in *Proc. IJCAI93, Chambery, (France)*, pp. 232–238, (August 1993).
15. O. Lhomme. Contribution à la résolution de contraintes sur les réels par propagation d'intervalles. PhD Thesis, University of Nice Sophia Antipolis - CNRS, Route des Colles, B.P. 145, 06903 Sophia Antipolis Cedex, France, 1994.
16. O. Lhomme and M. Rueher. Application des techniques CSP au raisonnement sur les intervalles. *RIA (Dunod)*,vol. 11 :3, pp. 283–312, 1997.
17. A. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, vol. 8, no. 1, pp. 99–118, 1977.
18. U. Montanari. Networks of constraints : Fundamental properties and applications to picture processing. *Information Science*, 7(2) : 95–132, 1974.
19. R. Moore, Interval Analysis. *Prentice Hall*, 1966.
20. A. Neumaier. Interval methods for systems of equations. *Cambridge University Press*, 1990.
21. W.J. Older and A. Velino. Extending prolog with constraint arithmetic on real intervals. In *Proc. of IEEE Canadian conference on Electrical and Computer Engineering.* IEEE Computer Society Press, 1990.
22. W. Older and A. Vellino. Constraint arithmetic on real intervals. in *Constraint Logic Programming : Selected Research*, eds., Frédéric Benhamou and Alain Colmerauer. MIT Press, (1993).
23. Prologia *PrologIV Constraints inside.* Parc technologique de Luminy - Case 919 13288 Marseille cedex 09 (France), 1996.
24. M. Rueher, C. Solnon. Concurrent Cooperating Solvers within the Reals. *Reliable Computing.* Kluwer Academic Publishers, Vol.3 :3, pp. 325-333, 1997.
25. E. Tsang. Foundations of Constraint Satisfaction. *Academic Press*, 1993.
26. P. Van Hentenryck, Y. Deville, and L. Michel. *Numerica. A modeling language for global optimization.* MIT Press, 1997.
27. P. Van Hentenryck, D. McAllester, and D. Kapur. *Solving Polynomial Systems Using a Branch and Prune Aprroach.* SIAM Journal (forthcomming).