Universally Quantified Interval Constraints

Frédéric Benhamou and Frédéric Goualard

Institut de Recherche en Informatique de Nantes
2, rue de la Houssinière, B.P. 92208, F-44322 Nantes Cedex 3
{benhamou,goualard}@irin.univ-nantes.fr

Abstract. Non-linear real constraint systems with universally and/or existentially quantified variables often need be solved in such contexts as control design or sensor planning. To date, these systems are mostly handled by computing a quantifier-free equivalent form by means of Cylindrical Algebraic Decomposition (CAD). However, CAD restricts its input to be conjunctions and disjunctions of polynomial constraints with rational coefficients, while some applications such as camera control involve systems with arbitrary forms where time is the only universally quantified variable. In this paper, the handling of universally quantified variables is first related to the computation of inner-approximation of real relations. Algorithms for solving non-linear real constraint systems with universally quantified variables are then presented along with the theoretical framework on inner-approximation of relations supporting them. These algorithms are based on the computation of outer-approximations of the solution set of the negation of involved constraints. An application to the devising of a declarative modeller for expressing camera motion using a cinematic language is sketched, and results from a prototype are presented.

1 Introduction

Interval constraint-based solvers such as clp(BNR) [8], ILOG Solver [18], and Numerica [24] have been shown to be efficient tools for solving some challenging non-linear constraint systems in various application areas (*e.g.* robotics, chemistry [16], or electronics [19]). Relying on *interval arithmetic* [17], these tools guarantee *completeness* (all solutions in the input are retained), and permit isolating punctual solutions with an "arbitrary" accuracy. Taking as input a constraint system and a Cartesian product of domains (intervals) for the variables occurring in the constraints, their output is a set $\mathcal{S}_o$ of boxes approximating each solution contained in the input box.

However, *soundness* is not guaranteed while it is sometimes a strong requirement. Consider, for instance, a civil engineering problem [20] such as floor design where retaining non-solution points may lead to a physically unfeasible structure. As pointed out by Ward *et al.* [25] and Shary [23], one may expect different properties from the boxes composing $\mathcal{S}_o$ depending on the problem at hand, namely: every element in any box is a solution, or there exists at least

R. Dechter (Ed.): CP 2000, LNCS 1894, pp. 67–82, 2000.

one solution in each box. The foregoing solvers ensure only, at best, the second property.

Furthermore, problems originating from *camera control* [12], *sensor planning* [1], and *control design* [2], not only require the output boxes to contain only solution points, but also that some input variables be universally quantified.

To date, constraint systems with universally/existentially quantified variables have mainly be handled by symbolic methods, among which one may single out Cylindrical Algebraic Decomposition (CAD) by Collins [11]. CAD is quite a powerful method since it permits handling more than one quantified variable for disjunctions/conjunctions of constraints. However, it has strong requirements on the form of the constraints it processes since they are limited to polynomial constraints. As far as camera control is concerned though, only one quantified variable—time—needs to be dealt with for conjunctions of (not necessarily polynomial) constraints. Consequently, there is room for a tailored algorithm to specifically solve temporal constraints.

This paper first presents an algorithm whose output is a set of sound boxes of variable domains for some constraint system. Soundness is achieved by computing *inner approximations* of the underlying real relations, using *box consistency* [7]—a well-known, efficient, *local consistency*—on the negation of the involved constraints. An algorithm is then applied to the solving of constraint systems where one variable is universally quantified. An application to temporal constraints arising from camera motion modelling (*virtual cameraman problem* [14]) is sketched: following the work of Jardillier and Languénou, a prototype for a declarative modeller has been devised, which should eventually allow a non-technician user to control the positioning of a camera by means of cinematic specifications of a *shot* (short "scene").

The outline of the paper is as follows: Section 2 introduces notations and some basic notions related to interval constraint solving: interval representation of real quantities, approximation of relations by supersets, and local consistencies are surveyed. Next, the notion of *inner approximation* of real relations is formally introduced in Section 3, and then related to the solving of constraints containing occurrences of universally quantified variables; the corresponding algorithms are given and compared to a previous approach by Jardillier and Languénou. The prototype of a declarative modeller for camera positioning is presented in Section 4; heuristics for speeding-up computation, along with results on some benchmarks are then given. Finally, Section 5 compares our approach with previous works in the field, and discusses directions for future researches.

2 Interval Constraint Solving

Finite representation of numbers by computers prevents them solving accurately real constraints. *Interval constraint solving* relies on *interval arithmetic* [17] to compute verified approximate solutions of real constraint systems. Underlying real relations may be approximated by considering one of their computer-

representable superset or subset. This section presents the basics related to the approximation of real relations the conservative way. Safe approximation by a subset is deferred until the next section.

The organization of the section is as follows: the shift from reals to *bounds* (numbers together with a "bracket") is first described; the notion of open and closed interval relying on bounds is then introduced, followed by a presentation of the way real relations are approximated and interval constraint systems solved.

The reader is referred to the above-mentionned references for a thorough presentation of interval arithmetic. A great part of what is exposed in the following is drawn from [8] and [4]. Proofs not given here may be found in these papers.

2.1 Approximation of a Relation

Solving real constraints requires the ability to represent the underlying relations. The approximate representation by a superset (Cartesian product of intervals) is described in the following.

From Reals to Floating-Point Intervals Let $\mathbb{R}$ be the set of reals and $\mathbb{F} \subset \mathbb{R}$ a finite subset of reals corresponding to *floating-point numbers*. Sets $\mathbb{R}$ and $\mathbb{F}$ are compactified in the usual way by using symbols $-\infty$ and $+\infty$. Let $\mathbb{F}^\infty = \mathbb{F} \cup \{-\infty, +\infty\}$. Hereafter, r and s (resp. g and h), possibly subscripted, are assumed to be elements of $\mathbb{R}$ (resp. $\mathbb{F}^\infty$).

Four new symbols (*brackets*) are introduced: let $\mathcal{L} = \{(, [\}$ and $\mathcal{U} = \{),]\}$ be respectively the set of left and right brackets. Let $\mathcal{B} = \mathcal{L} \cup \mathcal{U}$ be the *set of brackets* totally ordered by the ordering $\prec$ defined as follows [9]: $) \prec [\prec] \prec ($.

The set of *floating-point bounds* $\mathbb{F}^\diamond$ is defined from $\mathcal{B}$ and $\mathbb{F}$ as follows:

$$\mathbb{F}^\diamond = \mathbb{F}^\triangleleft \cup \mathbb{F}^\triangleright \quad \text{where} \quad \begin{cases} \mathbb{F}^\triangleleft = (\mathbb{F} \times \mathcal{L} \cup \{\langle -\infty, (\,\rangle, \langle +\infty, (\,\rangle\}) \\ \mathbb{F}^\triangleright = (\mathbb{F} \times \mathcal{U} \cup \{\langle -\infty,)\,\rangle, \langle +\infty,)\,\rangle\}) \end{cases}$$

Real bounds set $\mathbb{R}^\diamond$ is defined likewise. Given a bound $\beta = \langle\, x, \alpha\,\rangle$, let $\beta|_v = x$ and $\beta|_b = \alpha$. Floating-point bounds are totally ordered by the ordering $\lhd$: $\forall \beta_1 = \langle\, \mathsf{g}, \alpha_1\,\rangle, \beta_2 = \langle\, \mathsf{h}, \alpha_2\,\rangle \in \mathbb{F}^\diamond$: $\beta_1 \lhd \beta_2 \iff (\mathsf{g} < \mathsf{h}) \vee (\mathsf{g} = \mathsf{h} \wedge \alpha_1 \prec \alpha_2)$. A similar ordering may be defined over $\mathbb{R}^\diamond$.

For each $\mathsf{g} \in \mathbb{F}^\infty$, let g^+ be the smallest element in $\mathbb{F}^\infty$ greater than g, and g^- the greatest element in $\mathbb{F}^\infty$ smaller than g (with the IEEE 754 conventions: $(+\infty)^+ = +\infty$, $(-\infty)^- = -\infty$, $(+\infty)^- = \max(\mathbb{F})$, $(-\infty)^+ = \min(\mathbb{F})$).

Bounds are used to construct intervals as follows: let $\mathbb{I}_\circ = \mathbb{F}^\triangleleft \times \mathbb{F}^\triangleright$ be the set of *closed/open floating-point intervals* (henceforth referred to *intervals*), with the following notations used as shorthands: $\big(\langle\, \mathsf{g}, [\,\rangle, \langle\, \mathsf{h},]\,\rangle\big) \equiv [\mathsf{g} \,..\, \mathsf{h}] \equiv \{\mathsf{r} \in \mathbb{R} \mid \mathsf{g} \leqslant \mathsf{r} \leqslant \mathsf{h}\}$, $\big(\langle\, \mathsf{g}, [\,\rangle, \langle\, \mathsf{h},)\,\rangle\big) \equiv [\mathsf{g} \,..\, \mathsf{h}) \equiv \{\mathsf{r} \in \mathbb{R} \mid \mathsf{g} \leqslant \mathsf{r} < \mathsf{h}\}$, and so on.

For the sake of simplicity, the empty set $\varnothing$ is uniquely represented in $\mathbb{I}_\circ$ by the interval $(+\infty \,..\, -\infty)$. Let $\mathbb{I}_\square \subset \mathbb{I}_\circ$ be the set of *closed intervals*, together with the two special intervals: $(-\infty \,..\, +\infty)$ and $(+\infty \,..\, -\infty)$.

In the rest of the paper, interval quantities are written uppercase, reals or floats are sans-serif lowercase, and vectors are in boldface. A Cartesian product

of n intervals $\mathbf{B} = I_1 \times \cdots \times I_n$ is called a *box*. A non-empty interval $I = (\beta_1, \beta_2)$ with $\beta_1 \in \mathbb{F}^{\triangleleft}$ and $\beta_2 \in \mathbb{F}^{\triangleright}$ is said *canonical* whenever $\beta_2|_v \leqslant (\beta_1|_v)^+$. An n-ary box $\mathbf{B}$ is canonical whenever the intervals $I_1, \ldots, I_n$ are canonical. Given an interval $I = (\beta_1, \beta_2)$, let $\inf(I) = \beta_1|_v$ and $\sup(I) = \beta_2|_v$. Given a variable v, an interval I, and boxes $\mathbf{B}$ and $\mathbf{D}$, let $\mathsf{Dom}_{\mathbf{B}}(v) \in \mathbb{I}_\circ$ be the domain of v in box $\mathbf{B}$, $\mathbf{B}|_k = I_k$, the *k-th projection of* $\mathbf{B}$, and $\mathbf{B}|_{v,\mathbf{D}}$ (resp. $\mathbf{B}|_{v,I}$) the box obtained by replacing v's domain in box $\mathbf{B}$ by its domain in box $\mathbf{D}$ (resp. by interval I); given an interval J, let $\mathbf{B}|_{I_k,J}$ be the box $I_1 \times \cdots \times I_{k-1} \times J \times I_{k+1} \times \cdots \times I_n$. Given boxes $\mathbf{B}$ and $\mathbf{D}$, let $\mathbf{B} \setminus \mathbf{D} \subseteq \mathcal{P}(\mathbb{I}_\circ^n)$ be the set of boxes obtained from complementing $\mathbf{B}$ from $\mathbf{D}$.

Approximating a Relation by a Box This section introduces some more notations on constraints, sets, and relations, then presents the notion of *outer approximation*, *viz.* the approximation of a real relation by a computer-representable superset.

Let $\mathcal{V}_{\mathbb{R}} = \{x_1, x_2, \ldots\}$ (resp. $\mathcal{V}_{\mathbb{I}_\circ} = \{X_1, X_2, \ldots\}$) be a set of variables taking their values over $\mathbb{R}$ (resp. $\mathbb{I}_\circ$). Given $\Sigma_1 = \langle\, \mathbb{R}, \mathcal{F}_1, \mathcal{R}_1 \,\rangle$ a structure, a *real constraint* is defined as a first-order formula built from Σ_1 and $\mathcal{V}_{\mathbb{R}}$. An *interval constraint* is defined in the same way over the structure $\Sigma_2 = \langle\, \mathbb{I}_\circ, \mathcal{F}_2, \mathcal{R}_2 \,\rangle$ and $\mathcal{V}_{\mathbb{I}_\circ}$.

Without loss of generality, we take n as the default arity of a function, a constraint, or a relation, and k an integer belonging to the set $\{1, \ldots, n\}$. Sets are written in uppercase calligraphic letters. The power set of a set $\mathcal{S}$ is written $\mathcal{P}(\mathcal{S})$. Given a real constraint $c(x_1, \ldots, x_n)$ (resp. an interval constraint $C(X_1, \ldots, X_n)$), ρ_c (resp. ρ_C) denotes the underlying relation—that is, the subspace made of "points" verifying the constraint. For the sake of readability, the relation ρ_{c_i} for some constraint c_i is written ρ_i whenever that notation is non-ambiguous. Given an n-ary constraint c, let $\overline{c}$ be $\neg c$, implying that $\rho_{\overline{c}} = \mathbb{R}^n \setminus \rho_c$.

A real relation ρ may be conservatively approximated by the smallest (w.r.t. set inclusion) union of boxes $\mathsf{Union}_\circ(\rho)$ (resp. the smallest box, $\mathsf{Outer}_\circ(\rho)$) containing it. These operators have closed counterparts $\mathsf{Union}_\square(\rho)$ and $\mathsf{Outer}_\square(\rho)$. $\mathsf{Outer}_\circ(\rho)$ is a coarser approximation than $\mathsf{Union}_\circ(\rho)$ but is far more used, it being computationally easier to obtain.

Given a function f defined over reals, an *interval extension* of f is a function F defined over intervals as follows: $\forall I_1, \ldots, \forall I_n \in \mathbb{I}_\circ \colon \mathsf{r}_1 \in I_1, \ldots, \mathsf{r}_n \in I_n \Rightarrow f(\mathsf{r}_1, \ldots, \mathsf{r}_n) \in F(I_1, \ldots, I_n)$. An *interval extension of a real constraint* c is an interval constraint C defined by: $\forall I_1, \ldots, \forall I_n \in \mathbb{I}_\circ \colon \exists a_1 \in I_1 \wedge \cdots \wedge \exists a_n \in I_n \wedge c(a_1, \ldots, a_n) \Rightarrow C(I_1, \ldots, I_n)$.

The *projection* of an interval constraint $C(X_1, \ldots, X_n)$ w.r.t. an index $k \in \{1, \ldots, n\}$ and a box $\mathbf{B} = I_1 \times \cdots \times I_n$, written $C|_{k,\mathbf{B}}$, is defined as the univariate interval constraint obtained by replacing all variables X_j but X_k with the corresponding intervals I_j.

2.2 Local Consistencies

Discarding all inconsistent values from a box of variable domains is intractable when the constraints are real ones (consider, for instance, the constraint $\sin(x) = 1, x \in [0\,..\,2]$). Consequently, weak consistencies have been devised among which one may cite *hull consistency* [4] and *box consistency* [7]. Both consistencies permit narrowing variable domains to (hopefully) smaller domains, preserving the solution set. Since box consistency alone is used as a basis for the algorithms to be introduced in Section 3, it is the only one to be presented thereunder.

Contracting Operators Depending on the considered consistency, one may define different contracting operators for a constraint. In this section, box consistency is first formally presented. Therefrom, an operator based on it is given. The definition for box consistency given below is an instance of the extended one stated by Benhamou *et al.* [5] that slightly differs from the original definition [7] in that it is parameterized by approximation operators.

Definition 1 (Box consistency [5]) *Let c be a real constraint, C an interval extension for c, and $\mathbf{B} = I_1 \times \cdots \times I_n$ a box. The constraint c is said* box-consistent w.r.t. $\mathbf{B}$ *if and only if:*

$$\forall k \in \{1, \ldots, n\}: \quad I_k = \mathsf{Outer}_\circ(I_k \cap \{\mathsf{r} \in \mathbb{R} \mid \mathsf{Outer}_\circ(\{\mathsf{r}\}) \in \rho_{C|_{k,\mathbf{B}}}\})$$

Intuitively, a constraint c is box-consistent w.r.t. a box $\mathbf{B}$ when each projection $I_j, j \in \{1, \ldots, n\}$, of $\mathbf{B}$ is the smallest interval containing all the elements that cannot be distinguished from solutions of $C|_{k,\mathbf{B}}$ due to the inherently limited precision of the computation with floating-point numbers.

Using box consistency to narrow down the variable domains of a constraint leads to the notion of *outer-box contracting operator*:

Definition 2 (Outer-box contracting operator) *Given an n-ary constraint c and a box $\mathbf{B}$, an* outer-box contracting operator $\mathsf{OCb}_c\colon \mathbb{I}_\circ^n \longrightarrow \mathbb{I}_\circ^n$ *for c is defined by:*

$$\mathsf{OCb}_c(\mathbf{B}) = \max\{\mathbf{B}' \mid \mathbf{B}' \subseteq \mathbf{B} \text{ and } c \text{ is box-consistent w.r.t. } k \in \{1, \ldots, n\} \text{ and } \mathbf{B}'\}$$

Proposition 1 (Completeness of OCb) *Given a constraint c, the following relation does hold for any box $\mathbf{B}$: $(\mathbf{B} \cap \rho_c) \subseteq \mathsf{OCb}_c(\mathbf{B})$.*

The consistency and the associated contracting operator considered so far are such that completeness is guaranteed (no solution is lost during the narrowing process). Devising operators ensuring soundness of the results is the topic of the next section.

3 Solving Constraints with Universally Quantified Variables

Constraints arising from the translation of some desired properties for camera control and pathplanning are usually of the form: given I_v, find $x_1, \ldots, x_n$ such that:

$$\forall v \in I_v \colon c(x_1, \ldots, x_n, v) \tag{1}$$

From a practical standpoint, Eq. (1) must be translated into a stronger statement in order to allow picking out values from the variable (interval) domains, *viz.* : given I_v, find intervals $I_1, \ldots, I_n$ such that:

$$\forall x_1 \ldots \forall x_n \forall v \colon v \in I_v \wedge x_1 \in I_1 \wedge \cdots \wedge x_n \in I_n \Rightarrow c(x_1, \ldots, x_n, v)$$

The computed boxes need then be included into the relation ρ_c. More generally, solving constraints with an explicitly universally quantified variable boils down in practice to computing a (subset of) the "inner approximation" of real relations.

Several definitions for the inner-approximation of a real relation exist in the literature, depending on the intended application. Given an n-ary relation ρ, one may single out at least the two following definitions for an inner approximation $\mathsf{Inner}(\rho)$ of ρ:

1. **Def. A.** $\mathsf{Inner}(\rho) = \mathbf{B_1}$, where $\mathbf{B_1} \in \{\mathbf{B} \in \mathbb{I}^n_\circ \mid \mathbf{B} \subseteq \rho\}$ (an inner-approximation is any box included in the relation) [15,3];
2. **Def. B.** $\mathsf{Inner}(\rho) = \mathbf{B_1}$, where $\mathbf{B_1} \in \{\mathbf{B} = I_1 \times \cdots \times I_n \mid \mathbf{B} \subseteq \rho \wedge \forall j \in \{1, \ldots, n\}, \forall I'_j \supseteq I_j \colon \mathbf{B}|_{I_j, I'_j} \subseteq \rho \Rightarrow I'_j = I_j\}$ (an inner-approximation is a box included in the relation that cannot be extended in any direction without containing non-solution points) [23].

In this paper, we consider the following stronger definition for the inner-approximation of a relation :

– **Def. C.** $\mathsf{Inner}(\rho) = \{\mathsf{r} \in \mathbb{R}^n \mid \mathsf{Outer}_\circ(\{\mathsf{r}\}) \subseteq \rho\}$ (the inner-approximation contains all the elements whose enclosing box is included in the relation).

Definitions **A** and **B** imply that disconnected relations are only very partially represented by only one box, a drawback that is avoided when using Def. **C**.

This section first introduces the notion of inner approximation of a relation ρ (that is the approximation by a computable subset of ρ) based on Def. **C**. Contracting operators to compute this approximation are then defined. Therefrom, an algorithm due to Jardillier and Languénou [14] for solving constraint systems with one universally quantified variable is presented. A new approach based on the use of the complete but unsound operators presented in Section 2.2 for the negation of the involved constraints is then described, and compared to the one of Jardillier and Languénou.

Due to lack of space, the reader is referred to the associated research report [6] for the proofs of the propositions to be stated below.

3.1 Computing Inner Sets

From Def. **C** above, an inner-approximation operator may be defined as follows:

Definition 3 (Inner approximation operator) *Given an n-ary relation ρ, an inner-approximation operator* $\mathsf{Inner}_\circ\colon \mathbb{R}^n \to \mathbb{R}^n$ *is defined by:*

$$\mathsf{Inner}_\circ(\rho) = \{\mathsf{r} \in \mathbb{R}^n \mid \mathsf{Outer}_\circ(\{\mathsf{r}\}) \subseteq \rho\}$$

The Inner approximation operator enjoys the following properties:

Proposition 2 (Inner approximation operator properties) *The* Inner *operator is contracting, monotone, idempotent, and distributive w.r.t. the union and intersection of subsets of* $\mathbb{R}^n$.

Inner Contracting Operators The narrowing of variable domains occurring in a constraint is done in the same way as in the outer-approximation case: an *inner-contracting operator* associated to each constraint discards from the initial box all the inconsistent values along with some consistent values. The result is a set of boxes.

Definition 4 (Inner-contracting operator) *Let c be an n-ary constraint. An* inner-contracting operator for c *is a function* $\mathsf{IC}_c\colon \mathbb{I}_\circ^n \to \mathcal{P}(\mathbb{I}_\circ^n)$ *verifying:*

$$\forall \mathbf{B}\colon \mathsf{IC}_c(\mathbf{B}) \subseteq \mathsf{Inner}_\circ(\mathbf{B} \cap \rho_c)$$

Proposition 3 (Soundness of IC) *Given a constraint c and an inner-contracting operator* IC_c *for c, we have:* $\forall \mathbf{B}\colon \mathsf{IC}_c(\mathbf{B}) \subseteq (\mathbf{B} \cap \rho_c)$.

Inner-contracting operators with stronger properties (computation of the greatest representable set included in a relation) may also be defined. These operators are *optimal* in the sense that $\mathsf{IC}_c(\mathbf{B}) = \mathsf{Inner}_\circ(\mathbf{B} \cap \rho_c)$ for any box $\mathbf{B}$.

Devising an inner-contracting operator for a constraint is not as easy as devising an outer-contracting operator since interval techniques only permit to enforce some partial consistencies, that is, discarded values are guaranteed to be non-solutions while no information is known about those that are kept. However, it will be shown in Section 3.2 that outer-contracting operators may be used to obtain inner approximations provided they are applied onto the negation of the considered constraints.

The next section addresses the problem of solving constraint systems where each constraint possesses an occurrence of a universally quantified variable v. The first approach to be described, due to Jardillier and Languénou [14], relies on an evaluation/bisection process to compute an inner-approximation of the underlying relations considering v as a given constant domain.

3.2 Universally Quantified Variables

Given an $(n+1)$-ary constraint $c(x_1, \ldots, x_n, x_v)$ and a box $\mathbf{B} = I_1 \times \cdots \times I_n \times I_v$, applying an inner-contracting operator IC_c to $\mathbf{B}$ outputs a set of boxes $U = \{\mathbf{B'_1}, \ldots, \mathbf{B'_p}\}$ where each $\mathbf{B'_j} = D_1 \times \cdots \times D_n \times D_v$ is a sub-box of $\mathbf{B}$ such that: $\forall \mathsf{r}_1 \in D_1, \ldots, \forall \mathsf{r}_n \in D_n, \forall \mathsf{r}_v \in D_v : c(\mathsf{r}_1, \ldots, \mathsf{r}_n, \mathsf{r}_v)$ does hold.

Therefore, solving a constraint of the form $\forall v \in I_v : c(x_1, \ldots, x_n, v)$ boils down to retaining only boxes $\mathbf{B'} = D_1 \times \cdots \times D_n \times D_v$ of U such that $D_v = I_v$.

In this paper, we address the case of only one explicitly quantified variable. Given a constraint c and a variable v occurring in c, the underlying relation for the constraint $\forall v \in I_v : c$ is written $\tilde{\rho}_{c,v,I_v}$. When the names of the variable and its domain are non-ambiguous, the notation is shortened into $\tilde{\rho}_c$.

The Evaluation Approach In order to tighten a box $\mathbf{B}$ of variable domains for a problem of the form $\forall v \in I_v : c_1 \wedge \cdots \wedge c_m$, Jardillier and Languénou [14], compute an inner approximation of $\rho_1 \cap \cdots \cap \rho_m$ by decomposing the initial domain I_v of v into canonical intervals $I_v^1, \ldots, I_v^p$, and testing with a function of global satisfaction $\mathsf{GlobSat}$ whether $c_1 \wedge \cdots \wedge c_m$ does hold for the boxes $I_1 \times \cdots \times I_n \times I_v^1, \ldots, I_1 \times \cdots \times I_n \times I_v^p$. These evaluations give results in a three-valued logic (*true*, *false*, *unknown*). Boxes labeled *true* contain only solutions, boxes labeled *false* contain no solution at all, and boxes labeled *unknown* are recursively split and re-tested until they may be asserted true or false, or canonicity is reached. Retained boxes are those verifying:

$$\forall j \in \{1, \ldots, p\} : \mathsf{eval}_{\{c_1 \wedge \cdots \wedge c_m\}}(I_1 \times \cdots \times I_n \times I_v^j) = \mathit{true} \tag{2}$$

In this paper, we will call this algorithm $\mathsf{EIA4}$. This process is, in some way, related to the work of Sam-Haroud and Faltings [21] where true, false, or unknown boxes are organized into 2^k-trees to ease the computation of global consistencies.

Equation (2) implies that each retained box is included in the inner-approximation of ρ_c. Consequently, the property verified by each of them is the strong statement: $\forall x_1 \ldots \forall x_n \forall v : v \in I_v \wedge x_1 \in I_1 \wedge \cdots \wedge x_n \in I_n \Rightarrow c_1 \wedge \cdots \wedge c_m$.

Inner Approximation by Negation This section presents algorithms narrowing variable domains and handling one universally quantified variable by reasoning on the negation of the considered constraints. The algorithms described hereafter implement inner-contracting operators for every n-ary constraint c by using $\mathsf{OCb}_{\overline{c}}$. Since values discarded by this operator are guaranteed to be non-solution for $\overline{c}$—by completeness of OCb (see Prop. 1)—, they are guaranteed solutions for c. Formally, a statement of the form $\forall v \in I_v : c(x_1, \ldots, x_n, v)$ is replaced by $\neg\exists v : v \in I_v \wedge \neg c(x_1, \ldots, x_n, v)$ where Statement $\exists v : v \in I_v \wedge \neg c(x_1, \ldots, x_n, v)$ can be handled by the OCb operator. More generally, a constraint system of the form $\forall v \in I_v^1 : c_1(x_1, \ldots, x_n, v) \wedge \cdots \wedge \forall v \in I_v^m : c_m(x_1, \ldots, x_n, v)$ may be translated into the system $[\neg\exists v : v \in I_v^1 \wedge \neg c_1(x_1, \ldots, x_n, v)] \wedge \cdots \wedge [\neg\exists v : v \in I_v^m \wedge \neg c_m(x_1, \ldots, x_n, v)]$ where conjunctions at the highest level have been preserved.

Algorithm ICAb3, based on box consistency, to solve one constraint with a universally quantified variable is first presented below (see Alg. 1). Algorithm ICAb5 that handles several constraints is then described.

Proposition 4 (Correctness of ICAb3) *Let c be a constraint, ρ its underlying relation, v a variable, and $\mathbf{B}$ a box. Then, Alg.* ICAb3 *implements an inner-contracting operator for* $\forall v \in \mathsf{Dom}_{\mathbf{B}}(v)\colon c$.

Remark 1 *One may note that Line* 8 *in Alg. 1 may be replaced by "*$(\mathbf{D_1}, \mathbf{D_2}) \leftarrow \mathsf{Split}_v(\mathbf{D})$*," provided that the initial domain I_v^0 of Variable v is passed as a parameter of* ICAb3*; Line* 6 *would then become: "*$\mathcal{W} \leftarrow \mathbf{B} \setminus \mathbf{D}|_{v,I_v^0}$*."*

Alg. 1. $\mathsf{ICAb3}_c$ – Inner contracting algorithm for $\forall v \in \mathsf{Dom}_{\mathbf{B}}(v)\colon c$

```
1  ICAb3_c(in: B ∈ 𝕀°_n, v ∈ 𝒱_ℝ;  out: 𝒲 ∈ 𝒫(𝕀°_n))
2  begin
3        B' ← OCb_c(B)
4        if (Dom_B'(v) = Dom_B(v)) then
5              D ← OCb_c̄(B')
6              𝒲 ← B' \ D|_{v,B'}
7              if (D ≠ ∅  and  ¬Canonical_v(D)) then
8                    (D_1, D_2) ← Split_v(D|_{v,B'})
9                    𝒲 ← 𝒲 ∪ ICAb3_c(D_1, v) ∪ ICAb3_c(D_2, v)
10             endif
11             return (𝒲)
12       else
13             return (∅)
14 end
```

Function Split_v splits in two intervals one of the non-canonical domains of $\mathbf{D}$. Domain $\mathsf{Dom}_{\mathbf{D}}(v)$ is never considered for splitting. In the same way, $\mathsf{Canonical}_v$ tests canonicity for all domains but the one of variable v.

It is also worthwhile noting that lines 3 *and* 4 *in Alg.* ICAb3 *are only present to speed up computation: box consistency is first tested for the input box* $\mathbf{B}$*; if the domain of the universally quantified variable is narrowed at this stage, it is no longer necessary to continue further since it implies that there is no solution to the constraint* $\forall v \in \mathsf{Dom}_{\mathbf{B}}(v)\colon c$.

Handling constraint systems of the form:

$$(\forall v \in I^1\colon c_1) \wedge \cdots \wedge (\forall v \in I^m\colon c_m)$$

is done by Alg. ICAb5 (Alg. 2) as follows: each constraint of the system is considered in turn together with the sets of elements verifying all the considered constraints theretofore; the point concerning Alg. ICAb5 lies in that *each constraint needs only be invoked once*, since after having been considered for the first time, the elements remaining in the variable domains are all solutions of

the constraint. As a consequence, narrowing some domain later does not require additional work.

Proposition 5 (Soundness of ICAb5) *Let* $\mathcal{S} = \{(c_1, I^1), \ldots, (c_m, I^m)\}$ *be a set of pairs made of a constraint and a domain. Given* $\mathbf{B}$ *a box and* v *a variable, we have:*

$$\mathsf{ICAb5}(\mathcal{S}, \{\mathbf{B}\}, v) \subseteq \mathsf{Inner}_\circ(\mathbf{B} \cap \tilde{\rho}_1 \cap \cdots \cap \tilde{\rho}_m)$$

Comparison of Evaluation *vs.* Negation Approaches

Proposition 6 (EIA4 *vs.* ICAb5) *Let* v *be a variable,* $\mathbf{B}$ *a box,* I_v*, the domain of* v *in* $\mathbf{B}$*, and* $\mathsf{g} = \inf(I_v)$*. Given* $\mathcal{S} = \{(c_1, I_v), \ldots, (c_m, I_v)\}$*, the following property does hold:*

$$\mathsf{EIA4}(\{c_1, \ldots, c_m\}, \mathbf{B}, v, \mathsf{g}) \subseteq \mathsf{ICAb5}(\mathcal{S}, \{\mathbf{B}\}, v)$$

Alg. 2. ICAb5 – Inner contracting algorithm for $\forall v \in I^1 : c_1 \wedge \cdots \wedge \forall v \in I^m : c_m$

```
1  ICAb5(in: S = {(c_1, I^1), ..., (c_m, I^m)}, A ∈ P(I°_n), v ∈ V_R;  out: W ∈ P(I°_n))
2  begin
3      if (S ≠ ∅) then
4          B ← ∅
5          foreach D ∈ A do
7              B ← B ∪ ICAb3_{c_1}(D|_{v,I^1}, v,)
9          endforeach
10         if (B = ∅) then
11             return (∅)
12         else
13             return (ICA4(S \ {(c_1, I^1)}, B, v))
14         endif
15     else
16         return (A)
17     endif
18 end
```

Proposition 6 ensures us that decomposing the domain of the universally quantified variable into canonical intervals does not enhance the precision of the computed inner set.

Let ξ_j be the number of floating-point numbers in Interval I_j, and $\xi = \max_j \xi_j$. For a constraint system composed of m n-ary constraints, the number of calls to GlobSat in EIA4 in the worst case is:

$$\Gamma = m^n \prod_{i=1}^{n} (2\xi_i - 3) = O((m\xi)^n)$$

In the worst case, the number of calls to Alg. OCb in Alg. ICAb5 is also in $O((m\xi)^n)$. However, this evaluation is very pessimistic and does not reflect accurately what happens in practice: as it will be shown in Section 4.3, the filtering induced by Alg. ICAb5 when considering each constraint in turn drastically reduces the number of boxes to consider later, thus speeding up the computation.

Restricting the general framework: correctness In the sequel of this paper, the results presented so far are instantiated for a limited class of constraints, namely inequalities (constraints of the form: $f(x_1, \ldots, x_n) \diamond 0$ with $\diamond \in \{\leqslant, \geqslant\}$. Moreover, only closed intervals are used. Nevertheless, soundness of the algorithms is preserved since computed outer-approximations for the negation of the constraints may only be greater than the one computed on $\mathbb{I}_\circ$. Operator GlobSat used in Alg. EIA4 is implemented by a straight evaluation over intervals of $f(x_1, \ldots, x_n)$ to determine whether it is greater or equal (resp. lower or equal) to zero.

4 Experimental Results

The algorithms presented in Section 3 have been validated in the context of a high-level declarative modeller for camera motion. In this section, the benchmarks used to test the prototype are first described; Alg. ICAb5 is then compared with Alg. EIA4 both for speed and for the ability to provide as soon as possible the user with a representative sample of all solutions.

4.1 Benchmark Description

In the sequel, every benchmark is parameterized by both the number of variables (not counting time t) and the number of constraints to solve.

School Problem$_{3,1}$ [14] is a benchmark corresponding to finding all parabolas lying above a line:

$$\forall t \in [0\,..\,2]\colon at^2 + bt + c \geqslant 2t + 1 \quad \text{with} \quad a \in [0\,..\,1], b \in [0\,..\,1], c \in [0\,..\,1]$$

Benchmark *School Problem*$_{3,2}$ is an inconsistent variant:

$$\forall t \in [0\,..\,2]\colon \begin{cases} at^2 + bt + c \geqslant 2t + 1 \\ at^2 + bt + c \leqslant 2t \end{cases}$$

with same domains for a, b, and c.

Benchmark *Flying Saucer*$_{4,1}$ boils down to finding all pairs of points such that the distance between the flying saucer and the line linking both points is above a given value at any time in a given interval:

$$\sqrt{(x_1 + u(x_2 - x_1) - x_3^t)^2 + (y_1 + u(y_2 - y_1) - y_3^t)^2} \geqslant d$$

with $u = (x_3^t - x_1)(x_2 - x_1) + (y_3^t - y_1)(y_2 - y_1) / \parallel P_2 - P_1 \parallel^2$, where $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ are the unknowns, $P_3^t = (x_3^t, y_3^t)$ the coordinates of the flying

saucer at time t, and d the minimal distance between the flying saucer and the line (P_1, P_2).

Benchmark *Simple Circle* is also a collision problem: given B a point moving along a circling path, find all points A such that the distance between A and B is always greater than a given value. Benchmarks *Simple Circle*$_{2,2}$ and *Simple Circle*$_{2,3}$ are instances of the same problems with respectively 2 and 3 points moving round in circles. For only one circling point, we have:

$$\forall t \in [-\pi \,..\, \pi]: \quad \sqrt{(r_1 \sin t - x)^2 + (r_1 \cos t - y)^2} \geqslant d_1 \qquad \begin{cases} x \in [-5 \,..\, 5] \\ y \in [-5 \,..\, 5] \\ d_1 = 0.5 \end{cases}$$

where d_1 is the minimal required distance between A and B, and $r_1 = 2.5$ is the radius of B's circling path.

Benchmark *Projection*$_{3,4}$ checks whether a moving object projects itself into a frame on the screen for a given time. The static camera has three degrees of freedom: x_t^c, y_t^c, and θ_t^c (horizontal orientation). Given x_t^o, y_t^o, and z_t^o the coordinates of the object's path at time t, and x_t^c, y_t^c, z_t^c, ϕ_t^c, θ_t^c, ψ_t^c, γ_t^c the parameters for the camera, we have:

$$\begin{aligned}
x'_t &= -(x_t^o - x_t^c)\sin\theta_t^c + (y_t^o - y_t^c)\cos\theta_t^c \\
y'_t &= -(x_t^o - x_t^c)\cos\theta_t^c \sin\phi_t^c + (y_t^o - y_t^c)\sin\phi_t^c \sin\theta_t^c + (z_t^o - z_t^c)\cos\phi_t^c \\
z'_t &= -(x_t^o - x_t^c)\cos\theta_t^c \cos\phi_t^c + (y_t^o - y_t^c)\sin\theta_t^c \cos\phi_t^c + (z_t^o - z_t^c)\sin\phi_t^c \\
{}^{\blacktriangledown}x_t^f &\leqslant x'_t/(z'_t/\gamma_t^c) \qquad {}^{\blacktriangle}x_t^f \geqslant x'_t/(z'_t/\gamma_t^c) \\
{}^{\blacktriangledown}y_t^f &\leqslant y'_t/(z'_t/\gamma_t^c) \qquad {}^{\blacktriangle}y_t^f \geqslant y'_t/(z'_t/\gamma_t^c)
\end{aligned}$$

where ${}^{\blacktriangledown}x_t^f$ (resp. ${}^{\blacktriangle}y_t^f$) is the abscissa of the left bound (resp. the ordinate of the right bound) of the frame, $t \in [0 \,..\, 20]$, $x_c \in [-3 \,..\, 3]$, $y_c \in [-3 \,..\, 3]$, $z_c = 2$, $\phi_c \in [-0.5 \,..\, 0.5]$, $\theta_c = 0$, and $\gamma_c^t = 0.8$.

4.2 Improving Computation

Solvers such as Numerica usually isolate solutions with variable domains around 10^{-8} or 10^{-16} in width. By contrast, the applications this paper focuses on are less demanding since the resulting variable domains are used in the context of a display screen, a "low resolution" device. In practice, one can consider that a reasonable threshold ε for the splitting process is some value lower or equal to 10^{-3}.

One of the drawbacks of Alg. EIA4 [14] is that successive output solutions are very similar, while it is of importance to be able to provide the user with a representative sample of solutions as soon as possible.

Tackling this problem using Alg. ICAb5 is done as follows: given a constraint system of the form $\forall v \in I_v : c_1 \wedge \cdots \wedge c_m$ and a Cartesian product of domains $\mathbf{B} = I_1 \times \cdots \times I_n \times I_v$, the solving process has two degrees of freedom, *viz.* the selection of the next constraint to consider, and the selection of the next variable to split. Figure 1 presents the differences with regard to the order of generation of solutions for *Simple Circle*$_{2,2}$ for two strategies concerning the variable splitting

order: *depth-first*, where each constraint is considered in turn, and each domain is split to the threshold splitting limit; and *semi-depth-first* where each constraint is considered in turn, but each variable is split only once and put at the end of the domain queue.

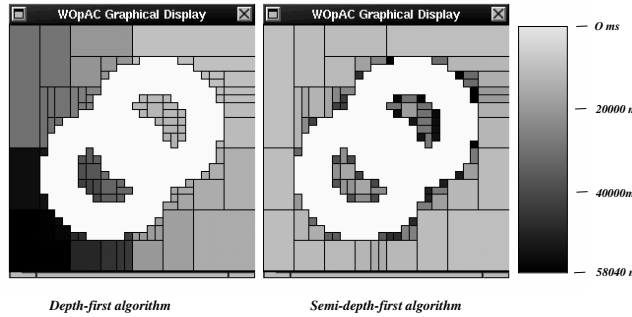

Fig. 1. Depth-first *vs.* semi-depth first

As one may see, the semi-depth-first algorithm computes consecutive solutions spread over all the search space, while the depth-first algorithm computes solutions downwards and from "right" to "left".

4.3 Comparing EIA4 and ICAb5

Algorithms EIA4 and ICAb5 provide different sets of solutions for the same problem. Consequently, a direct comparison of their performances is quite difficult. Moreover, the actual implementation of EIA4 differs from its ideal description since it uses a splitting threshold ω for the domain of the universally quantified variable v instead of checking consistency by eventually reaching canonicity for the samples of the domain I_v.

Tables 1 and 2 compare algorithms EIA4 and ICAb4 from the speed point of view for computing the first solution (*vs.* all solutions). Times are given in seconds on a SUN UltraSparc 166 MHz running Solaris 2.5.

Setting ω and ε to the same value leads to nearly the same solution sets for both algorithms EIA4 and ICAb5.

5 Conclusion

Unlike the methods used to deal with universally quantified variables described in [13], the algorithms presented in this paper are purely numerical ones (except for the negation of constraints). Since they rely on "traditional" techniques used by most of the interval constraint-based solvers, they may benefit from the active researches led to speed up these tools. What is more, they are applicable to

Table 1. EIA4(ω) *vs.* ICAb5 — First solution

Benchmark	EIA4(ω)	ICAb5	EIA4(ω)/ICAb5
$\text{Projection}_{3,8}(\varepsilon = 10^{-1})$	0.2	0.17	1.18
$\text{Projection}_{3,8}(\varepsilon = 10^{-2})$	38.59	0.16	241.19
$\text{Projection}_{3,8}(\varepsilon = 10^{-3})$	> 600	0.16	> 3750.00
$\text{Projection}_{3,4}(\varepsilon = 10^{-2})$	53.12	0.12	442.67
$\text{Projection}_{3,4}(\varepsilon = 10^{-3})$	> 600	0.12	> 5000.00
$\text{School Problem}_{3,1}(\varepsilon = 10^{-2})$	0.02	0.09	0.22
$\text{School Problem}_{3,1}(\varepsilon = 10^{-3})$	1.58	0.09	17.56
$\text{Simple Circle}_{2,2}(\varepsilon = 10^{-2})$	0.99	0.05	19.80
$\text{Simple Circle}_{2,2}(\varepsilon = 10^{-3})$	20.86	0.05	417.20

Table 2. EIA4(ω) *vs.* ICAb5 — All solutions

Benchmark	EIA4	ICAb5	EIA4/ICAb5
$\text{Projection}_{3,5}$	783.03	68.83	11.38
$\text{Projection}_{3,10}$	>9,000	3,634	> 2476.61
$\text{Projection}_{5,5}$	>9,000	3,612	> 2491.69
$\text{School Problem}_{3,1}$	156.02	12.72	12.27
$\text{Flying Saucer}_{4,1}$	1,459.01	1,078.03	1.35
$\text{Simple Circle}_{2,1}$	12,789.03	651.59	19.63
$\text{Simple Circle}_{2,2}$	1,579.05	55.95	28.22

the large range of constraints for which an outer-contracting operator may be devised. By contrast, CAD-based methods deal with polynomial constraints only.

However, constraints to be handled by our algorithm need be easily negated, a requirement trivially met with inequalities but not with equalities. The handling of equalities might be done as described by Sam-Haroud and Faltings [21,20] by relaxing the constraint $f = 0$ into $f = \pm\varepsilon$, thus replacing an equality by two inequalities.

Despite the dramatic improvement of the new method described herein over the one given by Jardillier and Languénou, handling of complex scenes with many objects and a camera allowed to move along all its degrees of freedom in a reasonable time is beyond reach for the moment. Nevertheless, a comforting idea is that most of the traditional camera movements involve but few of the degrees of freedom, thenceforth reducing the number of variables to consider.

Collavizza *et al.* [10] devised a scheme for computing inner-approximations of the relation underlying a real constraint system: starting from a "seed" known to be included in the relation, they expand the domain of the variables as much as possible until obtaining a "maximal" subset of the inner-approximation (with maximality to be understood in the sense of Shary [22]). A drawback of their method lies in that they do not provide any means to compute the seed. An interesting direction for research would be to try using our algorithm to quickly isolate such a seed for each connected subset of the inner-approximation, then resorting to their method to obtain maximal inner-approximations.

Acknowledgements

Discussions with Éric Languénou, Marc Christie, and Laurent Granvilliers that helped improving this paper are gratefully acknowledged. The research exposed here was supported in part by a project of the French/Russian A. M. Liapunov Institute.

References

1. S. Abrams and P. K. Alien. Computing camera viewpoints in an active robot workcell. Technical Report IBM Research Report: RC 20987, IBM Research Division, 1997. 68
2. B. D. O. Anderson, N. K. Bose, and E. I. Jury. Output feedback stabilization and related problems – solution via decision methods. IEEE Trans. on Automatic Control, AC-20(1), 1975. 68
3. J. Armengol, L. Travé-Massuyés, J. Veil, and M. Á. Sainz. Modal interval analysis for error-bounded semiqualitative simulation. In 1r Congrés Català d'lntelligència Artificial, pages 223-231, 1998. 72
4. F. Benhamou. Interval constraint logic programming. In A. Podelski, editor, Constraint programming: basics and trends, volume 910 of LNCS, pages 1-21. Springer-Verlag, 1995. 69, 71
5. F. Benhamou, F. Goualard, L. Granvilliers, and J.-F. Puget. Revising hull and box consistency. InProc. of the 16th Int. Conf. on Logic Programming (ICLP'99), pages 230-244, Las Cruces, USA, 1999. The MIT Press. ISBN 0-262-54104-1. 71
6. F. Benhamou, F. Goualard, É. Languénou, and M. Christie. Universally quantified constraint solving: an application to camera control. Research Report 00.5, Institut de Recherche en Informatique de Nantes, March 2000. Available at http://www.sciences.univ-nantes.fr/irin/Vie/RR/indexGB.html. 72
7. F. Benhamou, D. McAllester, and P. Van Hentenryck. CLP(Intervals) revisited. In Proc. of ILPS'94, pages 124-138. MIT Press, November 1994. 68, 71
8. F. Benhamou and W. J. Older. Applying interval arithmetic to real, integer and boolean constraints. JLP, 32(1): 1-24, 1997. 67, 69
9. J. G. deary. Logical arithmetic. Future Generation Computing Systems, 2(2): 125-149, 1987. 69
10. H. Collavizza, F. Delobel, and M. Rueher. Extending consistent domains of numeric CSP. In Proc. of the 16th IJCAI, volume 1, pages 406-411, July 1999. 80
11. G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In Proc. of the 2nd GI Conf. Automata Theory and Formal Languages, volume 33 of LNCS, pages 134-183, Kaiserslauten, 1975. Springer. 68
12. S. M. Drucker. Intelligent Camera Control for Graphical Environments. PhD thesis, MIT Media Lab, 1994. 68
13. H. Hong. Collision problems by an improved CAD-based quantifier elimination algorithm. Technical Report 91-05, RISC-Linz, 1991. 79
14. F. Jardillier and E. Langienou. Screen-space constraints for camera movements: the virtual cameraman. In N. Ferreira and M. Gobel, editors, Eurographics'98 proceedings, volume 17, pages 175-186. Blackwell Publishers, 1998. 68, 72, 73, 74, 77, 78
15. S. M. Markov. On directed interval arithmetic and its applications. JUCS, 1(7):514-526, 1995. 72

16. K. Meintjes and A. P. Morgan. Chemical equilibrium systems as numerical test problems. ACM TOMS, 16(2):143-151, June 1990. 67
17. R. E. Moore. Interval Analysis. Prentice-Hall, Englewood Clis, N. J., 1966. 67, 68
18. J.-F. Puget. A C $^{++}$ implementation of CLP. In Proc. of SPICIS'94, 1994. 67
19. J.-F. Puget and P. Van Hentenryck. A constraint satisfaction approach to a circuit design problem. J. of Global Optimization, 13:75-93, 1998. 67
20. J. Sam. Constraint Consistency Techniques for Continuous Domains. Phd. thesis, École polytechnique fédérale de Lausanne, 1995. 67, 80
21. J. Sam-Haroud and B. V. Fallings. Consistency techniques for continuous constraints. Constraints, 1:85-118, 1996. 74, 80
22. S. P. Shary. Algebraic solutions to interval linear equations and their applications. In G. Alefeld and J. Herzberger, editors, Numerical Methods and Error Bounds, proc. of the IMACS-GAMM Int. Symposium on Numerical Methods and Error Bounds, pages 224-233. Akademie Verlag, July 1995. 80
23. S. P. Shary. Interval Gauss-Seidel method for generalized solution sets to interval linear systems. In Proc. of MISC'99, pages 51-65, February 1999. 67, 72
24. P. Van Hentenryck, L. Michel, and Y. Deville. Numerica: A Modeling Language for Global Optimization. The MIT Press, 1997. 67
25. A. C. Ward, T. Lozano-Bsrez, and W. P. Seering. Extending the constraint propagation of intervals. InProc. of IJCAI'89, pages 1453-1458, 1989. 67