Reliable Computing **5:** 213–228, 1999.

Comparing Partial Consistencies*

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, e-mail: {collavizza,delobel,rueher}@essi.fr

(Received: 2 November 1998; accepted: 13 January 1999)

Abstract. Global search algorithms have been widely used in the constraint programming framework to solve constraint systems over continuous domains. This paper precisely states the relations among the different partial consistencies which are main emphasis of these algorithms.

The capability of these partial consistencies to handle the so-called dependency problem is analysed and some efficiency aspects of the filtering algorithms are mentioned.

1. Introduction

Global search algorithms that are based upon partial consistency filtering techniques have proven their efficiency to solve non-trivial constraint systems over the reals. For instance, systems like `Newton` and `Numerica` [2], [28] behave better than interval methods on classical benchmarks of numerical analysis and interval analysis (e.g., Moré-Cosnard non-linear integral equation, Broyden banded functions). Moreover, it has been shown recently [9] that combining algorithms based on different partial consistencies can even lead to better performances.

These global search algorithms are actually "branch and prune" algorithms, i.e., algorithms that can be defined as an iteration of two steps:

1. *Pruning the search space* by reducing the intervals associated with the variables.
2. *Generating subproblems* by splitting the domains of a variable (the choice of the variable may be non deterministic or based on some heuristic).

The pruning step achieves a filtering of the domains, in other words, it reduces the intervals associated with the variables until a given partial consistency property is satisfied.

1.1. Partial Consistencies

Informally speaking, a constraint system C satisfies a partial consistency property if a relaxation of C is consistent. For instance, local consistency just requires that, taken individually, the constraints are consistent. The relevance of consistency properties is that whenever a consistency property is violated, there is an associated

* This is a revised version of the paper presented at the 4th International Conference on Constraint Programming [6].

recipe for pruning some interval. Most constraint solvers over finite domains [18], [27] are based on a partial consistency named *Arc-Consistency*. Assume c is a k-ary constraint over variables $(x_1, \ldots, x_k)$; c is arc-consistent if, for any value in $\mathbf{x}_i$, there exists at least one value in each domain $\mathbf{x}_j$ $(j \neq i)$ such that c holds. In the same way, many solvers over continuous domains [1], [25], [28] rely upon relaxations of Arc-Consistency. A relaxation of Arc-Consistency has also been used in the context of global optimization [22].

2B-Consistency (also known as hull consistency) [3], [5], [15], [16] is a relaxation of Arc-Consistency which only requires to check the Arc-Consistency property for each bound of the intervals. The key point is that this relaxation is more easily verifiable than Arc-Consistency itself. Informally speaking, variable x is 2B-Consistent for constraint "$f(x, x_1, \ldots, x_n) = 0$" if the lower (resp. upper) bound of the domain of x is the smallest (resp. largest) solution of $f(x, x_1, \ldots, x_n)$. Box-Consistency [2], [11] is a coarser relaxation (i.e., it allows more stringent pruning) of Arc-Consistency than 2B-Consistency. Variable x is Box-Consistent for constraint "$f(x, x_1, \ldots, x_n) = 0$" if the bounds of the domain of x correspond to the leftmost and the rightmost zero of the optimal interval extension of $f(x, x_1, \ldots, x_n)$.

3B-Consistency and *Bound-Consistency* are higher order extensions of 2B-Consistency and Box-Consistency which have been introduced to limit the effects of a strictly local processing:

- *3B-Consistency* [16] is a relaxation of path consistency [8], a higher order extension of Arc-Consistency. 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* [24], [28] 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.

1.2. Aim of the Paper

This paper investigates the relations between *2B-Consistency*, *Box-Consistency*, *3B-Consistency* and *Bound-Consistency*. More precisely, we prove the following properties:

- 2B-Consistency algorithms actually achieve a weaker filtering (i.e., a filtering that yields bigger intervals) than Box-Consistency, especially when a variable occurs more than once in some constraint (see Proposition 4.2). 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 the one computed by 3B-Consistency algorithms (see Proposition 5.2).

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 [10].

LAYOUT OF THE PAPER

Section 2 reviews some basic concepts and introduces the notation used 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 issues.

2. Interval Constraint Solving

This section recalls some basics of interval analysis [2], [3], [12] and formally defines a constraint system over intervals of real numbers.

2.1. NOTATION

We mainly use the notations suggested by Kearfott [13]. Thus, throughout, boldface will denote intervals, lower case will denote scalar quantities, and upper case will denote vectors and sets. Brackets "[.]" will delimit intervals while parentheses "(.)" will delimit vectors. Underscores will denote lower bounds of intervals and overscores will denote upper bounds of intervals. $\tilde{x}$ denotes any value in interval $\mathbf{x}$ (usually *not* the center of $\mathbf{x}$).

We will also use the following notations,which are slightly non-standard:

- $\mathcal{R}^{\infty} = \mathcal{R} \cup \{-\infty, +\infty\}$ denotes the set of real numbers augmented with the two infinity symbols. $\overline{\mathbb{F}}$ denotes a finite subset of $\mathcal{R}^{\infty}$ containing $\{-\infty, +\infty\}$. Practically speaking, $\overline{\mathbb{F}}$ corresponds to the set of floating-point numbers used in the implementation of non linear constraint solvers;
- if a is a constant in $\overline{\mathbb{F}}$, a^{+} (resp. a^{-}) corresponds to the smallest (resp. largest) number of $\overline{\mathbb{F}}$ strictly greater (resp. lower) than a;
- f, g denote functions over the reals; $c : \mathcal{R}^n \rightarrow \mathcal{B}ool$ denotes a constraint over the reals, $\mathbf{c}$ denotes a constraint over the intervals; $Var(c)$ denotes the variables occurring in constraint c.

2.2. INTERVAL ANALYSIS

DEFINITION 2.1 (Interval). An interval $\mathbf{x} = [\underline{x}, \overline{x}]$, with $\underline{x}$ and $\overline{x} \in \overline{\mathbb{F}}$, is the set of real numbers $\{r \in \mathcal{R} \mid \underline{x} \leq r \leq \overline{x}\}$; if $\underline{x}$ or $\overline{x}$ is the infinity symbol, then $\mathbf{x}$ is an opened interval.

$\mathcal{I}$ denotes the set of intervals and is ordered by set inclusion. $\mathcal{U}(\mathcal{I})$ denotes the set of unions of intervals.

DEFINITION 2.2 (Set Extension). Let S be a subset of $\mathcal{R}$. The *Hull* of S—denoted $\Box S$—is the smallest interval I such that $S \subseteq I$.

The term "smallest subset" (w.r.t. inclusion) must be understood according to the precision of floating-point operations. In the rest of the paper, we consider—as in [15], [2]—that results of floating-point operations are outward-rounded to preserve the correctness of the computation. However, we also 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.

DEFINITION 2.3 (Interval Extension [10], [20]).

- $\mathbf{f} : \mathcal{I}^n \to \mathcal{I}$ is an interval extension of $f : \mathcal{R}^n \to \mathcal{R}$ iff $\forall \mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{I} : f(\tilde{x}_1, \ldots, \tilde{x}_n) \in \mathbf{f}(\mathbf{x}_1, \ldots, \mathbf{x}_n)$.
- $\mathbf{c} : \mathcal{I}^n \to \mathcal{B}ool$ is an interval extension of $c : \mathcal{R}^n \to \mathcal{B}ool$ iff $\forall \mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{I} : c(\tilde{x}_1, \ldots, \tilde{x}_n) \Rightarrow \mathbf{c}(\mathbf{x}_1, \ldots, \mathbf{x}_n)$.

Similarly, $\mathbf{f}$ is the *natural* interval extension of f (see [20]) if $\mathbf{f}$ is obtained by replacing in f each constant k with the smallest interval containing k, each variable x with an interval variable $\mathbf{x}$, and each arithmetic operation with its optimal interval extension [20].

In the rest of this paper, $\mathbf{c}$ denotes the natural interval extension of c and $\oplus$, $\ominus$, $\otimes$, $\oslash$ denote the optimal interval extensions of $+$, $-$, $\times$, $/$.

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

PROPOSITION 2.1 [20]. *Let* $\mathbf{f} : \mathcal{I}^n \to \mathcal{I}$ *be the natural interval extension of* $f : \mathcal{R}^n \to \mathcal{R}$. *If each* x_i *occurs only once in* f *then* $\Box\{f(\tilde{x}_1, \ldots, \tilde{x}_n)\} = \mathbf{f}(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ *else* $\Box\{f(\tilde{x}_1, \ldots, \tilde{x}_n)\} \subseteq \mathbf{f}(\mathbf{x}_1, \ldots, \mathbf{x}_n)$.

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

PROPOSITION 2.2. *Let* $\mathbf{c} : \mathcal{I}^n \to \mathcal{B}ool$ *be the natural extension of* $c : \mathcal{R}^n \to \mathcal{B}ool$, *if each* x_i *occurs only once in* c, *then* $\mathbf{c}(\mathbf{x}_1, \ldots, \mathbf{x}_n) \Leftrightarrow c(\tilde{x}_1, \ldots, \tilde{x}_n)$.

2.3. Interval Constraint System

DEFINITION 2.4 (CSP). A CSP (Constraint System Problem) [18] is a couple (X, C) where $X = \{x_1, \ldots, x_n\}$ denotes a set of variables with associated interval domains $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, and $C = \{c_1, \ldots, c_m\}$ denotes a set of constraints.

$P_\emptyset$ denotes an empty CSP, i.e., a CSP with at least one empty domain. $\mathbf{X}' \subseteq \mathbf{X}$ means $\mathbf{x}'_i \subseteq \mathbf{x}_i$ for all i. We define a CSP $P = (X, C)$ to be smaller than a CSP $P' = (X', C)$ if $\mathbf{X}' \subseteq \mathbf{X}$. We write $P \preceq P'$ for this relation. By convention, $P_\emptyset$ is the smallest CSP.

In the following passage, we define and discuss several kinds of consistency, and the associated filtering of a CSP P.

3. 2B-Consistency

Most of the CSP systems over intervals (e.g., [1], [3], [23], [25]) compute a relaxation of Arc-Consistency [18] 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 [16] 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 $\underline{x}$ and $\overline{x}$.

DEFINITION 3.1 (2B-Consistency). Let (X, C) be a CSP and $c \in C$ a k-ary constraint over $(x_1, \ldots, x_k)$. c is 2B-Consistent iff:

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

A CSP is 2B-Consistent iff all its constraints are 2B-Consistent.

By definition, 2B-Consistency is weaker than Arc-Consistency. This point is illustrated in Example 3.1.

EXAMPLE 3.1. Let $P_1 = (\{x_1, x_2\}, \{x_1 = x_2 * x_2\})$ be a CSP with $\mathbf{x}_1 = [1, 4], \mathbf{x}_2 = [-2, 2]$. P_1 is 2B-Consistent but not arc-Consistent since there is no value in $\mathbf{x}_1$ which satisfies the constraint when $x_2 = 0$.

DEFINITION 3.2 (Closure by 2B-Consistency [16]). The filtering by 2B-Consistency of $P = (X, C)$ is the CSP $P' = (X', C)$ such that:

- P and P' have the same solutions;
- P' is 2B-Consistent;
- $X' \subseteq X$ and the domains in X' are the largest ones for which P' is 2B-Consistent.

We note $\Phi_{2B}(P)$ the filtering by 2B-Consistency of P. In the following we will use the term *closure* by 2B-Consistency to emphasize the fact that this filtering always exists and is unique [16].

Proposition 3.1 states a property which is useful when comparing 2B-Consistency and Box-Consistency.

IN(in C**, inout** $\mathbf{X}$**)**
Queue $\leftarrow C$;
while Queue $\neq \emptyset$
  $c \leftarrow$ POP Queue;
  $\mathbf{X}' \leftarrow$ narrow$(c, \mathbf{X})$;
  if $\mathbf{X}' \neq \mathbf{X}$ **then**
    $\mathbf{X} \leftarrow \mathbf{X}'$;
    Queue $\leftarrow$ Queue $\cup$ $\{c' \in C \mid Var(c) \cap Var(c') \neq \emptyset\}$
  endif
endwhile

Figure 1. Algorithm IN.

PROPOSITION 3.1. *Let $P = (X, C)$ be a CSP such that no variable occurs more than once in any constraint of C. Let $c \in C$ be a k-ary constraint over the variables $(x_1, \ldots, x_k)$. P is 2B-Consistent iff $\forall c \in C$, $\forall i \in 1, \ldots, k$ the following relations hold:*

- $\mathbf{c}(\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, [\underline{\mathbf{x}_i}, \underline{\mathbf{x}_i}^+), \mathbf{x}_{i+1}, \ldots, \mathbf{x}_k)$, *and*
- $\mathbf{c}(\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, (\overline{\mathbf{x}_i}^-, \overline{\mathbf{x}_i}], \mathbf{x}_{i+1}, \ldots, \mathbf{x}_k)$.

Proof. Assume that both $\mathbf{c}(\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, [\underline{\mathbf{x}_i}, \underline{\mathbf{x}_i}^+), \mathbf{x}_{i+1}, \ldots, \mathbf{x}_k)$ and $\mathbf{c}(\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, (\overline{\mathbf{x}_i}^-, \overline{\mathbf{x}_i}], \mathbf{x}_{i+1}, \ldots, \mathbf{x}_k)$ hold. By Proposition 2.2 we have:

1. $\exists \tilde{x}_1 \in \mathbf{x}_1, \ldots, \exists \tilde{x}_{i-1} \in \mathbf{x}_{i-1}, \exists x_i \in [\underline{x_i}, \underline{x_i}^+), \exists \tilde{x}_{i+1} \in \mathbf{x}_{i+1}, \ldots, \exists \tilde{x}_k \in \mathbf{x}_k$ such that $c(\tilde{x}_1, \ldots, \tilde{x}_{i-1}, \tilde{x}_i, \tilde{x}_{i+1}, \ldots, \tilde{x}_k)$ holds, and
2. $\exists \tilde{x}_1 \in \mathbf{x}_1, \ldots, \exists \tilde{x}_{i-1} \in \mathbf{x}_{i-1}, \exists x_i \in (\overline{x_i}^-, \overline{x_i}], \exists \tilde{x}_{i+1} \in \mathbf{x}_{i+1}, \ldots, \exists \tilde{x}_k \in \mathbf{x}_k$ such that $c(\tilde{x}_1, \ldots, \tilde{x}_{i-1}, \tilde{x}_i, \tilde{x}_{i+1}, \ldots, \tilde{x}_k)$ holds.

Thus, $\mathbf{x}_i = \square\{\tilde{x}_i \mid \exists \tilde{x}_1 \in \mathbf{x}_1, \ldots, \exists \tilde{x}_{i-1} \in \mathbf{x}_{i-1}, \exists \tilde{x}_{i+1} \in \mathbf{x}_{i+1}, \ldots, \exists \tilde{x}_k \in \mathbf{x}_k$ such that $c(\tilde{x}_1, \ldots, \tilde{x}_i, \ldots, \tilde{x}_k)$ holds$\}$.

The counterpart results from the definition of 2B-Consistency. □

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 [18]—can be written down as in Figure 1. IN implements the computation of the closure by 2B-Consistency of a CSP $P = (X, C)$. *narrow*$(c, \mathbf{X})$ 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 the narrowing of domains in *narrow*$(c, \mathbf{X})$. Let c be a k-ary constraint over $X = (x_1, \ldots, x_k)$: for

each i in $1, \ldots, k$, $\pi_i(c, \mathbf{X})$ denotes the projection over x_i of the solutions of c in the space delimited by $\mathbf{X}$.

DEFINITION 3.3 (Projection of a Constraint). $\pi_i(c, \mathbf{X}) : (C, I^k) \rightarrow \mathcal{U}(\mathcal{I})$ is the projection of c on x_i iff $\pi_i(c, \mathbf{X}) = \{\tilde{x}_i \mid \exists(\tilde{x}_1, \ldots, \tilde{x}_{i-1}, \tilde{x}_{i+1}, \ldots, \tilde{x}_k) \in \mathbf{x}_1 \times \cdots \times \mathbf{x}_{i-1} \times \mathbf{x}_{i+1}, \cdots \times \mathbf{x}_k$ such that $c(\tilde{x}_1, \ldots, \tilde{x}_i, \ldots, \tilde{x}_k)$ holds$\}$.

DEFINITION 3.4 (Approximation of the Projection). $AP_i(c, \mathbf{X}) : (C, \mathcal{I}^k) \rightarrow \mathcal{I}$ is an approximation of $\pi_i(c, \mathbf{X})$ iff $AP_i(c, \mathbf{X}) = \square\, \pi_i(c, \mathbf{X}) = [\min\ \pi_i(c, \mathbf{X}), \max\ \pi_i(c, \mathbf{X})]$. In other words, $AP_i(c, \mathbf{X})$ is the smallest interval encompassing projection $\pi_i(c, \mathbf{X})$.

The following proposition trivially holds:

PROPOSITION 3.2. *Constraint c is 2B-Consistent on $\mathbf{X}$ iff for all i in $\{1, \ldots, k\}$, $\mathbf{x}_i = AP_i(c, \mathbf{X})$.*

In general, AP_i cannot be computed efficiently 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*. Since such a symbolic transformation is not always possible, this problem is usually solved by decomposing the constraint system into a set of primitive constraints for which the AP_i can easily be computed [17]. Primitive constraints are generated syntactically by introducing new variables.

DEFINITION 3.5 (Decomposition of a Constraint System). Let $P = (X, C)$ be a CSP and $c \in C$ a constraint. We define $\mathcal{M}_c \subseteq X$ 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 $x \in \mathcal{M}_c$ with a new variable y with domain $\mathbf{y} = \mathbf{x}$ 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, $X_{New} = \bigcup\{New_{(x,c)} \mid x \in X$ and $c \in C\}$. P_{decomp} is the CSP (X', C') where $X' = X \cup X_{New}$, and $C' = \{decomp(c) \mid c \in C\}$.

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** of P. However, a *local* consistency like Arc-Consistency is not preserved by such a rewriting. Indeed, decomposition reduces the scope of local consistency filtering algorithms. Thus, P_{decomp} is a *relaxation* of P when computing a relaxation of Arc-Consistency.

* B. Faltings [7] 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.

** 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}.

EXAMPLE 3.2 (Decomposition of the Constraint System). Let $c : x_1 + x_2 - x_1 = 0$ be a constraint and $\mathbf{x}_1 = [-1, 1]$, $\mathbf{x}_2 = [0, 1]$ 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: $decomp(c) = \{x_1 + x_2 - x_3 = 0, x_1 = x_3\}$.

In this new constraint system, each projection can easily be computed with interval arithmetic. For instance, $AP_1(x_1 + x_2 - x_3 = 0, (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3))$ is $\mathbf{x}_1 \cap (\mathbf{x}_3 \ominus \mathbf{x}_2)$. 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 $x_1 = -1$ and $x_3 = 0$ satisfy $x_1 + x_2 - x_3 = 0$ when $x_2 = 1$. On the initial constraint, 2B-Consistency reduces $\mathbf{x}_2$ to $[0, 0]$ while it yields $\mathbf{x}_1 = [-1, 1], \mathbf{x}_2 = [0, 1]$ for $decomp(c)$.

Remark. Like almost all other examples in this paper, Example 3.2 can be trivially 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 [2], [11] is a coarser relaxation of Arc-Consistency than 2B-Consistency. It mainly consists of replacing every existentially quantified variable but one with its interval in the definition of 2B-Consistency. Thus, Box-Consistency generates a system of univariate interval 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 4.1 (Box-Consistency). Let (X, C) be a CSP and $c \in C$ a k-ary constraint over the variables $(x_1, \ldots, x_k)$. c is Box-Consistent if, for all x_i the following relations hold:

1. $\mathbf{c}(\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, [\underline{\mathbf{x}_i}, \underline{\mathbf{x}_i}^+), \mathbf{x}_{i+1}, \ldots, \mathbf{x}_k)$,
2. $\mathbf{c}(\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, (\overline{\mathbf{x}_i}^-, \overline{\mathbf{x}_i}], \mathbf{x}_{i+1}, \ldots, \mathbf{x}_k)$.

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

PROPOSITION 4.1. *$\Phi_{2B}(P) \preceq \Phi_{Box}(P)$ and $\Phi_{2B}(P) \equiv \Phi_{Box}(P)$ when no variable occurs more than once in the constraints of 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 2.2 the equivalence holds when no variable occurs more than once in the constraints of C. □

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 4.1).

EXAMPLE 4.1. Example 3.2 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.

Of course, 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 4.2. $\Phi_{Box}(P) \preceq \Phi_{2B}(P_{decomp})$.

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 $\Phi_{Box}(P) \preceq \Phi_{Box}(P_{decomp})$. By Proposition 4.1 we have: $\Phi_{Box}(P_{decomp}) \equiv \Phi_{2B}(P_{decomp})$, and thus $\Phi_{Box}(P) \preceq \Phi_{2B}(P_{decomp})$. □

EXAMPLE 4.2. Let c be the constraint $x_1 + x_2 - x_1 - x_1 = 0$ where $\mathbf{x}_1 = [-1,1]$ and $\mathbf{x}_2 = [0.5,1]$. 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 $\mathbf{x}_1$ and $\mathbf{x}_2$.

Box-Consistency can tackle some dependency problems in a constraint c which contains only one variable occurring more than once. More precisely, Box-Consistency enables us to reduce the domain $\mathbf{x}$ if variable x occurs more than once in c and if $\mathbf{x}$ contains inconsistent values. For instance, in Example 4.2, filtering by Box-Consistency reduces $\mathbf{x}_1$ because value -1 of $\mathbf{x}_1$ has no support in $\mathbf{x}_2$.

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 3.2, value 1 of $\mathbf{x}_2$ has no support in $\mathbf{x}_1$ but Box-Consistency fails to detect the inconsistency because $[-1,1] \oplus [1^-,1] \ominus [-1,1] \cap [0,0]$ is not empty.

4.2. COMPUTING BOX-CONSISTENCY

The Box-Consistency filtering algorithm proposed in [2], [28], [29] 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. The function *narrow*(c, $\mathbf{X}$) prunes the domains of

```
function LNAR (IN: f_x, x, RETURN: Interval)
  r ← x̄
  if 0 ∉ f_x(x) then return ∅
    else i ← NEWTON(f_x, x)
      if 0 ∈ f_x([i, i⁺]) then return [i, r]
        else SPLIT(i, i_1, i_2)
          l_1 ← LNAR(f_x, i_1)
          if l_1 ≠ ∅ then return [l_1, r]
            else return [LNAR(f_x, i_2), r]
          endif
      endif
  endif
```

Figure 2. Function LNAR.

the variables of c until c is Box-Consistent. Roughly speaking, for each variable x of constraint c, an interval univariate function $\mathbf{f_x}$ is generated from c by replacing all variables but x with their intervals. The narrowing process consists of finding the leftmost and rightmost zeros of $\mathbf{f_x}$. Figure 2 shows function LNAR which computes the leftmost zero of $\mathbf{f_x}$ for initial domain I_x of variable x (this procedure is given in [29]).

Function LNAR first prunes interval $\mathbf{x}$ with function NEWTON which is an interval version of the classical Newton method. However, depending on the value of $\mathbf{x}$, Newton may not reduce $\mathbf{x}$ enough to make $\mathbf{x}$ Box-Consistent. So, a split step is applied in order to ensure that the left bound of $\mathbf{x}$ is actually a zero. Function SPLIT divides interval $\mathbf{i}$ in two intervals $\mathbf{i_1}$ and $\mathbf{i_2}$, $\mathbf{i_1}$ being the left part of the interval. The splitting process avoids the problem of finding a safe starting box for Newton (see [11]). As mentioned in [29], even if $\mathbf{f_x}$ 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 and Box-Consistency are only partial consistencies which are often too weak for computing an relevant superset of solutions of a CSP. In the same way that Arc-Consistency has been generalized to higher consistencies (e.g., path consistency [18]), 2B-Consistency and Box-Consistency can be generalized to higher order consistencies [16].

5.1. 3B-CONSISTENCY

DEFINITION 5.1 (3B-Consistency [16]). Let $P = (X, C)$ be a CSP and x a variable of X. Let also:

- $P_{\mathbf{x} \leftarrow [\underline{x}, \underline{x}^+)}$ be the CSP derived from P by substituting $\mathbf{x}$ with $[\underline{x}, \underline{x}^+)$;
- $P_{\mathbf{x} \leftarrow (\overline{x}^-, \overline{x}]}$ be the CSP derived from P by substituting $\mathbf{x}$ with $(\overline{x}^-, \overline{x}]$.

$\mathbf{x}$ is 3B-Consistent iff $\Phi_{2B}(P_{\mathbf{x} \leftarrow [\underline{x}, \underline{x}^+)}) \neq P_\emptyset$ and $\Phi_{2B}(P_{\mathbf{x} \leftarrow (\overline{x}^-, \overline{x}]}) \neq P_\emptyset$. A CSP is 3B-Consistent iff all its domains are 3B-Consistent.

It results from Definition 5.1 that any CSP which is 3B-Consistent is also 2B-Consistent [16]. The generalization of the 3B-Consistency to kB-Consistency is straightforward and is given in [16], [17].

3B-Consistency is less local than 2B-Consistency or Box-Consistency. Proposition 5.1 shows that 3B-Consistency always prunes more strongly than Box-Consistency, even if 3B-Consistency is achieved on the decomposed system and Box-Consistency on the initial system.

PROPOSITION 5.1. *Let $P = (X, C)$ 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 $(x_1, \ldots, x_k)$, x is one of the variables occuring more than once in c and $New_{(x,c)} = (x_{k+1}, \ldots, x_{k+m})$ is the set of variables introduced for replacing the multiple occurrences of x in c. Suppose that P_{decomp} is 3B-Consistent for $\mathbf{x}$.

Consider P_1, the CSP derived from P_{decomp} by reducing domain $\mathbf{x}$ to $[\underline{x}, \underline{x}^+)$. P_1 is 2B-Consistent for $\mathbf{x}$ and thus the domain of all variables in $New_{(x,c)}$ is reduced to $[\underline{x}, \underline{x}^+)$; this is due to the equality constraints added when introducing new variables. From Proposition 3.1, it results that the following relation holds:

$$\mathbf{c}'(\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, [\underline{\mathbf{x}}, \underline{\mathbf{x}}^+), \mathbf{x}_{i+1}, \ldots, \mathbf{x}_k, [\underline{\mathbf{x}}, \underline{\mathbf{x}}^+), \ldots, [\underline{\mathbf{x}}, \underline{\mathbf{x}}^+), \mathbf{x}_{k+m+1}, \ldots, \mathbf{x}_n)$$

$\mathbf{c}'$ is the very same syntactical expression as $\mathbf{c}$ (where some variables have been renamed).

$(\mathbf{x}_{k+m+1}, \ldots, \mathbf{x}_n)$ 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 $\mathbf{c}(\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, [\underline{\mathbf{x}}, \underline{\mathbf{x}}^+), \mathbf{x}_{i+1}, \ldots, \mathbf{x}_k)$ holds too.

The same reasoning can be applied when x is replaced with its upper bound $(\overline{x}^-, \overline{x}]$. So we conclude that $\mathbf{x}$ is also Box-Consistent. □

EXAMPLE 5.1. Let $C = \{x_1 + x_2 = 100, x_1 - x_2 = 0\}$ and $\mathbf{x}_1 = [0, 100], \mathbf{x}_2 = [0, 100]$ 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 5.1:

PROPOSITION 5.2. $\Phi_{3B}(P_{decomp}) \preceq \Phi_{Box}(P)$.

Thus, 3B-Consistency allows us 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 5.2).

EXAMPLE 5.2. Let c be the constraint $x_1 * x_2 - x_1 + x_3 - x_1 + x_1 = 0$ where $\mathbf{x}_1 = [-4, 3]$, $\mathbf{x}_2 = [1, 2]$ and $\mathbf{x}_3 = [-1, 5]$. $decomp(c) = \{x_1 * x_2 - x_4 + x_3 - x_5 + x_6 = 0, x_1 = x_4 = x_5 = x_6\}$. c is not 2B-Consistent since there are no values in $\mathbf{x}_1$ and $\mathbf{x}_2$ 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 5.3 shows that Φ_{2B}(P) $\preceq \Phi_{3B}(P_{decomp})$ does not hold and Example 5.2 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 5.3. Let P be a CSP defined by $C = \{x_1 + x_2 = 10; x_1 + x_1 - 2x_2 = 0\}$ where $\mathbf{x}_1 = \mathbf{x}_2 = [-10, 10]$. $decomp(x_1 + x_1 - 2x_2 = 0) = \{x_1 + x_3 - 2x_2 = 0, x_3 = x_1\}$. P is 2B-Consistent but P_{decomp} is not 3B-Consistent: Indeed, when x_1 is fixed to 10, $\Phi_{2B}(P_{\mathbf{x_1} \leftarrow [10^-, 10]}) = P_\emptyset$ since $\mathbf{x}_2$ is reduced to $\emptyset$. In this case, the link between x_1 and x_3 is preserved and 3B-Consistency reduces $\mathbf{x}_2$ to [5, 5].

5.2. BOUND-CONSISTENCY

Bound-Consistency was suggested in [17] and was formally defined in [28]. 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 5.2 (Bound-Consistency). Let (X, C) be a CSP and $c \in C$ a k-ary constraint over the variables $(x_1, \ldots, x_k)$. c is Bound-Consistent if for all x_i, the following relations hold:

1. $\Phi_{Box}(\mathbf{c}(\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, [\underline{\mathbf{x}}, \underline{\mathbf{x}}^+), \mathbf{x}_{i+1}, \ldots, \mathbf{x}_k)) \neq P_\emptyset$,
2. $\Phi_{Box}(\mathbf{c}(\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, (\overline{\mathbf{x}}^-, \overline{\mathbf{x}}), \mathbf{x}_{i+1}, \ldots, \mathbf{x}_k)) \neq P_\emptyset$.

Since $\Phi_{Box}(P) \preceq \Phi_{2B}(P_{decomp})$ 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 5.1 and 3.2.

6. Efficiency Issues

The aim of numerical CSP is not to compute partial consistencies but to find accurate solutions; that is, either small intervals containing isolated solutions, or intervals which tightly encompass sets of continuous solutions. Thus, in practical systems (e.g., `Numerica` [28], `PROLOG IV` [25]), partial consistencies are combined with several search heuristics and splitting techniques. Experimental results of `Numerica` and `Newton` are very impressive. However it is difficult to draw a conclusion from the published benchmarks because these systems differ in several critical points:

- They use different splitting heuristics.
- There are significant variations in the implementation of the filtering algorithms (precision parameters in order to force an early halt to the propagation process, constraint ordering, detection of cycles, ...).
- They use different implementation languages (`Prolog`, `C`, ...).

So we limit the discussion to a brief examination of three key points which help to better understand the performances of the different systems:

1. **Cost of the basic narrowing operator**: Performing interval newton method on univariate functions is more expensive than computing projection functions on primitive constraints. For instance, let $C = \{x^2 = 2\}$ and $\mathbf{x} = [1, 10]$. Box-Consistency requires 6 narrowing steps with the Newton method (about 100 interval operations) whereas 2B-Consistency only requires the computation of one relational square root operation and one intersection over intervals [9]. Thus, 2B is more efficient than Box on problems where the projection functions compute an accurate result.

   The gain of performance due to accurate projection functions is well illustrated using the pentagon problem. This problem consists of finding the coordinates of five points of a circle such that these points define a convex pentagon. The constraint system consists of five quadratic equations. To avoid an infinite number of solutions, the first point is given, and the five points are ordered to avoid symmetrical solutions. On this example, Bound-Consistency and 3B-Consistency achieve the same pruning. According to [1] Box is about forty times slower than 2B on this example.

2. **Expansion of the constraint system**: Decomposition of the initial constraint system may generate a huge number of primitive constraints when variables occur more than once. For instance, consider the classical "Broyden 160" example (160 initial variables, 160 constraints). Box-Consistency will generate 160 variables, 1184 univariate functions whereas 2B-Consistency will generate 2368 variables, 6944 ternary projection functions.

   From a practical point of view, 2B-Consistency is seriously weakened by the decomposition required for computing the narrowing functions. On the other

hand, the univariate functions generated by Box-Consistency can be handled very efficiently using Newton-like methods.

3. **Precision of the computation**: For a fixed final precision, the efficiency of the computation may strongly depend on the accuracy of the partial consistency filtering algorithm. For instance, consider again the resolution of the "Broyden 160" problem by combining Box-Consistency filtering and a domain splitting strategy. If the final intervals have to be computed with a size smaller than or equal to 10^{-8}, the computation is about 10 times faster with a coarse relaxation of Box-Consistency than with an accurate one [9].

It appears that the following approaches are promising:

- combining different partial consistencies [9];
- pre-processing of the constraints (e.g., symbolic transformations);
- intelligent search strategies (e.g., use of extrapolation techniques before starting a costly filtering process [14], dynamic choice of the filtering precision).

7. Conclusion

This paper has investigated the relations among 2B-Consistency, 3B-Consistency, Box-Consistency and Bound-Consistency. The main result is a proof of the following properties:

$$\Phi_{Bound}(\mathbf{P}) \preceq \Phi_{3B}(P_{decomp}) \preceq \Phi_{Box}(P),$$
$$\Phi_{2B}(\mathbf{P}) \preceq \Phi_{Box}(\mathbf{P}) \preceq \Phi_{2B}(\mathbf{P}_{decomp}),$$
$$\Phi_{3B}(\mathbf{P}_{decomp}) \quad \text{and} \quad \Phi_{2B}(\mathbf{P}) \quad \text{are not comparable.}$$

The advantage of Box-Consistency is 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.

Efficiency of the filtering algorithms is a critical issue, but it is difficult to draw a conclusion from the published benchmarks. Further experimentation combining these different partial consistencies with various search techniques is required to better understand their advantages and drawbacks and to define the class of application in which each of them is most relevant.

8. 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, Laurent Granvilliers, and Arnold Neumaier for interesting discussions.

Thanks also go to a referee for his many comments, which hopefully led to significant improvements of this paper.

References

1. Benhamou, F., Goualard, F., and Granvilliers, L.: Programming with the DecLIC Language, in: *Proceedings of the Second Workshop on Interval Constraints*, Port-Jefferson, NY, 1997.
2. Benhamou, F., Mc Allester, D., and Van Hentenryck, P.: CLP(Intervals) Revisited, in: *Proc. Logic Programming: Proceedings of the 1994 International Symposium*, MIT Press, 1994.
3. Benhamou, F. and Older, W.: Applying Interval Arithmetic to Real, Integer and Boolean Constraints, *Journal of Logic Programming* **32** (1997), pp. 1–24.
4. Brent, R. P.: A FORTRAN Multiple-Precision Arithmetic Package, *ACM Trans. on Math. Software* **4** (1) (1978), pp. 57–70.
5. Cleary, J. C.: Logical Arithmetic, *Future Computing Systems* **2** (2) (1987), pp. 125–149.
6. Collavizza, H., Delobel, F., and Rueher, M.: A Note on Partial Consistencies over Continuous Domains Solving Techniques, in: *Proc. CP98 (Fourth International Conference on Principles and Practice of Constraint Programming)*, LNCS 1520, Springer Verlag, 1998.
7. Faltings, B.: Arc-Consistency for Continuous Variables, *Artificial Intelligence* **65** (1994), pp. 363–376.
8. Freuder, E. C.: Synthesizing Constraint Expressions, *Communications of the ACM* **21** (1978), pp. 958–966.
9. Granvilliers, L.: *On the Combination of Box-Consistency and Hull-Consistency*, Workshop ECAI on Non Binary-Constraints, Brighton, United Kingdom, 1998
10. Hansen, E.: *Global Optimization Using Interval Analysis*, Marcel Dekker, NY, 1992.
11. Hong, H., Stahl, V.: Safe Starting Regions by Fixed Points and Tightening, *Computing* **53** (1994), pp. 323–335.
12. Hyvönen, E.: Constraint Reasoning Based on Interval Arithmetic: The Tolerance Propagation Approach, *Artificial Intelligence* **58** (1992), pp. 71–112.
13. Kearfott, R. B.: *Rigorous Global Search: Continuous Problems*, Kluwer Academic Publishers, Dordrecht, Netherlands, 1996.
14. Lebbah, Y. and Lhomme, O.: Acceleration Methods for Numeric CSPs AAAI, MIT Press, 1998.
15. Lee, J. H. M. and Van Emden, M. H.: Interval Computation as Deduction in CHIP, *Journal of Logic Programming* **16** (1993), pp. 255–276.
16. Lhomme, O.: Consistency Techniques for Numeric CSPs, in: *Proc. IJCAI93*, Chambery, France, 1993, pp. 232–238.
17. Lhomme, O. and Rueher, M.: Application des techniques CSP au raisonnement sur les intervalles, *RIA (Dunod)* **11** (3) (1997), pp. 283–312.
18. Mackworth, A.: Consistency in Networks of Relations, *Artificial Intelligence* **8** (1) (1997), pp. 99–118.
19. Montanari, U.: Networks of Constraints: Fundamental Properties and Applications to Picture Processing, *Information Science* **7** (2) (1974), pp. 95–132.
20. Moore, R.: *Interval Analysis*, Prentice Hall, 1966.
21. Neumaier, A.: *Interval Methods for Systems of Equations*, Cambridge University Press, 1990.
22. Neumaier, A., Dallwig, S., and Schichl, H.: GLOPT—A Program for Constrained Global Optimization, in: Bomze, I. et al. (eds.), *Developments in Global Optimization*, Kluwer Academic Publishers, Dordrecht, 1997, pp. 19–36.
23. Older, W. J. and Vellino, A.: Extending Prolog with Constraint Arithmetic on Real Intervals, in: *Proc. of IEEE Canadian Conference on Electrical and Computer Engineering*, IEEE Computer Society Press, 1990.
24. Puget, J.-F. and Van Hentenryck, P.: A Constraint Satisfaction Approach to a Circuit Design Problem, *Journal of Global Optimization* **13** (1998), pp. 75–93.

25. Prologia: *PrologIV Constraints Inside*, Parc technologique de Luminy—Case 919 13288 Marseille cedex 09, France, 1996.
26. Rueher, M. and Solnon, C.: Concurrent Cooperating Solvers within the Reals, *Reliable Computing* **3** (3) (1997), pp. 325–333.
27. Tsang, E.: *Foundations of Constraint Satisfaction*, Academic Press, 1993.
28. Van Hentenryck, P., Deville, Y., and Michel, L.: *Numerica. A Modeling Language for Global Optimization*, MIT Press, 1997.
29. Van Hentenryck, P., McAllester, D., and Kapur, D.: Solving Polynomial Systems Using a Branch and Prune Approach, *SIAM Journal on Numerical Analysis* **34** (2) (1997).