A new interval arithmetic to generate the complementary of contractors

Pierre Filiol, Théotime Bollengier, Luc Jaulin, Jean-Christophe Le Lann

A new interval arithmetic to generate the complementary of contractors

Pierre Filiol, Théotime Bollengier, Luc Jaulin, Jean-Christophe Le Lann

Keywords: Intervals, Contractors, complement, Not a Number

Abstract: Contractor algebra is used to characterize a set defined as a composition of sets defined by inequalities. It mainly uses interval methods combined with constraint propagation. This algebra includes the classical operations we have for sets such as the intersection, the union and the inversion. Now, it does not include to complement operator. The reason for this is probably related to the interval arithmetic itself. In this paper, we show that it we change the arithmetic used for intervals adding a single flag, similar to *not a number*, we are able to include easily the complement in the algebra of contractors.

I. Introduction

Contractor algebra as defined in [3] does not allow any non-monotonic operation. It means that if a contractor $\mathcal{C}$ is defined by an expression $\mathcal{E}$ of other contractors $\mathcal{C}_i$ then we always have

$$\forall i, \mathcal{C}_i \subset \mathcal{C}_i^{'} \Rightarrow \mathcal{E}\left(\mathcal{C}_1, \mathcal{C}_2, \dots\right) \subset \mathcal{E}\left(\mathcal{C}_1^{'}, \mathcal{C}_2^{'}, \dots\right). \quad (1)$$

As a consequence the complementary $\overline{\mathcal{C}}$ of a contractor $\mathcal{C}$ or the restriction $\mathcal{C}_1 \setminus \mathcal{C}_2$ of two contractors $\mathcal{C}_1, \mathcal{C}_2$ (which both correspond to non-monotonic operations) is not defined.

To me more precise, contractor algebra allows to construct a contractor for expressions of sets defined by union, intersection and inversion of other sets. Take for instance the set

$$\mathbb{X} = \mathbb{X}_1 \cup \mathbf{f}^{-1}(\mathbb{X}_2 \cap \mathbb{X}_3). \quad (2)$$

We can represent its expression by the tree of Figure 1(a) or equivalently by the following expressions

$$\begin{array}{rcl} \mathbb{X} & = & \mathbb{X}_1 \cup \mathbb{B} \\ \mathbb{B} & = & \mathbf{f}^{-1}(\mathbb{A}) \\ \mathbb{A} & = & \mathbb{X}_2 \cap \mathbb{X}_3 \end{array} \quad (3)$$

The intermediate sets $\mathbb{A}$ and $\mathbb{B}$ correspond to nodes of the tree. In practice, the leaves $\mathbb{X}_i$ of the tree are *set inverse* (or equivalently *inequality constraints*) of the form

$$\mathbb{X}_i = \varphi_i^{-1}([\mathbf{y}_i]) = \{\mathbf{x}_i \,|\, \varphi_i(\mathbf{x}_i) \in [\mathbf{y}_i]\} \quad (4)$$

where φ_i is a function defined by an algorithm and $[\mathbf{y}_i]$ is a box of $\mathbb{R}^n$. A contractor for $\mathbb{X}_i$ is usually built by a forward-backward procedure as for instance $HC4$-revised [1]. The contractor associated with the constraint $\varphi(\mathbf{x}) \in [\mathbf{y}]$ is denoted by $\mathcal{C}^{\updownarrow}_{\varphi^{-1}([\mathbf{y}])}$.

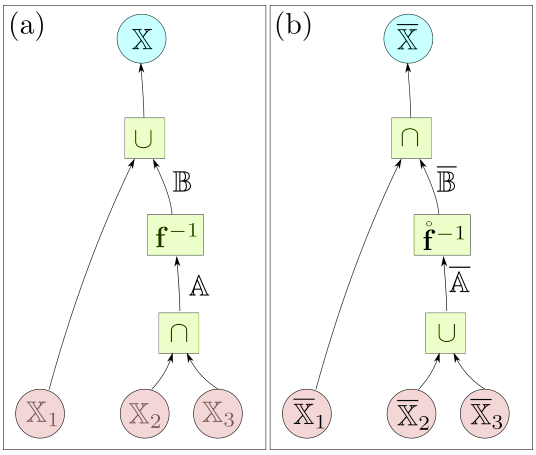

Fig. 1. (a) Contractor tree for $\mathbb{X}_1 \cup \mathbf{f}^{-1}(\mathbb{X}_2 \cap \mathbb{X}_3)$; (b) its complementary

Once the contractor for $\mathbb{X}$ is built from the tree, a paver [5] is called to provide an outer approximation for $\mathbb{X}$. More precisely, the paver generates boxes $[\mathbf{x}]$ of $\mathbb{R}^n$ that have to be contracted by the available contractors. The resulting procedure for contracting the set $\mathbb{X}$ defined by 2 is given by the following algorithm.

Algorithm 1 Contractor for $\mathbb{X} = \mathbb{X}_1 \cup \mathbf{f}^{-1}(\mathbb{X}_2 \cap \mathbb{X}_3)$

	Input: $[\mathbf{x}]$
1	$[\mathbf{x}_1] = \mathcal{C}_{\mathbb{X}_1}([\mathbf{x}])$
2	$[\mathbf{b}] = [\mathbf{x}]$
3	$[\mathbf{a}] = \mathbf{f}([\mathbf{b}])$
4	$[\mathbf{x}_2] = \mathcal{C}_{\mathbb{X}_2}([\mathbf{a}])$
5	$[\mathbf{x}_3] = \mathcal{C}_{\mathbb{X}_3}([\mathbf{a}])$
6	$[\mathbf{a}] = [\mathbf{x}_2] \cap [\mathbf{x}_3]$
7	$[\mathbf{b}] = \mathcal{C}^{\updownarrow}_{\mathbf{f}^{-1}([\mathbf{a}])}([\mathbf{b}])$
8	$[\mathbf{x}] = [\mathbf{x}_1] \sqcup [\mathbf{b}]$
9	return $[\mathbf{x}]$

This procedure is approximately what is performed by IBEX [2] even if IBEX does not admit a set expression as an input.

To express the complement $\overline{\mathbb{X}}$ we need to use the *De Morgan's laws* which states that:

- the complement of the union of two sets is the same as the intersection of their complements

- the complement of the intersection of two sets is the same as the union of their complements

We get

$$\overline{\mathbb{X}} = \overline{\mathbb{X}}_1 \cap \left(\mathbf{f}^{-1}(\overline{\mathbb{X}}_2 \cup \overline{\mathbb{X}}_3) \cup \overline{dom(\mathbf{f})} \right). \quad (5)$$

Note that we had to introduce the domain of $\mathbf{f}$, denoted by $dom(\mathbf{f})$, to take into account the fact that $\mathbf{f}$ may be a partial (*i.e.*, not defined everywhere).

If we define the set-valued function $\mathring{\mathbf{f}}^{-1} : \mathcal{P}(\mathbb{R}^m) \mapsto \mathcal{P}(\mathbb{R}^n)$ as

$$\mathring{\mathbf{f}}^{-1}(\mathbb{Y}) = \mathbf{f}^{-1}(\mathbb{Y}) \cup \overline{dom(\mathbf{f})}, \quad (6)$$

then we have

$$\overline{\mathbb{X}} = \overline{\mathbb{X}}_1 \cap \left(\mathring{\mathbf{f}}^{-1}(\overline{\mathbb{X}}_2 \cup \overline{\mathbb{X}}_3) \right). \quad (7)$$

The decomposition for $\overline{\mathbb{X}}$ is defined by

$$\begin{array}{rcl} \overline{\mathbb{X}} & = & \overline{\mathbb{X}}_1 \cap \overline{\mathbb{B}} \\ \overline{\mathbb{B}} & = & \mathring{\mathbf{f}}^{-1}(\overline{\mathbb{A}}) \\ \overline{\mathbb{A}} & = & \overline{\mathbb{X}}_2 \cup \overline{\mathbb{X}}_3 \end{array} \quad (8)$$

which corresponds to the tree of Figure 1(b). Since the sets $\mathbb{X}_i$ where defined by $\varphi_i(\mathbf{x}_i) \in [\mathbf{y}_i]$, the complement is defined by

$$\overline{\mathbb{X}}_i = \mathring{\varphi}_i^{-1}(\overline{[\mathbf{y}_i]}). \quad (9)$$

To implement, the complementary of a contractor using the *De Morgan low,* the only brick we need is the forward-backward contractor for the set

$$\mathring{\mathbf{f}}^{-1}([\mathbf{y}]) = \mathbf{f}^{-1}([\mathbf{y}]) \cup \overline{dom(\mathbf{f})} \quad (10)$$

Now, the set $\mathring{\mathbf{f}}^{-1}([\mathbf{y}])$ is not a set inverse as defined by (4) and thus we cannot apply a forward-backward contractor without an extension which will be proposed in this paper.

L'algorithme avec l'arbre n'a jamais été implementé. Ca pourrait être une contribution intéressante.

The paper is organized as follows. Section II presents an extension of the arithmetic on real numbers, named *total real arithmetic*, and shows the role of a flag named ι in the case where partial functions are involved. Section III introduces of the total interval arithmetic. Section III provides the notion of total contractors and extends the classical forward-backward contractor to total intervals. Section V concludes the paper.

II. Total extension

A. Definitions

In mathematics, a function $f : X \mapsto Y$ which is defined for all $x \in X$ is said to be *total*. Equivalently, a function $f : X \mapsto Y$ is total if

$$\forall x \in X, \exists y \in Y \text{ such that } f(x) = y. \quad (11)$$

A *partial* function f is not defined for all x. Given a partial function f, the *total extension* is obtained by adding an element to Y, say ι which collects all $x \notin dom(f)$. To be more precise, we give the following definition.

Definition 1. The *total extension* of the partial function $f : X \mapsto Y$ is $\mathring{f} = X \cup \{\iota\} \mapsto Y \cup \{\iota\}$ with

$$\mathring{f} = \begin{cases} f(x) & \text{if } x \in dom(f) \\ \iota & \text{otherwize} \end{cases} \quad (12)$$

B. Illustration

Consider the partial function f as given in Figure 2. We have

$$\begin{array}{rcl} f^{-1}(\mathbb{Y}) & = & \{\beta, \gamma, \varepsilon\} \\ f^{-1}(\overline{\mathbb{Y}}) & = & \{\alpha\} \\ \text{dom} f & = & \{\alpha, \beta, \gamma, \varepsilon\} \end{array} \quad (13)$$

Now, since $f(\{\gamma, \delta\}) \subset \mathbb{Y}$, some would classify δ inside $f^{-1}(\mathbb{Y})$ which is wrong. This is be true if f is total.

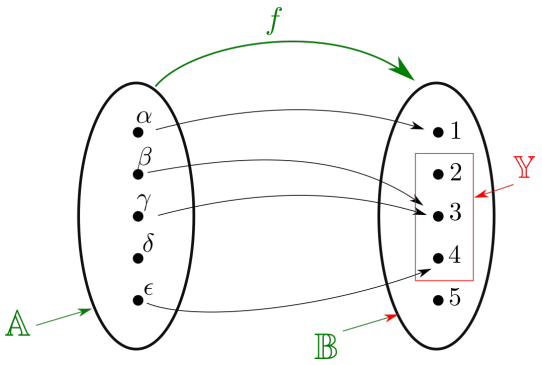

Fig. 2. A partial function f

Introducing the indeterminate *NaN* (Not a number), denoted by ι, in the sets allows us to get rid of the problem involved by the partiality of f.

We define the *extended total function* of the partial function f as

$$\mathring{f}(x) = \begin{cases} f(x) & \text{if } x \in \text{dom} f \\ \iota & \text{otherwise} \end{cases} \quad (14)$$

Given a set $\mathbb{A}$, we define the extended total set as $\mathring{\mathbb{A}} = \mathbb{A} \cup \{\iota\}$. Thus, $\mathring{f} : \mathring{\mathbb{A}} \mapsto \mathring{\mathbb{B}}$ is the extended total function of $f : \mathbb{A} \mapsto \mathbb{B}$ as illustrated by Figure 3. Extended functions can be extended to set as follows:

$$\mathring{f}(\mathbb{X}) = \begin{cases} f(\mathbb{X}) & \text{if } \mathbb{X} \subset \text{dom}(f) \\ f(\mathbb{X}) \cup \{\iota\} & \text{otherwise} \end{cases} \quad (15)$$

where $\mathbb{X} \subset \mathring{\mathbb{A}}$. Note that, in the figure, whereas $f(\{\gamma, \delta\}) = \{3\} \subset \mathbb{Y}$, we have $\mathring{f}(\{\gamma, \delta\}) = \{3, \iota\} \not\subset \mathbb{Y}$.

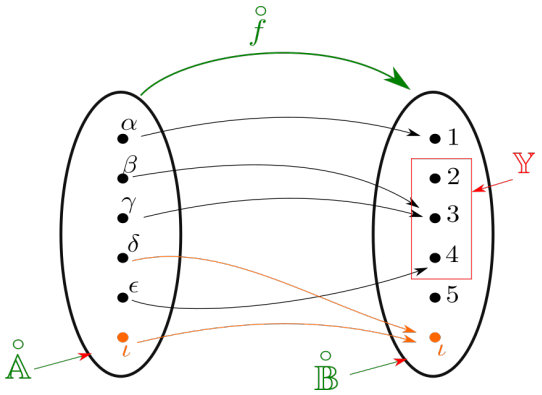

Fig. 3. Introduction of Not a Number ι

C. Properties

For total functions, we have some properties that will be useful in our algorithms

Proposition 2. *If $\mathring{f}$ is total we have*

$$\begin{array}{cc} \mathring{f}^{-1}(\overline{\mathbb{Y}}) = \overline{\mathring{f}^{-1}(\mathbb{Y})} & (i) \\ \mathring{f}(\mathbb{X}) \subset \mathbb{Y} \Rightarrow \mathbb{X} \subset \mathring{f}^{-1}(\mathbb{Y}) & (ii) \\ \mathring{f} \circ \mathring{f}^{-1}(\mathbb{Y}) = \mathbb{Y} & (iii) \end{array} \tag{16}$$

Proof: Let us prove (ii) only. We have:

$$\begin{array}{lcc} \mathring{f}(\mathbb{X}) \subset \mathbb{Y} & \Leftrightarrow & \mathring{f}(\mathbb{X}) \cap \overline{\mathbb{Y}} = \emptyset \\ & \Leftrightarrow & \underbrace{\mathring{f}^{-1} \circ \mathring{f}(\mathbb{X})}_{\supset \mathbb{X}} \cap \underbrace{\mathring{f}^{-1}(\overline{\mathbb{Y}})}_{= \overline{\mathring{f}^{-1}(\mathbb{Y})}} = \emptyset \\ & \Rightarrow & \mathbb{X} \cap \overline{\mathring{f}^{-1}(\mathbb{Y})} = \emptyset \\ & \Leftrightarrow & \mathbb{X} \subset \mathring{f}^{-1}(\mathbb{Y}) \end{array} \tag{17}$$

■

Proposition 3. *If $\mathring{f}$ and $\mathring{g}$ are total extension of f and g then the total extension of $f \circ g$ is $\mathring{f} \circ \mathring{g}$.*

Proof: If $h = f \circ g$, we have

$$\mathring{f} \circ \mathring{g} = \left\{ \begin{array}{cc} f \circ g(x) & \text{if } x \in dom(g) \text{ and } g(x) \in dom(f) \\ f(\iota) & \text{if } x \notin dom(g) \\ \iota & \text{if } g(x) \notin dom(h) \end{array} \right. \tag{18}$$

i.e.

$$\mathring{f} \circ \mathring{g} = \left\{ \begin{array}{cc} h(x) & \text{if } x \in dom(h) \\ \iota & \text{if } x \notin dom(h) \end{array} \right. \tag{19}$$

which corresponds to $\mathring{h}$. ■

D. Total real arithmetic

We define the total extension of the classical arithmetic on real numbers. Consider the extended total set of reals:

$$\mathring{\mathbb{R}} = \mathbb{R} \cup \iota. \tag{20}$$

Adding such a decoration for real numbers is now classical since it has been introduced by the IEEE 754 floating-point standard in 1985. Operations on real number can be extended to $\mathring{\mathbb{R}}$ as follows:

$$\begin{array}{ll} f(x) = \iota & \text{if } x \notin \text{dom}(f) \\ f(\iota) = \iota & \\ \iota \diamond x = \iota & \end{array} \tag{21}$$

where f is any partial function and $x \in \mathbb{R}$ and any binary operator $\diamond$.

Proposition 4. *If $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$ is a partial function given by an expression $\mathbf{f}(x_1, \ldots, x_n)$ including elementary functions (sin, $\sqrt{}$, log,...) and elementary operators ($+, -, /, \ldots$) then an expression for $\mathring{\mathbf{f}}$ can be obtained by the total real arithmetic.*

Proof: The proof is a direct consequence of the fact that the total extension is preserved by composition. ■

An element of the Cartesian product $\mathring{\mathbb{R}}^n = \mathring{\mathbb{R}} \times \cdots \times \mathring{\mathbb{R}}$ is called a *total vector*.

III. Total intervals

In this section, we introduce the notion of intervals for $\mathring{\mathbb{R}}$, called *total intervals*.

A. Intervals in unions of lattices

On a lattice $(\mathbb{A}, \leq_{\mathbb{A}})$,we can define the notion of intervals, interval hull, contractors. This has been used for several type of lattices such as real numbers, integers, trajectories, graphs, etc. To be able to use interval methods, the lattice structure is required. We show here that it is not strictly necessary by considering union of lattices.

Definition 5. Consider two lattices $(\mathbb{A}, \leq_{\mathbb{A}})$ and $(\mathbb{B}, \leq_{\mathbb{B}})$ that are disjoint. We can define intervals of $\mathbb{C} = \mathbb{A} \cup \mathbb{B}$ as subsets $\mathbb{C}$ which have the form

$$[c] = [a] \cup [b], \tag{22}$$

where $[a] \in \mathbb{I}\mathbb{A}$ and $[b] \in \mathbb{I}\mathbb{B}$.

Indeed, the set $(\mathbb{C}, \leq_{\mathbb{C}})$ can be equipped with an order relation:

$$x \leq_{\mathbb{C}} y \Leftrightarrow \left\{ \begin{array}{ll} & x \in \mathbb{A}, y \in \mathbb{A}, x \leq_{\mathbb{A}} y \\ \text{or} & x \in \mathbb{B}, y \in \mathbb{B}, x \leq_{\mathbb{B}} y \end{array} \right. \tag{23}$$

Now, $\mathbb{C}$ is not a lattice: if $x \in \mathbb{A}$, $y \in \mathbb{B}$ we cannot define $x \wedge y$ and $x \vee y$. This is due to the fact that we cannot provide a common lower or upper bounds for x, y.

Example 6. Consider the case where $\mathbb{A} = \mathbb{R}$ the set of real numbers and $\mathbb{B} = \{a, b, c, \ldots, z\}$ the set of letters. Both can be equipped with an order relation and are lattices. Examples of intervals for the set $\mathbb{C} = \mathbb{A} \cup \mathbb{B}$ are

$$\begin{array}{l} [c_1] = [2, 5] \\ [c_2] = \{e, f, g, h\} \\ [c_3] = [2, 5] \cup \{e, f, g, h\} \\ [c_4] = [4, 9] \cup \{g, h, i\} \\ [c_5] = \emptyset \\ [c_6] = \mathbb{A} \cup \mathbb{B} \end{array} \tag{24}$$

It is easy to check that the intervals of $\mathbb{C}$ is closed under intersection. It is thus a Moore family. As a consequence, contractor methods can be used.

B. Total intervals

Consider the singleton $\{\iota\}$ which is equipped with the trivial order relation : $\iota \leq \iota$. The set of all intervals of $\{\iota\}$ is $\{\emptyset, \{\iota\}\}$.

The set $\mathring{\mathbb{R}}$ can be equipped with a partial order relation $\leq_{\mathring{\mathbb{R}}}$ derived from $\mathbb{R}$:

$$\begin{array}{c} \iota \leq_{\mathring{\mathbb{R}}} \iota \\ a \in \mathbb{R}, b \in \mathbb{R} \quad \text{then} \quad a \leq_{\mathring{\mathbb{R}}} b \text{ iff } a \leq_{\mathbb{R}} b \end{array} \tag{25}$$

Total intervals are denoted by $[\mathring{x}]$.

Examples of intervals of $\mathring{\mathbb{R}}$ are:

$$\begin{array}{c} [\mathring{a}] = [1, \infty] \\ [\mathring{b}] = [-1, 0] \cup \{\iota\} \\ [\mathring{c}] = \{\iota\} \\ [\mathring{d}] = \emptyset \end{array} \tag{26}$$

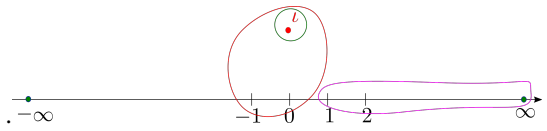

Fig. 4. Total intervals are intervals of $\mathring{\mathbb{R}} = \mathbb{R} \cup \{\iota\}$

as illustrated by Figure 4.

The set of total intervals is denoted by $\mathbb{I}\mathring{\mathbb{R}}$. We define the *hull* of a subset of $\mathring{\mathbb{X}}$ of $\mathring{\mathbb{R}}$ as the smallest total interval $[\mathring{x}]$ which encloses $\mathring{\mathbb{X}}$. We will write $[\mathring{x}] = [\![\mathring{\mathbb{X}}]\!]$. For instance

$$\begin{array}{rcl} [\![\{1,2,3\}]\!] & = & [1,3] \\ [\![\{1,2,3,\iota\}]\!] & = & [1,3] \cup \{\iota\} \\ [\![\{\iota\}]\!] & = & \{\iota\} \end{array} \tag{27}$$

C. Total interval arithmetic

Consider a partial function $f : \mathbb{R} \mapsto \mathbb{R}$. We define its *total interval extension* as follows

$$[\mathring{f}] = [\![\{\mathring{f}(\mathring{x}), \mathring{x} \in [\mathring{x}]\}]\!]. \tag{28}$$

For instance $\sqrt{[-1,4]} = [0,2] \cup \{\iota\}$.

In the same manner, if $\diamond \in \{+,-,\cdot,/\}$,we define

$$[\mathring{a}] \diamond [\mathring{b}] = [\![\{\mathring{a} \diamond \mathring{b}, \mathring{a} \in [\mathring{a}], \mathring{b} \in [\mathring{b}]\}]\!] \tag{29}$$

D. Total interval vector

The set of interval vectors $\mathring{\mathbb{R}}^n$ is a lattice . We can thus define intervals of $\mathring{\mathbb{R}}^n$. The set of interval vectors has the form $\mathbb{I}\mathring{\mathbb{R}}^n = \mathbb{I}\mathring{\mathbb{R}} \times \cdots \times \mathbb{I}\mathring{\mathbb{R}}$. We define the *hull* of a subset of $\mathring{\mathbb{X}}$ of $\mathring{\mathbb{R}}^n$ as the smallest $[\mathring{\mathbf{x}}]$ which encloses $\mathring{\mathbb{X}}$. We will write $[\mathring{\mathbf{x}}] = [\![\mathring{\mathbb{X}}]\!]$. For instance,

$$[\![([1,2] \times \{\iota\}) \cup ([3,4] \times [5,6])]\!] = [1,4] \times ([5,6] \cup \iota). \tag{30}$$

IV. Total contractors

This section extends the notion of contractor to total intervals. We first consider the case of elementary contractors built from elementary functions. Then, we consider the case of contractors defined from elementary operators.

A. Total directed contractor for a binary constraint

Consider a constraint of the form $y = f(x)$, where $f : \mathbb{R} \mapsto \mathbb{R}$: is a partial function with domain $dom\, f$. We can extend the constraint to $\mathring{\mathbb{R}}$ by the following decomposition

$$\left\{ \begin{array}{c} \mathring{y} = f(\mathring{x}) \\ \mathring{x} \in \mathring{\mathbb{R}} \\ \mathring{y} \in \mathring{\mathbb{R}} \end{array} \right. \Leftrightarrow \left\{ \begin{array}{cc} & \mathring{y} = f(\mathring{x}),\, \mathring{x} \in dom(f),\, \mathring{y} \in \mathbb{R} \\ \text{or} & \mathring{x} \in \mathbb{R} \setminus dom(f)\,,\ \mathring{y} = \iota \\ \text{or} & \mathring{x} = \iota,\, \mathring{y} = \iota \end{array} \right. \tag{31}$$

This means that $\iota = f(x)$ is considered as true only if and only if $x = \iota$ or is $x \notin dom(f)$. We define the *forward directional contractor*

$$\overrightarrow{\mathcal{C}_f}([\mathring{x}]) = [\![\{\mathring{y} \,|\, \exists \mathring{x} \in [\mathring{x}], \mathring{y} = f(\mathring{x})\}]\!] \tag{32}$$

and the *backward directional contractor*

$$\overleftarrow{\mathcal{C}_f}([\mathring{x}], [\mathring{y}]) = [\![\{\mathring{x} \in [\mathring{x}] \,|\, \exists \mathring{y} \in [\mathring{y}],\, \mathring{y} = f(\mathring{x})\}]\!] \tag{33}$$

Proposition 7. *The forward directional contractor associated with f is*

$$\overrightarrow{\mathcal{C}_f}([\mathring{x}]) = [\![f([\mathring{x}] \cap \mathbb{R})]\!] \cup ([\mathring{x}] \cap \{\iota\}) \cup \iota([\mathring{x}] \cap (\mathbb{R} \setminus dom(f))), \tag{34}$$

where ι is the the constant function ι, i.e,

$$\iota(\mathbb{A}) = \left\{ \begin{array}{ll} \iota & \text{if } \mathbb{A} \neq \emptyset \\ \emptyset & \text{if } \mathbb{A} = \emptyset \end{array} \right. \tag{35}$$

Proof: Since

$$\begin{array}{rcl} \mathring{x} \in dom\, f & \Rightarrow & \mathring{y} = f(\mathring{x}) \\ \mathring{x} = \iota & \Rightarrow & \mathring{y} = \iota \\ \mathring{x} \in \mathbb{R} \setminus dom\, f & \Rightarrow & \mathring{y} = \iota \end{array} \tag{36}$$

we have

$$f([\mathring{x}]) = \underbrace{f([\mathring{x}] \cap dom\, f)}_{f([\mathring{x}] \cap \mathbb{R})} \cup \underbrace{\iota([\mathring{x}] \cap \{\iota\})}_{=[\mathring{x}] \cap \{\iota\}} \cup \iota([\mathring{x}] \cap (\mathbb{R} \setminus dom\, f)). \tag{37}$$

Thus

$$\begin{array}{rcl} \overrightarrow{\mathcal{C}_f}([\mathring{x}]) & = & [\![\{\mathring{y} \,|\, \exists \mathring{x} \in [\mathring{x}], \mathring{y} = f(\mathring{x})\}]\!] \\ & = & [\![f([\mathring{x}] \cap \mathbb{R})]\!] \cup ([\mathring{x}] \cap \{\iota\}) \cup \iota([\mathring{x}] \cap (\mathbb{R} \setminus dom\, f)) \end{array} \tag{38}$$

■

As a consequence, the following algorithm implements $\overrightarrow{\mathcal{C}_f}([\mathring{x}])$:

Algorithm 2 Forward directional contractor $\overrightarrow{\mathcal{C}_f}$

	Input: $f, [\mathring{x}]$
1	$[\mathring{y}] = [\![f([\mathring{x}] \cap \mathbb{R})]\!]$
2	$[\mathring{y}] = [\mathring{y}] \cup ([\mathring{x}] \cap \{\iota\})$
3	if $[\mathring{x}] \not\subset dom\, f$, $[\mathring{y}] = [\mathring{y}] \cup \{\iota\}$
5	return $[\mathring{y}]$

Proposition 8. *The backward directional contractor associated with f is*

$$\overleftarrow{\mathcal{C}_f}([\mathring{x}], [\mathring{y}]) = [\mathring{x}] \cap \left([\![f^{-1}([\mathring{y}] \cap \mathbb{R})]\!] \cup \mathbb{I}([\mathring{y}]) \right) \tag{39}$$

where

$$\mathbb{I}([\mathring{y}]) = \left\{ \begin{array}{cc} \{\iota\} \cup (\mathbb{R} \setminus dom\, f) & \text{if } \iota \in [\mathring{y}] \\ \emptyset & \text{otherwize} \end{array} \right. \tag{40}$$

Proof: We have

$$\begin{array}{rcl} \mathring{y} \in \mathbb{R} & \Leftrightarrow & \mathring{x} \in f^{-1}(\{\mathring{y}\}) \\ \mathring{y} = \iota & \Leftrightarrow & (\mathring{x} = \iota) \vee (\mathring{x} \in \mathbb{R} \setminus dom\, f) \\ & \Leftrightarrow & \mathring{x} \in \{\iota\} \cup (\mathbb{R} \setminus dom\, f) \end{array} \tag{41}$$

■

As a consequence, the following algorithm implements $\overleftarrow{\mathcal{C}_f}([\mathring{x}], [\mathring{y}])$:

Algorithm 3 Backward directional contractor $\overleftarrow{\mathcal{C}_f}$

	Input: $f^{-1}, [\mathring{x}], [\mathring{y}]$
1	$[\mathring{r}] = \emptyset$
2	if $[\mathring{y}] = \emptyset$, return $[\mathring{r}]$
3	$[\mathring{r}] = [\![f^{-1}([\mathring{y}] \cap \mathbb{R})]\!]$
3	if $\iota \in [\mathring{y}]$, $[\mathring{r}] = [\mathring{r}] \cup (\mathbb{R} \setminus dom\, f) \cup \{\iota\}$
5	return $[\mathring{r}] \cap [\mathring{x}]$

Example 9. *Total contractor for the square root*. Consider the constraint

$$y = \sqrt{x} \tag{42}$$

where all variables belong to $\mathring{\mathbb{R}}$. The values $(9,3),(-4,\iota),(\iota,\iota)$ for (x,y) are consistent with the constraint (42) whereas $(9,2),(-4,2),(9,\iota),(\iota,2)$ are inconsistent.

For instance, assume that we have $x \in [\mathring{x}] = [-2,9], y \in [\mathring{y}] = [-1,2] \cup \{i\}$. We obtain

$$\begin{array}{lcl} \overrightarrow{\mathcal{C}_{\sqrt{\cdot}}}([\mathring{x}]) & = & \sqrt{[-2,9]} = [0,3] \cup \{i\} \\ \overrightarrow{\mathcal{C}_{\sqrt{\cdot}}}([\mathring{x}]) \cap [\mathring{y}] & = & ([0,3] \cup \{i\}) \cap ([-1,2] \cup \{i\}) = [0,2] \cup \{i\} \\ \overleftarrow{\mathcal{C}_{\sqrt{\cdot}}}([\mathring{x}],[\mathring{y}]) & = & [-2,4] \end{array} \tag{43}$$

It means that $x \in [-2,4]$ and $y \in [0,2] \cup \{i\}$.

Assume now that $x \in [\mathring{x}] = [4,9], y \in [\mathring{y}] = [3,15] \cup \{i\}$. We obtain

$$\begin{array}{lcl} \overrightarrow{\mathcal{C}_{\sqrt{\cdot}}}([\mathring{x}]) & = & \sqrt{[4,9]} = [2,3] \\ \overrightarrow{\mathcal{C}_{\sqrt{\cdot}}}([\mathring{x}]) \cap [\mathring{y}] & = & [2,3] \cap ([3,15] \cup \{i\}) = \{3\} \\ \overleftarrow{\mathcal{C}_{\sqrt{\cdot}}}([\mathring{x}],[\mathring{y}]) & = & \{9\} \end{array} \tag{44}$$

It means that $x = 9$ and $y = 3$.

B. Total directed contractor for a ternary constraint

Consider the ternary constraint of $z = x + y$. The case of constraints involving $-, \cdot, /$ can be defined from $+$ and binary constraints already treated in the previous section. The following reasoning can also be done for these operators.

We can extend the constraint $z = x+y$ to $\mathring{\mathbb{R}}$ by the following decomposition

$$\left\{ \begin{array}{l} \mathring{z} = \mathring{x} + \mathring{y} \\ \mathring{x} \in \mathring{\mathbb{R}} \\ \mathring{y} \in \mathring{\mathbb{R}} \\ \mathring{z} \in \mathring{\mathbb{R}} \end{array} \right. \Leftrightarrow \left\{ \begin{array}{ll} & \mathring{z} = \mathring{x} + \mathring{y},\ \mathring{x} \in \mathbb{R}, \mathring{y} \in \mathbb{R},\ \mathring{z} \in \mathbb{R} \\ \text{or} & (\mathring{x} = \iota \vee \mathring{y} = \iota) \wedge \mathring{z} = \iota \end{array} \right. \tag{45}$$

Note that in $\mathring{\mathbb{R}}$, we do not have

$$\mathring{z} = \mathring{x} + \mathring{y} \Leftrightarrow \mathring{x} = \mathring{z} - \mathring{y}. \tag{46}$$

Indeed, take $\mathring{x} = 1, \mathring{y} = \iota, \mathring{z} = \iota$. We have $\mathring{z} = \mathring{x} + \mathring{y}$ whereas $\mathring{x} \neq \mathring{z} - \mathring{y}$. As a consequence, the values $(2,3,5),(2,\iota,\iota),(\iota,\iota,\iota)$ for (x,y,z) are consistent with the constraint whereas $(2,3,6),((2,\iota,4),(2,3,\iota)$ are inconsistent.

We define the forward directed contractor

$$\overrightarrow{\mathcal{C}_+}([\mathring{x}],[\mathring{y}]) = [\![\{\mathring{z} \,|\, \exists \mathring{x} \in [\mathring{x}], \exists \mathring{y} \in [\mathring{y}], \mathring{z} = \mathring{x} + \mathring{y}\}]\!] \tag{47}$$

and the backward directed contractor

$$\overleftarrow{\mathcal{C}_+}([\mathring{x}],[\mathring{y}],[\mathring{z}]) = [\![\{(\mathring{x},\mathring{y}) \in [\mathring{x}] \times [\mathring{y}] \,|\, \exists \mathring{z} \in [\mathring{z}],\ \mathring{z} = \mathring{x} + \mathring{y}\}]\!] \tag{48}$$

If we apply the same reasoning as for the binary constraint and we get the following algorithms for $\overrightarrow{\mathcal{C}_\oplus}$ and $\overleftarrow{\mathcal{C}_\oplus}$

Algorithm 4 Forward directed contractor $\overrightarrow{\mathcal{C}_+}$

	Input: $[\mathring{x}], [\mathring{y}]$
1	$\overrightarrow{\mathcal{C}_f}([\mathring{x}]) = ([\mathring{x}] \cap \mathbb{R}) + ([\mathring{y}] \cap \mathbb{R})$
2	$[\mathring{z}] = [\mathring{z}] \cup ([\mathring{x}] \cap \{\iota\}) \cup ([\mathring{y}] \cap \{\iota\})$
3	return $[\mathring{z}]$

Step 1 computes to the interval containing of all feasible $z \in \mathbb{R}$.

Step 2 adds ι when $\iota \in [\mathring{x}]$ of when $\iota \in [\mathring{y}]$.

Algorithm 5 the Backward directed contractor $\overleftarrow{\mathcal{C}_+}$

	Input: $[\mathring{x}], [\mathring{y}], [\mathring{z}]$
1	if $[\mathring{z}] \cap \{\iota\} = \emptyset$ then
2	$[\mathring{x}] = [\mathring{x}] \cap ([\mathring{z}] - [\mathring{y}])$
3	$[\mathring{y}] = [\mathring{y}] \cap ([\mathring{z}] - [\mathring{x}])$
4	return $[\mathring{x}], [\mathring{y}]$

C. Total forward-backward contractor

We show how the forward-backward contractor works on two small examples. Il faut donc en rajouter un.

Example 1. Consider the set

$$\mathbb{S} = \{(x,y) \,|\, y + \sqrt{x+y} \in [1,2]\}. \tag{49}$$

We built the AST (Abstract Syntax Tree) associated with $\mathbb{S}$ as shown in Figure 5(a). We also build the AST for $\overline{\mathbb{S}}$ as in Figure 5(b). Note that the two trees are identical except the images that are complementary in $\mathring{\mathbb{R}}$, i.e.,

$$[1,2] \cup ([-\infty,1] \cup [2,\infty] \cup \{\iota\}) = \mathring{\mathbb{R}}. \tag{50}$$

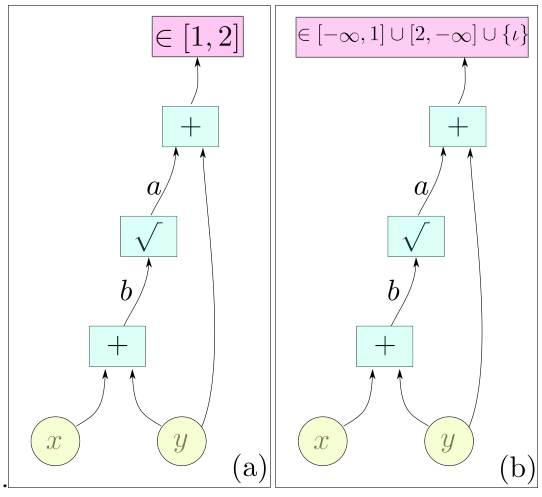

Fig. 5. AST for the constraint $y + \sqrt{x+y} \in [1,2]$ (left) and its complementary (right)

A forward-backward contractor yields the following algorithms

Algorithm 6 Contractor for the constraint $y + \sqrt{x+y} \in \mathbb{Z}$

	Input: $[\mathring{x}], [\mathring{y}], \mathbb{Z}$
1	$[\mathring{b}] = \overrightarrow{\mathcal{C}_+}([\mathring{x}], [\mathring{y}])$
2	$[\mathring{a}] = \overrightarrow{\mathcal{C}_{\sqrt{}}}([\mathring{b}])$
3	$[\mathring{z}] = \overrightarrow{\mathcal{C}_+}([\mathring{a}], [\mathring{y}])$
4	$[\mathring{z}] = [\mathring{z}] \cap \mathbb{Z}$
5	$[\mathring{a}], [\mathring{y}] = \overleftarrow{\mathcal{C}_+}([\mathring{z}], [\mathring{a}], [\mathring{y}])$
6	$[\mathring{b}] = \overrightarrow{\mathcal{C}_{\sqrt{}}}([\mathring{b}], [\mathring{a}])$
7	$[\mathring{x}], [\mathring{y}] = \overleftarrow{\mathcal{C}_+}([\mathring{b}], [\mathring{x}], [\mathring{y}])$
8	return $[\mathring{x}], [\mathring{y}]$

To have a contractor for $\mathbb{S}$ we call Algorithm IV-C with $\mathbb{Z} = [1, 2]$. To get a contractor for $\overline{\mathbb{S}}$, we call the algorithm with $\mathbb{Z} = [-\infty, 1] \cup [2, , \infty] \cup \{\iota\}$. Using a paver with these two contractors, we are able to generate the approximation illustrated by Figure 6. The frame box is $[-10, 10] \times [-10, 10]$.

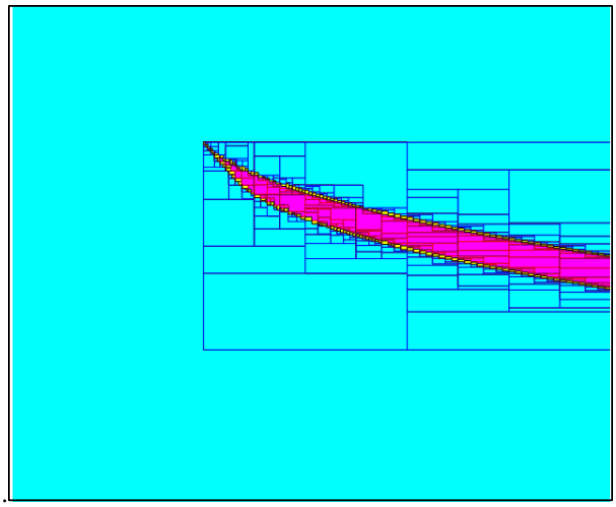

Fig. 6. Paving obtained using the contractor for $\mathbb{S}$ and its complementary

An implementation is given at: https://replit.com/@aulin/iota

Example 2. Consider the discrete-time state space system of the form $\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k))$ where

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} \sqrt{x^2 + \log x + 1} \\ \log(x - \sqrt{x+2}) \end{pmatrix}. \tag{51}$$

We want to compute the set of all initial vector such that $\mathbf{x}(0)$ leads to an undefined state when $k = 5$. We define

$$\begin{array}{rcl} \mathbf{f}^0(\mathbf{x}) & = & \mathbf{x} \\ \mathbf{f}^{k+1}(\mathbf{x}) & = & \mathbf{f}^k \circ \mathbf{f}(\mathbf{x}) \end{array} \tag{52}$$

And we compute

$$\mathbb{X}_0 = (\mathbf{f}^5)^{-1}(\{\iota\}). \tag{53}$$

.

V. Conclusion

In this paper, we have proposed to extend the interval arithmetic developed by Moore [6] in order to facilitate the implementation of complementary of contractors. For this purpose, we proposed to add a flag ι to each interval to form *total intervals*. The associated arithmetic has been derived. The flag ι has similarities with some decorations already used in the context of interval computation [4]. The main advantage of our extension is to allow the interval propagation when some partial functions are involved in the definition of constraints. We have presented a generalization of the forward-backward propagation to total intervals. The efficiency has been illustrated on two examples.

References

[1] F. Benhamou, F. Goualard, L. Granvilliers, and J. F. Puget. Revising hull and box consistency. In *Proceedings of the International Conference on Logic Programming*, pages 230–244, Las Cruces, NM, 1999.

[2] G. Chabert. *IBEX 2.0, available at* , `http://www.emn.fr/z-info/ibex/`. Ecole des mines de Nantes, 2013.

[3] G. Chabert and L. Jaulin. Contractor Programming. *Artificial Intelligence*, 173:1079–1100, 2009.

[4] D. Defour, G. Hanrot, V. Lefevre, J.M. Muller, N. Revol, and P. Zimmermann. Proposal for a standardization of mathematical function implementation in floating-point arithmetic. *Numerical algorithms*, 37(4):367–375, 2004.

[5] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics*. Springer-Verlag, London, 2001.

[6] R.E. Moore, R.B. Kearfott, and M.J. Cloud. *Introduction to Interval Analysis*. SIAM, Philadelphia, PA, 2009.