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 the complement operator. The reason for this is probably related to the interval arithmetic itself. In this paper, we show that if 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

Interval analysis [11] is a numerical tool used to solve nonlinear problems such as non convex optimization [8] or solving nonlinear equations [13]. In control or robotics, it is often needed to compute inner and outer approximations for sets [9] [16].

The algorithms we use to characterize a set $\mathbb{X}$ are pavers that classify areas of the search space using contractors [4]. A *contractor* $\mathcal{C}$ for the set $\mathbb{X} \subset \mathbb{R}^n$ is an operator $\mathbb{IR}^n \mapsto \mathbb{IR}^n$ which satisfies

$$\begin{array}{lll} \mathcal{C}([\mathbf{x}]) \subset [\mathbf{x}] & & \text{(contractance)} \\ [\mathbf{x}] \subset [\mathbf{y}] \;\Rightarrow\; \mathcal{C}([\mathbf{x}]) \subset \mathcal{C}([\mathbf{y}]). & & \text{(monotonicity)} \\ \mathcal{C}([\mathbf{x}]) \cap \mathbb{X} = [\mathbf{x}] \cap \mathbb{X} & & \text{(consistency)} \end{array} \quad (1)$$

where $\mathbb{IR}^n$ is the set of axis-aligned boxes of $\mathbb{R}^n$. The paver bisects boxes and uses $\mathcal{C}$ to eliminate parts of the search space that are outside $\mathbb{X}$. In this paper, sets $\mathbb{X}$ of $\mathbb{R}^n$ will be represented in *mathbb* font and intervals $[x]$ or boxes $[\mathbf{x}]$ within brackets.

If $\mathcal{C}_1$ and $\mathcal{C}_2$ are two contractors, we define the following operations on contractors:

$$(\mathcal{C}_1 \cap \mathcal{C}_2)([\mathbf{x}]) = \mathcal{C}_1([\mathbf{x}]) \cap \mathcal{C}_2([\mathbf{x}]) \quad (2)$$

$$(\mathcal{C}_1 \cup \mathcal{C}_2)([\mathbf{x}]) = \mathcal{C}_1([\mathbf{x}]) \sqcup \mathcal{C}_2([\mathbf{x}]) \quad (3)$$

where $[\mathbf{a}] \sqcup [\mathbf{b}]$ is the smallest box which contains both $[\mathbf{a}]$ and $[\mathbf{b}]$.

We also use the contractors to eliminate parts that are inside the solution set, but we observe some problems in existing solvers as soon as the domains of the functions involved in the problem are restricted. We now illustrate thus on a simple example.

Consider the set

$$\mathbb{X} = \{(x_1, x_2) \mid x_2 + \sqrt{x_1 + x_2} \in [1, 2]\}. \quad (4)$$

and let us try to compute an inner and outer approximations of $\mathbb{X}$ using an existing solver. For instance, if we use Codac [15] with the following script:

```
 from codac import *
from vibes import *
X0=IntervalVector([[-10,10],[-10,10]])
f = Function("x1","x2","x2+sqrt(x1+x2)")
S=SepFwdBwd(f,sqr(Interval(1,2)))
vibes.beginDrawing()
SIVIA(X0,S,0.01)
```

We get the paving illustrated by Figure 1 where the blue boxes are proved to be outside $\mathbb{X}$ and the magenta boxes are supposed to be inside. We observe that this is not the case. Indeed some boxes are wrongly classified as inside whereas they are outside. This phenomenon occurs for all existing solvers which are able to provide an inner approximation. The reasons for this is that contractor-based methods obtain an inner approximation by considering a contractor for the complementary of $\mathbb{X}$ as

$$\{(x_1, x_2) \mid x_2 + \sqrt{x_1 + x_2} \notin [1, 2]\} \quad (5)$$

whereas it should be

$$\overline{\mathbb{X}} = \{(x_1, x_2) \mid x_2 + \sqrt{x_1 + x_2} \notin [1, 2] \text{ or } x_1 + x_2 < 0\}. \quad (6)$$

In the figure, some magenta zones are wrongly classified as inside because in these zones, $\sqrt{x_1 + x_2}$ is not defined. The goal of this paper is to provide a rigorous way to build contractors associated with the complementary of a set in the case where functions involved in the constraint are not defined everywhere.

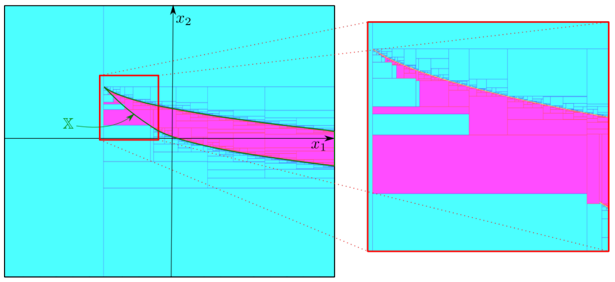

Fig. 1. Left: Paving obtained by classical methods to approximate $\mathbb{X}$; Right: A zoom on the red box

The paper is organized as follows. Section II explains the approach that will motivate a new arithmetic. Section III 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 IV introduces the notion of the total interval arithmetic. Section IV provides the notion of total contractors and extends the classical forward-backward contractor to total intervals. Section VI concludes the paper.

II. Approach

Contractor algebra as defined in [4] 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

$$\forall i, \mathcal{C}_i \subset \mathcal{C}_i^{'} \Rightarrow \mathcal{E}\left(\mathcal{C}_1, \mathcal{C}_2, \dots\right) \subset \mathcal{E}\left(\mathcal{C}_1^{'}, \mathcal{C}_2^{'}, \dots\right). \tag{7}$$

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

$$\mathbb{X} = \mathbb{X}_1 \cup \mathbf{f}^{-1}(\mathbb{X}_2 \cap \mathbb{X}_3). \tag{8}$$

We can represent its expression by the tree of Figure 2 or equivalently by the following expressions

$$\begin{array}{rcl} \mathbb{X} & = & \mathbb{X}_1 \cup \mathbb{B} \\ \mathbb{B} & = & \mathbf{f}^{-1}(\mathbb{A}) \\ \mathbb{A} & = & \mathbb{X}_2 \cap \mathbb{X}_3 \end{array} \tag{9}$$

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 reverse* (or equivalently *inequality constraints*) of the form

$$\mathbb{X}_i = \varphi_i^{-1}([\mathbf{y}_i]) = \{\mathbf{x}_i \mid \varphi_i(\mathbf{x}_i) \in [\mathbf{y}_i]\} \tag{10}$$

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 $\mathcal{C}^{\updownarrow}_{\varphi^{-1}([\mathbf{y}])}$.

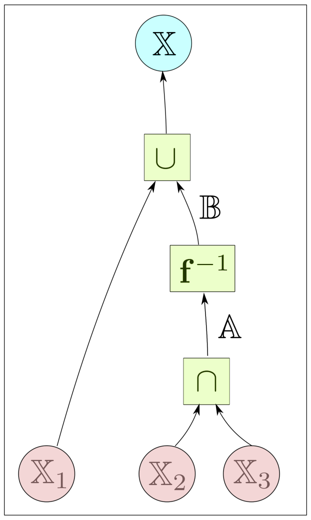

Fig. 2. Contractor tree for $\mathbb{X}_1 \cup \mathbf{f}^{-1}(\mathbb{X}_2 \cap \mathbb{X}_3)$

Once the contractor for $\mathbb{X}$ is built from the tree, a paver [9] 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 (8) 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 [3] 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

$$\overline{\mathbb{X}} = \overline{\mathbb{X}}_1 \cap \left(\mathbf{f}^{-1}(\overline{\mathbb{X}}_2 \cup \overline{\mathbb{X}}_3) \cup \overline{\mathrm{dom}\mathbf{f}}\right). \tag{11}$$

Note that we had to introduce the domain of $\mathbf{f}$, denoted by $\mathrm{dom}\mathbf{f}$, to take into account the fact that $\mathbf{f}$ may be a partial function (*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

$$\mathring{\mathbf{f}}^{-1}(\mathbb{Y}) = \mathbf{f}^{-1}(\mathbb{Y}) \cup \overline{\mathrm{dom}\mathbf{f}}, \tag{12}$$

then we have

$$\overline{\mathbb{X}} = \overline{\mathbb{X}}_1 \cap \left(\mathring{\mathbf{f}}^{-1}(\overline{\mathbb{X}}_2 \cup \overline{\mathbb{X}}_3)\right). \tag{13}$$

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

$$\begin{array}{rcl} \overline{\mathbb{X}} & = & \overline{\mathbb{X}}_1 \cap \overline{\mathbb{B}} \\ \overline{\mathbb{B}} & = & \mathring{\mathbf{f}}^{-1}(\overline{\mathbb{A}}) \\ \overline{\mathbb{A}} & = & \overline{\mathbb{X}}_2 \cup \overline{\mathbb{X}}_3 \end{array} \tag{14}$$

which corresponds to the tree of Figure 3.

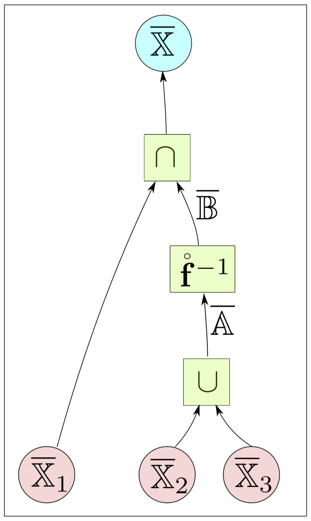
$\overline{\mathbb{X}}$
$\cap$
$\overline{\mathbb{B}}$
$\mathring{\mathbf{f}}^{-1}$
$\overline{\mathbb{A}}$
$\cup$
$\overline{\mathbb{X}}_1$
$\overline{\mathbb{X}}_2$
$\overline{\mathbb{X}}_3$

Fig. 3. Contractor tree for the complementary of $\mathbb{X}_1 \cup \mathbf{f}^{-1}(\mathbb{X}_2 \cap \mathbb{X}_3)$

Since the sets $\mathbb{X}_i$ were defined by $\varphi_i(\mathbf{x}_i) \in [\mathbf{y}_i]$, the complement is by

$$\overline{\mathbb{X}}_i = \mathring{\varphi}_i^{-1}(\overline{[\mathbf{y}_i]}). \tag{15}$$

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

$$\mathring{\mathbf{f}}^{-1}([\mathbf{y}]) = \mathbf{f}^{-1}([\mathbf{y}]) \cup \overline{\text{dom}\mathbf{f}} \tag{16}$$

Now, the set $\mathring{\mathbf{f}}^{-1}([\mathbf{y}])$ is not a set reverse as defined by (10) and thus we cannot apply a forward-backward contractor without an extension which will be proposed in this paper.

III. 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

$$\forall x \in X, \exists y \in Y \text{ such that } f(x) = y. \tag{17}$$

A function f which is not defined for all x is said to be *partial*. Given a partial function f, the *total extension* is obtained by adding an element to Y, say ι which collects all $x \notin \text{dom}f$. 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

$$\mathring{f}(x) = \left\{ \begin{array}{ll} f(x) & \text{if } x \in \text{dom}f \\ \iota & \text{otherwise} \end{array} \right. \tag{18}$$

Note that since $\iota \notin \text{dom}f$, we have $\mathring{f}(\iota) = \iota$.

B. Illustration

Consider the partial function f as given in Figure 4. We have

$$\begin{array}{rcl} f^{-1}(\mathbb{Y}) & = & \{\beta, \gamma, \varepsilon\} \\ f^{-1}(\overline{\mathbb{Y}}) & = & \{\alpha\} \\ \text{dom}f & = & \{\alpha, \beta, \gamma, \varepsilon\} \end{array} \tag{19}$$

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

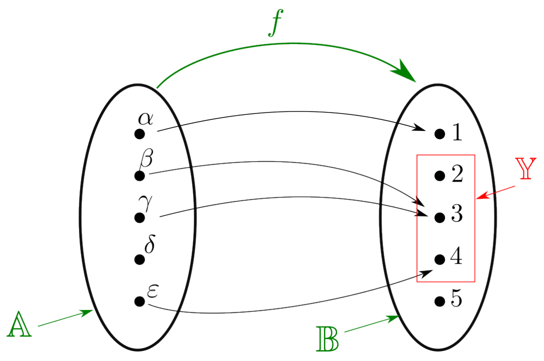

Fig. 4. A partial function f

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.

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 total extension of $f : \mathbb{A} \mapsto \mathbb{B}$ as illustrated by Figure 5. Following Definition 1, extended functions can be used to set as follows:

$$\mathring{f}(\mathbb{X}) = \left\{ \begin{array}{ll} f(\mathbb{X}) & \text{if } \mathbb{X} \subset \text{dom}f \\ f(\mathbb{X}) \cup \{\iota\} & \text{otherwise} \end{array} \right. \tag{20}$$

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

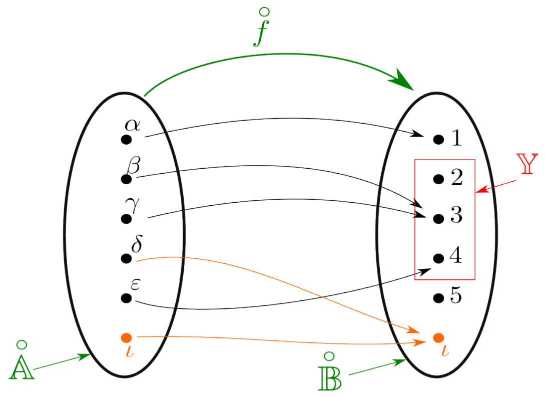

Fig. 5. Introduction of Not a Number ι

C. Properties

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

Proposition 1. *If $\mathring{f}$ is a total extension of f, we have*

$$\begin{array}{lr} \mathring{f}^{-1}(\overline{\mathbb{Y}}) = \overline{\mathring{f}^{-1}(\mathbb{Y})} & (i) \\ \mathring{f}(\mathbb{X}) \subset \mathbb{Y} \Rightarrow \mathbb{X} \subset \mathring{f}^{-1}(\mathbb{Y}) & (ii) \\ \mathring{f} \circ \mathring{f}^{-1}(\mathbb{Y}) = \mathbb{Y} & (iii) \end{array} \quad (21)$$

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

$$\begin{array}{lll} \mathring{f}(\mathbb{X}) \subset \mathbb{Y} & \Leftrightarrow & \mathring{f}(\mathbb{X}) \cap \overline{\mathbb{Y}} = \emptyset \\ & \Leftrightarrow & \underbrace{\mathring{f}^{-1} \circ \mathring{f}(\mathbb{X})}_{\supset \mathbb{X}} \cap \underbrace{\mathring{f}^{-1}(\overline{\mathbb{Y}})}_{=\overline{\mathring{f}^{-1}(\mathbb{Y})}} = \emptyset \\ & \Rightarrow & \mathbb{X} \cap \overline{\mathring{f}^{-1}(\mathbb{Y})} = \emptyset \\ & \Leftrightarrow & \mathbb{X} \subset \mathring{f}^{-1}(\mathbb{Y}) \end{array} \quad (22)$$

■

Proposition 2. *If $\mathring{f}$, $\mathring{g}$ are total extensions of f, g then the total extension of $f \circ g$ is $\mathring{f} \circ \mathring{g}$.*

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

$$\mathring{f} \circ \mathring{g}(x) = \begin{cases} f \circ g(x) & \text{if } x \in \text{dom}g \text{ and } g(x) \in \text{dom}f \\ \iota & \text{otherwise} \end{cases} \quad (23)$$

Now, since

$$\text{dom}h = \text{dom}f \circ g = \text{dom}g \cap g^{-1}(\text{dom}f) \quad (24)$$

We get

$$\mathring{f} \circ \mathring{g}(x) = \begin{cases} h(x) & \text{if } x \in \text{dom}h \\ \iota & \text{if } x \notin \text{dom}h \end{cases} \quad (25)$$

which corresponds to $\mathring{h}(x)$. ■

Example. To illustrate the proposition, take

$$\begin{array}{lcl} f(x) & = & \sqrt{1-x} \\ g(x) & = & \sqrt{x-1} \end{array} \quad (26)$$

Note that

$$\begin{array}{lcl} \text{dom}f & = & (-\infty, 1] \\ \text{dom}g & = & [1, \infty) \end{array} \quad (27)$$

We have

$$h(x) = f \circ g(x) = \sqrt{1 - \sqrt{x-1}} \quad (28)$$

Since

$$\begin{array}{lcl} \text{dom}h & = & \text{dom}g \cap g^{-1}(\text{dom}f) \\ & = & [1, \infty) \cap g^{-1}((-\infty, 1]) \\ & = & [1, \infty) \cap [0, 2] = [1, 2] \end{array} \quad (29)$$

we get

$$\mathring{h}(x) = \begin{cases} \sqrt{1 - \sqrt{x-1}} & \text{if } x \in [1, 2] \\ \iota & \text{otherwise} \end{cases} \quad (30)$$

D. Total real arithmetic

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

$$\mathring{\mathbb{R}} = \mathbb{R} \cup \{\iota\}. \quad (31)$$

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

$$\begin{array}{ll} f(x) = \iota & \text{if } x \notin \text{dom}(f) \\ f(\iota) = \iota & \\ \iota \diamond x = \iota & \end{array} \quad (32)$$

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

Note that we do not define comparisons, which means that if we have the relation $a \leq b$ then both a and b belong to $\mathbb{R}$ (or equivalently neither a nor b can be equal to ι).

Proposition 3. *Consider a partial function $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$ given by an expression $\mathbf{f}(x_1, \dots, x_n)$ including elementary functions (sin, $\sqrt{}$, log,...) and elementary operators $(+, -, *, /, \dots)$. 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.*

E. Link with the complex number i

The set of complex numbers $\mathbb{C}$ extends the set of real numbers by adding a number i such that $i^2 = -1$. The extension preserves some properties such that the fact $\mathbb{C}$ is a group with respect to the addition. Due to this, i has an opposite: $-i$. Indeed, $i + (-i) = 0$.

Take now the set $\mathring{\mathbb{R}}$ and let us check if ι has an opposite. We solve $\iota + x = 0$ and we get no solution for x. This means that $\mathring{\mathbb{R}}$ is not anymore a group and many properties we had for $\mathbb{R}$ are lost. As a consequence, symbolic resolution and group-based simplifications are not allowed in $\mathring{\mathbb{R}}$.

Moreover, adding i to build $\mathbb{C}$ involves the addition of many numbers of the form $a+ib$. In $\mathring{\mathbb{R}}$, we just add a single number: ι.

There exists a tiny link between $\mathring{\mathbb{R}}$ and $\mathbb{C}$ in the construction since we add one number. But the link stops here. Whereas complex numbers can be used to build a huge numbers of theorems and theories, the total numbers will be used as a tool to build the complementary of contractors.

IV. 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 and contractors. This has been used for several types 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. Consider two lattices $(\mathbb{A}, \leq_{\mathbb{A}})$ and $(\mathbb{B}, \leq_{\mathbb{B}})$ that are disjoint. Denote by $\mathbb{IA}$ and $\mathbb{IB}$, the set of all intervals of $\mathbb{A}$ and $\mathbb{B}$. We can define intervals of $\mathbb{C} = \mathbb{A} \cup \mathbb{B}$ as subsets $\mathbb{C}$ which have the form

$$[c] = [a] \cup [b], \quad (33)$$

where $[a] \in \mathbb{IA}$ and $[b] \in \mathbb{IB}$.

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

$$x \leq_{\mathbb{C}} y \Leftrightarrow \left\{ \begin{array}{ll} & x \in \mathbb{A}, y \in \mathbb{A}, x \leq_{\mathbb{A}} y \\ \text{or} & x \in \mathbb{B}, y \in \mathbb{B}, x \leq_{\mathbb{B}} y \end{array} \right. \quad (34)$$

Now, $\mathbb{C}$ is not a lattice, *i.e.*, 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. 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 both are lattices. Examples of intervals for the set $\mathbb{C} = \mathbb{A} \cup \mathbb{B}$ are

$$\begin{array}{l} [c_1] = [2,5] \\ [c_2] = \{e, f, g, h\} \\ [c_3] = [2,5] \cup \{e, f, g, h\} \\ [c_4] = [4,9] \cup \{g, h, i\} \\ [c_5] = \emptyset \\ [c_6] = \mathbb{A} \cup \mathbb{B} \end{array} \quad (35)$$

It is easy to check that the intervals of $\mathbb{C}$ is closed under intersection. It is thus a Moore family [10][2]. 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 $\mathbb{R}$:

$$\begin{array}{c} \iota \leq_{\mathring{\mathbb{R}}} \iota \\ a \in \mathbb{R}, b \in \mathbb{R} \quad \text{then} \quad a \leq_{\mathring{\mathbb{R}}} b \text{ iff } a \leq_{\mathbb{R}} b \end{array} \quad (36)$$

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

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

$$\begin{array}{c} [\mathring{a}] = [1, \infty) \\ [\mathring{b}] = [-1, 0] \cup \{\iota\} \\ [\mathring{c}] = \{\iota\} \\ [\mathring{d}] = \emptyset, \end{array} \quad (37)$$

as illustrated by Figure 10.

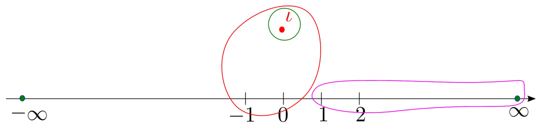

Fig. 6. Total intervals are intervals of $\mathring{\mathbb{R}} = \mathbb{R} \cup \{\iota\}$

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

$$\begin{array}{rcl} [\![\{1,2,3\}]\!] & = & [1,3] \\ [\![\{1,2,3,\iota\}]\!] & = & [1,3] \cup \{\iota\} \\ [\![\{\iota\}]\!] & = & \{\iota\}. \end{array} \quad (38)$$

C. Total interval arithmetic

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

$$[\mathring{f}] = [\![\{\mathring{f}(\mathring{x}), \mathring{x} \in [\mathring{x}]\}]\!]. \quad (39)$$

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

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

$$[\mathring{a}] \diamond [\mathring{b}] = [\![\{\mathring{a} \diamond \mathring{b}, \mathring{a} \in [\mathring{a}], \mathring{b} \in [\mathring{b}]\}]\!] \quad (40)$$

D. Total interval vector

The set of interval vectors $\mathring{\mathbb{R}}^n$ is a lattice [6]. 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,

$$[\![([1,2] \times \{\iota\}) \cup ([3,4] \times [5,6])]\!] = [1,4] \times ([5,6] \cup \{\iota\}). \quad (41)$$

V. 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

$$\left\{ \begin{array}{c} \mathring{y} = f(\mathring{x}) \\ \mathring{x} \in \mathring{\mathbb{R}} \\ \mathring{y} \in \mathring{\mathbb{R}} \end{array} \right. \Leftrightarrow \left\{ \begin{array}{lc} & \mathring{y} = f(\mathring{x}),\ \mathring{x} \in \text{dom}f,\ \mathring{y} \in \mathbb{R} \\ \text{or} & \mathring{x} \in \mathbb{R} \setminus \text{dom}f\,,\ \mathring{y} = \iota \\ \text{or} & \mathring{x} = \iota,\ \mathring{y} = \iota \end{array} \right. \quad (42)$$

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

$$\overrightarrow{\mathcal{C}_f}([\mathring{x}]) = [\![\{\mathring{y} \mid \exists \mathring{x} \in [\mathring{x}], \mathring{y} = f(\mathring{x})\}]\!] \quad (43)$$

and the *backward directional contractor*

$$\overleftarrow{\mathcal{C}_f}([\mathring{x}], [\mathring{y}]) = [\![\{\mathring{x} \in [\mathring{x}] \mid \exists \mathring{y} \in [\mathring{y}],\ \mathring{y} = f(\mathring{x})\}]\!]. \quad (44)$$

Proposition 4. *The forward directional contractor associated with f is*

$$\overrightarrow{\mathcal{C}_f}([\mathring{x}]) = [\![f([\mathring{x}] \cap \mathbb{R})]\!] \cup ([\mathring{x}] \cap \{\iota\}) \cup \iota([\mathring{x}] \cap (\mathbb{R} \setminus \textit{dom}f)), \quad (45)$$

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

$$\iota(\mathbb{A}) = \left\{ \begin{array}{ll} \iota & \textit{if } \mathbb{A} \neq \emptyset \\ \emptyset & \textit{if } \mathbb{A} = \emptyset \end{array} \right. \quad (46)$$

Proof: Since

$$\begin{array}{rcl} \mathring{x} \in \text{dom}f & \Rightarrow & \mathring{y} = f(\mathring{x}) \\ \mathring{x} = \iota & \Rightarrow & \mathring{y} = \iota \\ \mathring{x} \in \mathbb{R} \setminus \text{dom}f & \Rightarrow & \mathring{y} = \iota \end{array} \quad (47)$$

we have

$$f([\mathring{x}]) = \underbrace{f([\mathring{x}] \cap \text{dom } f)}_{f([\mathring{x}]\cap\mathbb{R})} \cup \underbrace{\iota([\mathring{x}] \cap \{\iota\})}_{=[\mathring{x}]\cap\{\iota\}} \cup \iota([\mathring{x}] \cap (\mathbb{R} \setminus \text{dom} f)). \tag{48}$$

Thus

$$\begin{array}{rcl} \overrightarrow{\mathcal{C}_f}([\mathring{x}]) & = & [\![\{\mathring{y} \mid \exists \mathring{x} \in [\mathring{x}], \mathring{y} = f(\mathring{x})\}]\!] \\ & = & [\![f([\mathring{x}] \cap \mathbb{R})]\!] \cup ([\mathring{x}] \cap \{\iota\}) \\ & & \cup\, \iota([\mathring{x}] \cap (\mathbb{R} \setminus \text{dom } f)) \end{array} \tag{49}$$

■

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 $[\mathring{x}] \not\subset \text{dom } f$, $[\mathring{y}] = [\mathring{y}] \cup \{\iota\}$
4  return $[\mathring{y}]$
```

Proposition 5. *The backward directional contractor associated with f is*

$$\overleftarrow{\mathcal{C}_f}([\mathring{x}], [\mathring{y}]) = [\mathring{x}] \cap \left([\![f^{-1}([\mathring{y}] \cap \mathbb{R})]\!] \cup \mathbb{I}([\mathring{y}])\right) \tag{50}$$

where

$$\mathbb{I}([\mathring{y}]) = \begin{cases} \{\iota\} \cup (\mathbb{R} \setminus \textit{dom } f) & \textit{if } \iota \in [\mathring{y}] \\ \emptyset & \textit{otherwise} \end{cases} \tag{51}$$

Proof: We have

$$\begin{array}{rcc} \mathring{y} \in \mathbb{R} & \Leftrightarrow & \mathring{x} \in f^{-1}(\{\mathring{y}\}) \\ \mathring{y} = \iota & \Leftrightarrow & (\mathring{x} = \iota) \vee (\mathring{x} \in \mathbb{R} \setminus \text{dom } f) \\ & \Leftrightarrow & \mathring{x} \in \{\iota\} \cup (\mathbb{R} \setminus \text{dom } f) \end{array} \tag{52}$$

■

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  $[\mathring{r}] = [\![f^{-1}([\mathring{y}] \cap \mathbb{R})]\!]$
4  if $\iota \in [\mathring{y}]$, $[\mathring{r}] = [\mathring{r}] \cup (\mathbb{R} \setminus \text{dom } f) \cup \{\iota\}$
5  return $[\mathring{r}] \cap [\mathring{x}]$
```

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

$$y = \sqrt{x} \tag{53}$$

where all variables belong to $\mathring{\mathbb{R}}$. The values $(9,3), (-4,\iota), (\iota,\iota)$ for (x,y) are consistent with the constraint (53) whereas $(9,2), (-4,2), (9,\iota), (\iota,2)$ are inconsistent.

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

$$\begin{array}{rcl} \overrightarrow{\mathcal{C}_{\sqrt{\cdot}}}([\mathring{x}]) & = & \sqrt{[-2,9]} = [0,3] \cup \{\iota\} \\ \overrightarrow{\mathcal{C}_{\sqrt{\cdot}}}([\mathring{x}]) \cap [\mathring{y}] & = & ([0,3] \cup \{\iota\}) \cap ([-1,2] \cup \{\iota\}) \\ & = & [0,2] \cup \{\iota\} \\ \overleftarrow{\mathcal{C}_{\sqrt{\cdot}}}([\mathring{x}], [\mathring{y}]) & = & [-2,4] \end{array} \tag{54}$$

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

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

$$\begin{array}{rcl} \overrightarrow{\mathcal{C}_{\sqrt{\cdot}}}([\mathring{x}]) & = & \sqrt{[4,9]} = [2,3] \\ \overrightarrow{\mathcal{C}_{\sqrt{\cdot}}}([\mathring{x}]) \cap [\mathring{y}] & = & [2,3] \cap ([3,15] \cup \{\iota\}) = \{3\} \\ \overleftarrow{\mathcal{C}_{\sqrt{\cdot}}}([\mathring{x}], [\mathring{y}]) & = & \{9\} \end{array} \tag{55}$$

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

$$\left\{ \begin{array}{c} \mathring{z} = \mathring{x} + \mathring{y} \\ \mathring{x} \in \mathring{\mathbb{R}} \\ \mathring{y} \in \mathring{\mathbb{R}} \\ \mathring{z} \in \mathring{\mathbb{R}} \end{array} \right. \Leftrightarrow \left\{ \begin{array}{cc} & \mathring{z} = \mathring{x} + \mathring{y},\, \mathring{x} \in \mathbb{R}, \mathring{y} \in \mathbb{R},\, \mathring{z} \in \mathbb{R} \\ \text{or} & (\mathring{x} = \iota \vee \mathring{y} = \iota) \wedge \mathring{z} = \iota \end{array} \right. \tag{56}$$

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

$$\mathring{z} = \mathring{x} + \mathring{y} \Leftrightarrow \mathring{x} = \mathring{z} - \mathring{y}. \tag{57}$$

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

$$\overrightarrow{\mathcal{C}_+}([\mathring{x}], [\mathring{y}]) = [\![\{\mathring{z} \mid \exists \mathring{x} \in [\mathring{x}], \exists \mathring{y} \in [\mathring{y}], \mathring{z} = \mathring{x} + \mathring{y}\}]\!] \tag{58}$$

and the backward directed contractor

$$\overleftarrow{\mathcal{C}_+}([\mathring{x}], [\mathring{y}], [\mathring{z}]) = [\![\{(\mathring{x}, \mathring{y}) \in [\mathring{x}] \times [\mathring{y}] \mid \exists \mathring{z} \in [\mathring{z}],\, \mathring{z} = \mathring{x} + \mathring{y}\}]\!] \tag{59}$$

We get Algorithms 4 and 5 for $\overrightarrow{\mathcal{C}_+}$ and $\overleftarrow{\mathcal{C}_+}$.

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

```
   Input: $[\mathring{x}], [\mathring{y}]$
1  $[\mathring{z}] = ([\mathring{x}] \cap \mathbb{R}) + ([\mathring{y}] \cap \mathbb{R})$
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 $\iota \in [\mathring{x}]$ or when $\iota \in [\mathring{y}]$.

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

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

The implementation for $\overleftarrow{\mathcal{C}_+}$ is simplified by the fact that it is called after $\overrightarrow{\mathcal{C}_+}$.

Remark. The contractor $\overleftarrow{\mathcal{C}_+}$ is often minimal, but not always. Indeed, there exist some rare counterexamples. Consider for instance the case

$$\begin{array}{c} x \in [1,2] \\ y \in [3,4] \cup \{\iota\} \\ z \in [6,9] \cup \{\iota\} \end{array} \tag{60}$$

If we call $\overrightarrow{\mathcal{C}_+}$, we get

$$\begin{array}{c} x \in [1,2] \\ y \in [3,4] \cup \{\iota\} \\ z \in [6,6] \cup \{\iota\} \end{array} \tag{61}$$

The backward contractor $\overleftarrow{\mathcal{C}_+}$ (see Algorithm 5) yields no contraction for x and y whereas it should conclude the following contraction for y :

$$y \in [4,4] \cup \{\iota\}.$$

An optimal backward contractor could be obtained by the following algorithm:

```
    Input: $[\mathring{x}], [\mathring{y}], [\mathring{z}]$
1   $[\mathring{r}_x] = \emptyset$,$[\mathring{r}_y] = \emptyset$
2   $[x] = [\mathring{x}] \cap \mathbb{R}$; $[y] = [\mathring{y}] \cap \mathbb{R}$; $[z] = [\mathring{z}] \cap \mathbb{R}$;
3   if $[x] \neq \emptyset, [y] \neq \emptyset, [z] \neq \emptyset$ then
4       $[\mathring{r}_x] = [x] \cap ([z] - [y])$;$[\mathring{r}_y] = [y] \cap ([z] - [x])$
5   $[y] = [\mathring{y}] \cap \{\iota\}$; $[z] = [\mathring{z}] \cap \{\iota\}$;
6   if $[x] \neq \emptyset, [y] \neq \emptyset, [z] \neq \emptyset$ then
7       $[\mathring{r}_y] = [\mathring{r}_y] \cup \{\iota\}$
8   $[y] = [\mathring{y}] \cap \mathbb{R}$; $[x] = [\mathring{x}] \cap \{\iota\}$; $[z] = [\mathring{z}] \cap \{\iota\}$;
9   if $[x] \neq \emptyset, [y] \neq \emptyset, [z] \neq \emptyset$ then
10      $[\mathring{r}_x] = [\mathring{r}_x] \cup \{\iota\}$
11  $[x] = [\mathring{x}] \cap \{\iota\}$; $[y] = [\mathring{y}] \cap \{\iota\}$; $[z] = [\mathring{z}] \cap \{\iota\}$;
12  if $[x] \neq \emptyset, [y] \neq \emptyset, [z] \neq \emptyset$ then
13      $[\mathring{r}_x] = [\mathring{r}_x] \cup \{\iota\}$; $[\mathring{r}_y] = [\mathring{r}_y] \cup \{\iota\}$
14  return $[\mathring{r}_x], [\mathring{r}_y]$
```

Now, this algorithm improves the efficiency of a propagation only for rare situations. This is why we will preferred the use of the backward contractor of Algorithm 5, even if not fully minimal.

C. Total forward-backward contractor

We show how the forward-backward contractor works on two test-cases.

Test-case 1. Consider the set

$$\mathbb{S} = \{(x,y) \,|\, y + \sqrt{x+y} \in [1,2]\}. \tag{62}$$

We built the AST (Abstract Syntax Tree) associated with $\mathbb{S}$ as shown in Figure 7(a). We also build the AST for $\overline{\mathbb{S}}$ as in Figure 7(b). Note that the two trees are identical except the images that are complementary in $\mathring{\mathbb{R}}$, i.e.,

$$[1,2] \cup ((-\infty,1] \cup [2,\infty) \cup \{\iota\}) = \mathring{\mathbb{R}}. \tag{63}$$

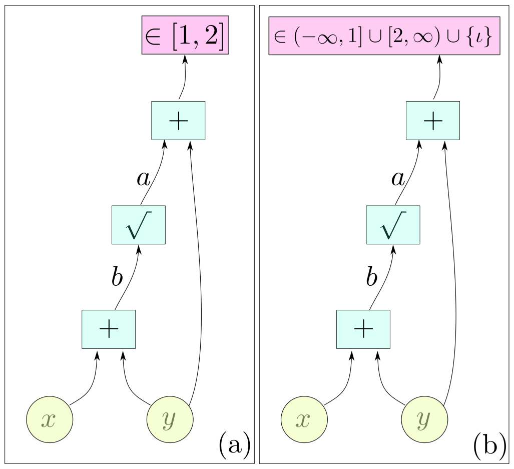

Fig. 7. AST for the constraint $y + \sqrt{x+y} \in [1,2]$ (left) and its complementary (right)

A forward-backward contractor yields Algorithm 6. Note that below the set $\mathbb{Z}$ is not the set of integers (as often used in math books), but an interval of $\mathbb{R}$.

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  $[\mathring{b}] = \overleftarrow{\mathcal{C}_{\sqrt{}}}([\mathring{b}], [\mathring{a}])$
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 6 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 8. The frame box is $[-10,10] \times [-10,10]$.

An implementation is given in [7].

Test-case 2. Consider the discrete-time state space system, inspired from Henon map, of the form $\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k))$ with

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} bx_1 \\ 1 + ax_1^2 + \sqrt{x_2^2 + c} \end{pmatrix} \tag{64}$$

where $a = -1.4$, $b = 0.3$ and $c = 0.075$. The behavior of this system may not lead to a well defined state $\mathbf{x}(k)$ if the initial state vector $\mathbf{x}(0)$ is not chosen properly. We want to compute

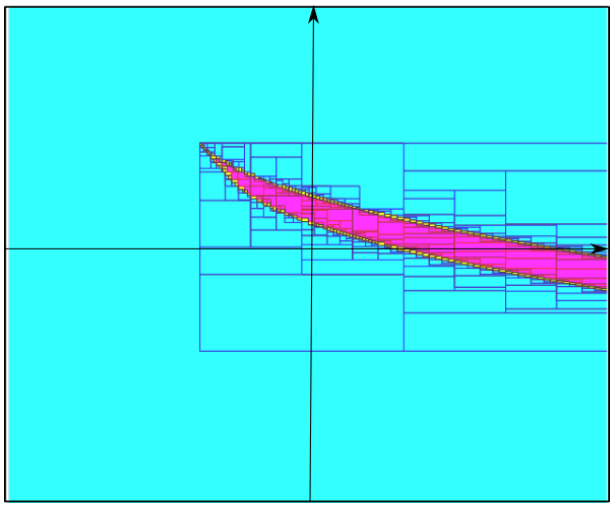

Fig. 8. Paving obtained using the contractor for $\mathbb{S}$ and its complementary

the set $\mathbb{S}$ of all initial vectors $\mathbf{x}(0)$ which lead to a state vector of $\mathbb{R}^2$ when $k = 5$. We define

$$\begin{array}{rcl} \mathbf{f}^0(\mathbf{x}) & = & \mathbf{x} \\ \mathbf{f}^{k+1}(\mathbf{x}) & = & \mathbf{f}^k \circ \mathbf{f}(\mathbf{x}). \end{array} \quad (65)$$

We have

$$\mathbb{S} = \{\mathbf{x} \in \mathbb{R}^2 \,|\, \mathbf{f}^5(\mathbf{x}) \in \mathbb{R}^2\}. \quad (66)$$

The forward backward contractor (see Algorithm 7) associated to the constraint $\mathbf{f}(\mathbf{x}) \in \mathbb{Y}$ is based on the AST of Figure 9.

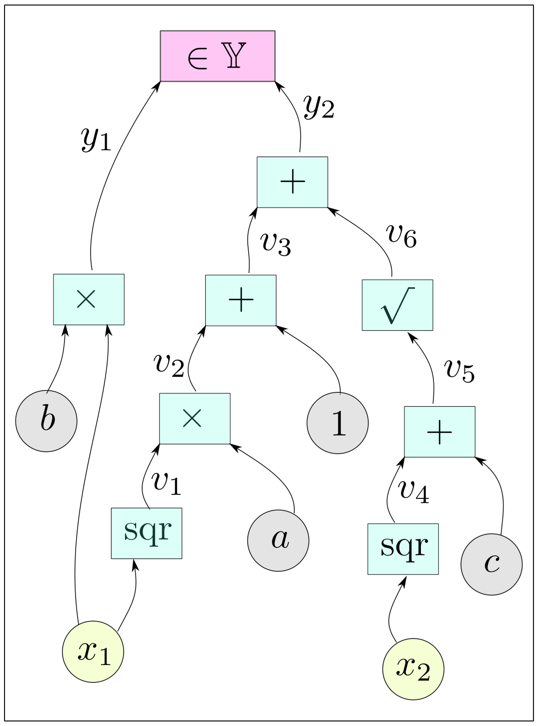

Fig. 9. AST for the constraint $\mathbf{f}(\mathbf{x}) \in \mathbb{Y}$

Algorithm 7 Contractor for a constraint $\mathbf{f}(\mathbf{x}) \in \mathbb{Y}$ for Test-case 2

```
    Input: $[\mathring{x}_1], [\mathring{x}_2], \mathbb{Y}$
1   $[\mathring{a}] = -1.4;\ [\mathring{b}] = 0.3;\ [\mathring{c}] = -0.075$
2   $[\mathring{v}_1] = \overrightarrow{\mathcal{C}_{sqr}}([\mathring{x}_1])$
3   $[\mathring{v}_2] = \overrightarrow{\mathcal{C}_{\times}}([\mathring{a}], [\mathring{v}_1])$
4   $[\mathring{v}_3] = 1 + [\mathring{v}_2]$
5   $[\mathring{v}_4] = \overrightarrow{\mathcal{C}_{sqr}}([\mathring{x}_2])$
6   $[\mathring{v}_5] = \overrightarrow{\mathcal{C}_{+}}([\mathring{v}_4], [\mathring{c}])$
7   $[\mathring{v}_6] = \overrightarrow{\mathcal{C}_{\surd}}([\mathring{v}_5])$
8   $[\mathring{y}_1] = \overrightarrow{\mathcal{C}_{\times}}([\mathring{x}_1], [\mathring{b}])$
9   $[\mathring{y}_2] = \overrightarrow{\mathcal{C}_{+}}([\mathring{v}_3], [\mathring{v}_6])$
10  $[\mathring{y}_1] \times [\mathring{y}_2] = [\![([\mathring{y}_1] \times [\mathring{y}_2]) \cap \mathbb{Y}]\!]$
11  $[\mathring{v}_3], [\mathring{v}_6] = \overleftarrow{\mathcal{C}_{+}}([\mathring{y}_2], [\mathring{v}_3], [\mathring{v}_6])$ (see Step 9)
12  $[\mathring{x}_1], [\mathring{b}] = \overleftarrow{\mathcal{C}_{\times}}([\mathring{y}_1], [\mathring{x}_1], [\mathring{b}])$ (see Step 8)
13  $[\mathring{v}_5] = \overleftarrow{\mathcal{C}_{\surd}}([\mathring{v}_5], [\mathring{v}_6])$ (see Step 7)
14  $[\mathring{v}_4], [\mathring{c}] = \overleftarrow{\mathcal{C}_{+}}([\mathring{v}_5], [\mathring{v}_4], [\mathring{c}])$ (see Step 6)
15  $[\mathring{x}_2] = \overleftarrow{\mathcal{C}_{sqr}}([\mathring{v}_4], [\mathring{x}_2])$ (see Step 5)
16  $[\mathring{v}_2] = ([\mathring{v}_3] - 1) \cap [\mathring{v}_2]$ (see Step 4)
17  $[\mathring{a}], [\mathring{v}_1] = \overleftarrow{\mathcal{C}_{\times}}([\mathring{v}_2], [\mathring{a}], [\mathring{v}_1])$ (see Step 3)
18  $[\mathring{x}_1] = \overleftarrow{\mathcal{C}_{sqr}}([\mathring{x}_1], [\mathring{v}_1])$ (see Step 2)
19  return $[\mathring{x}_1], [\mathring{x}_2]$
```

To have a contractor for $\mathbb{S}$ we call Algorithm 7 with $\mathbb{Y} = \mathbb{R}^2$. To get a contractor for $\overline{\mathbb{S}}$, we call the algorithm with $\mathbb{Y} = (\mathbb{R} \times \{\iota\}) \cup (\{\iota\} \times \mathbb{R}) \cup (\{\iota\} \times \{\iota\})$ which is the complementary of $\mathbb{R}^2$ in $(\mathbb{R} \cup \{\iota\})^2$ Using a paver with these two complementary contractors, we are able to generate the approximation illustrated by Figure 10. The frame box is $[-5, 5] \times [-5, 5]$.

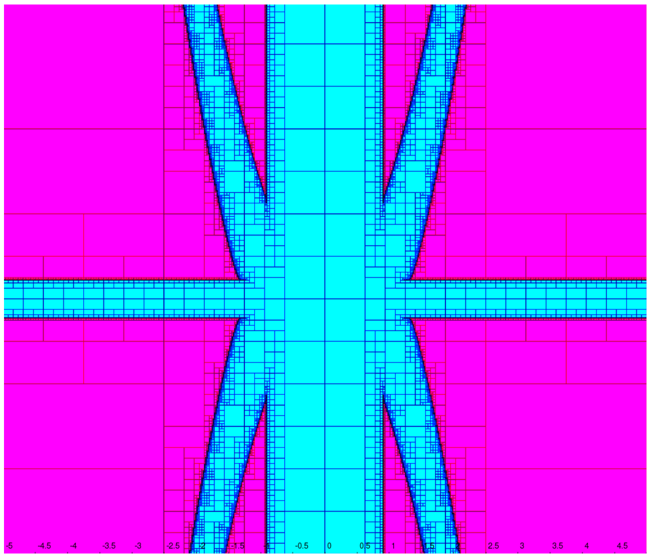

Fig. 10. Paving representing the solution set for Test-case 2

VI. Conclusion

In this paper, we have proposed to extend the interval arithmetic developed by Moore [12] 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. In our our new interval arithmetic, we have $\sqrt{[-1,1]} = [0,1] \cup \{\iota\}$ instead of $\sqrt{[-1,1]} = [0,1]$. This is due to the fact that $\sqrt{\cdot}$, which is a partial function, has been made total. The ι number is not seen anymore as an exception, but as a possible value.

The flag ι has similarities with some decorations already used in the context of interval computation [14], [5]. The main advantage of our extension is to allow the interval propagation when some partial functions are involved in the definition of the constraints. We have presented a generalization of the forward-backward propagation to total intervals. The efficiency has been illustrated on two test-cases.

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] T.S. Blyth. *Lattices and Ordered Algebraic Structures*. Springer, ISBN 1-85233-905-5, 2005.

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

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

[5] IEEE Microprocessor Standards Commitee. IEEE 1788-2015 Standard for Interval Arithmetic. *https://standards.ieee.org/ieee/1788/4431/*, 2015.

[6] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, (ISBN 0521784514), 2002.

[7] P. Filiol, T. Bollengier, L. Jaulin, and J.C. Le Lann. Codes associated with the paper entitled: A new interval arithmetic to generate the complementary of contractors. `www.ensta-bretagne.fr/jaulin/iota.html`, 2022.

[8] E. R. Hansen. *Global Optimization using Interval Analysis*. Marcel Dekker, New York, NY, 1992.

[9] 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.

[10] E.H. Moore. *Introduction to a form of general analysis*, volume 2. Yale University Press, 1910.

[11] R. Moore. *Methods and Applications of Interval Analysis*. Society for Industrial and Applied Mathematics, jan 1979.

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

[13] A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge University Press, Cambridge, UK, 1990.

[14] N. Revol. Introduction to the IEEE 1788-2015 Standard for Interval Arithmetic. *10th International Workshop on Numerical Software Verification - NSV 2017.*

[15] S. Rohou. *Codac (Catalog Of Domains And Contractors), available at* `http://codac.io/`. Robex, Lab-STICC, ENSTA-Bretagne, 2021.

[16] S. Rohou, L. Jaulin, L. Mihaylova, F. Le Bars, and S. Veres. *Reliable Robot Localization*. Wiley, dec 2019.