Combining Local Consistency, Symbolic Rewriting and Interval Methods

Frédéric Benhamou and Laurent Granvilliers

LIFO, Université d'Orléans,
I.I.I.A., Rue Léonard de Vinci
B.P. 6759 45067 ORLEANS Cedex 2 France
{benhamou,granvil}@lifo.univ-orleans.fr

Abstract. This paper is an attempt to address the processing of non-linear numerical constraints over the Reals by combining three different methods: local consistency techniques, symbolic rewriting and interval methods. To formalize this combination, we define a generic two-step constraint processing technique based on an extension of the Constraint Satisfaction Problem, called Extended Constraint Satisfaction Problem (ECSP). The first step is a rewriting step, in which the initial ECSP is symbolically transformed. The second step, called approximation step, is based on a local consistency notion, called weak arc-consistency, defined over ECSPs in terms of fixed point of contractant monotone operators. This notion is shown to generalize previous local consistency concepts defined over finite domains (arc-consistency) or infinite subsets of the Reals (arc B-consistency and interval, hull and box-consistency). A filtering algorithm, derived from AC-3, is given and is shown to be correct, confluent and to terminate. This framework is illustrated by the combination of Gröbner Bases computations and Interval Newton methods. The computation of Gröbner Bases for subsets of the initial set of constraints is used as a rewriting step and operators based on Interval Newton methods are used together with enumeration techniques to achieve weak arc-consistency on the modified ECSP. Experimental results from a prototype are presented, as well as comparisons with other systems.

Keywords: Constraint Satisfaction Problem, local consistency, arc-consistency, filtering algorithms, non-linear constraint solving, Gröbner bases, Newton methods, interval arithmetic, interval constraints.

1 Introduction

The notion of Constraint Satisfaction Problems (CSP) was introduced in the seventies [30, 23, 19] to address combinatorial problems over finite domains and has been heavily studied since then. A Constraint Satisfaction Problem is given by a set of variables, a set of domains associated to these variables and representing their possible values and a set of relations defined over these variables. Solving of CSPs being a NP-hard problem, several approximations of the solution space,

computed by local consistency methods have been proposed, the most famous being arc-consistency [19] and path-consistency [23].

Since the introduction of local consistency in Constraint Logic Programming [29] various extensions have been proposed, among which methods to solve so-called interval constraints [10, 8, 26, 15, 14, 5, 18, 3, 4, 28, 27]. More recently several authors have studied various combinations of solvers in the case of continuous real constraints [20, 22, 13] and, in particular, combinations of techniques from computer algebra (Gröbner bases), Interval Constraint methods and techniques from numerical analysis. Similar ideas for the case of 0/1 linear constraints are suggested in [2] which proposes to compute cutting planes to prune the search space before applying a classical finite domain constraint solver.

We propose in this paper to formalize this combination by defining a generic two-step constraint processing technique based on an extension of the Constraint Satisfaction Problem, called Extended Constraint Satisfaction Problem (ECSP). The first step is a rewriting step, in which the initial ECSP is symbolically transformed. The second step, called approximation step is based on a local consistency notion, called weak arc-consistency, defined over ECSPs in terms of fixed point of contractant monotone operators. This notion is shown to generalize arc-consistency [23, 19], arc B-consistency [18] and interval, hull and box consistency [4]. A filtering algorithm, derived from AC-3 is given and is shown to be correct, confluent and to terminate. This framework is illustrated by the combination of Gröbner Bases computations and Interval Newton methods. In order to maximize the trade-off between pruning and computation time, Gröbner Bases are computed for subsets of the initial set of constraints in the rewriting step. Operators based on Interval Newton methods are used together with enumeration techniques to achieve weak arc-consistency on the modified ECSP. These operators are basically the ones proposed in [4, 28]. Experimental results from a prototype are presented, as well as comparisons with other systems.

With respect to the related papers mentioned above, we believe that this paper generalizes the local consistency notions addressing numerical constraints processing that we are aware of, and proposes a novel combination of the use of Gröbner Bases as a preprocessing step applied to reasonably small subsets of the initial constraint set and of state of the art local consistency techniques based on Interval Newton methods. Finally, preliminary experimental results, detailed at the end of the paper, show significant speed-ups with respect to other recent publications.

The remaining of the paper is organized as follows. Section 2 defines Extended Constraint Satisfaction Problems. Section 3 defines weak arc-consistency and the corresponding filtering algorithm and gives the main theoretical results of the paper. Section 4 introduces a rewriting step used to preprocess the initial CSP and to generate an ECSP preserving the declarative semantics. Section 5 presents an instance of this framework combining Gröbner bases and Interval Newton methods. Section 6 gives experimental results and we conclude in section 7.

2 Extended Constraint Satisfaction Problems

2.1 Definitions

We define an *Extended Constraint Satisfaction Problem* over the Reals[1] as a finite set of constraints (i.e. conjunctions of atomic constraints over the Reals) S over variables $\{x_1, \ldots, x_n\}$ to which are associated contractant monotone operators called *constraint narrowing operators*, defined over subsets of $\mathbb{R}^n$ called *approximate domains* and an n-ary Cartesian product X of elements of $\mathbb{R}$ representing the domains of the variables. Note that to each constraint corresponds a relation over $\mathbb{R}^n$ but that ECSPs are defined over formulae and not relations. This will be used to define symbolic transformations of ECSPs. In the rest of the paper, when no confusion is possible, we will use constraints to denote the relations they represent.

Before giving a more formal definition of ECSPs, we introduce approximate domains and constraint narrowing operators, already discussed in [3] in the framework of interval constraint programming.

Definition 1 *An* approximate domain A *over* $\mathbb{R}$ *is a subset of* $2^{\mathbb{R}}$*, closed under (eventually infinite) intersection, such that* $\mathbb{R} \in$ A*, and for which the inclusion is a well-founded ordering (an approximate domain contains no infinite decreasing sequence of elements).*

Approximate domains, partially ordered by set inclusion, constitute complete lattices in which the meet operation is defined by set intersection. The join of sets is defined as the smallest set larger than all of them, i.e. the approximation of their union. (For a discussion on a lattice theoretic view of interval constraints, see [26]).

Given an approximate domain A, and an n-ary relation ρ over $\mathbb{R}$, a *narrowing operator* for the relation ρ is a correct, contractant, monotone, idempotent function $N : A^n \longrightarrow A^n$. More formally,

Definition 2 *Let A be an approximate domain. Let ρ be an n-ary relation over $\mathbb{R}$. The function $N : A^n \longrightarrow A^n$ is a* constraint narrowing operator *for the relation ρ iff for every $u, v \in A^n$, the four following properties hold:*

$$(1) \qquad N(u) \subset u, \qquad \textit{(Contractance)}$$

$$(2) \qquad u \cap \rho \subset N(u), \qquad \textit{(Correctness)}$$

$$(3)\ u \subset v \textit{ implies } N(u) \subset N(v), \qquad \textit{(Monotonicity)}$$

$$(4) \qquad N(N(u)) = N(u). \qquad \textit{(Idempotence)}$$

We can now introduce formally ECSPs.

[1] This definition can be generalized to any set but it is not the topic of this paper.

Definition 3 *Let A be an approximate domain over $\mathbb{R}$.*
An Extended Constraint Satisfaction Problem *is a pair (S, X), where $S = \{(C_1, N_1), (C_2, N_2), \ldots, (C_m, N_m)\}$ is a set of pairs made of a real constraint C_i and of a constraint narrowing operator N_i for C_i, and X is an n-ary Cartesian product of elements of A.*

Declarative and approximate semantics of ECSPs

First, note that classical CSPs, assuming that the relations are given intensionally, can be represented as pairs (S, X) where S is a set of constraints and X the Cartesian product of the domains associated to the variables appearing in S. It follows that to every ECSP $E = (\{(C_1, N_1), (C_2, N_2), \ldots, (C_m, N_m)\}, X)$ corresponds a CSP $E' = (\{C_1, C_2, \ldots, C_m\}, X)$.

The declarative semantics of the ECSP E, denoted E^*, is then defined as the declarative semantics of E', that is the set of n-ary tuples $(x_1, \ldots, x_n) \in X$ satisfying the conjunction of the constraints $C_1, C_2, \ldots, C_m$.

Definition 4 (Declarative semantics of ECSPs)
Let $E = (\{(C_1, N_1), (C_2, N_2), \ldots, (C_m, N_m)\}, X)$ be an ECSP. The declarative semantics *of E is defined as*

$$E^* = \bigcap_{i=1}^{m} C_i \cap X$$

Unfortunately this set is generally either computationaly intractable or even uncomputable. This leads to define what we call the approximate semantics of E, denoted $\overline{E}$, as being the greatest common fixed point of the N_i's included in X.

Definition 5 (Approximate semantics of ECSPs)
Let $E = (\{(C_1, N_1), (C_2, N_2), \ldots, (C_m, N_m)\}, X)$ be an ECSP. If $fp(N_i)$ denotes the set of fixed points of N_i, then the approximate semantics *of E is defined as*

$$\overline{E} = max(\{u \in \bigcap_{i=1}^{m} fp(N_i) \mid u \subseteq X\})$$

The proof of existence of $\overline{E}$ is constructive and based on properties of the filtering algorithm presented in section 3.1.

Due to the correctness property of narrowing operators with respect to their associated constraints, we have the following, expected, result concerning the two semantics:

Property 1 *Let E be an ECSP. Then, $E^* \subseteq \overline{E}$.*

We describe in the next section the local consistency notions related to the approximate semantics of ECSPs.

3 Weak arc-consistency

As for CSPs, ECSPs are processed by local consistency techniques, that is propagation. The local consistency used in ECSPs is an extension of arc-consistency, called here *weak arc-consistency*.

We recall here first the definition of arc-consistency for Constraint Satisfaction Problems. Intuitively, a CSP is said to be arc-consistent iff for every variable appearing in a constraint C, there is no value of its associated domain which cannot be "completed" with values from the domains of the other variables in order to satisfy C. More formally:

Definition 6 (arc-consistency)
Let $E = (\{C_1, \ldots, C_m\}, X)$ be a CSP. Then, E is arc-consistent iff $\forall i \in \{1, \ldots, m\}, \forall j \in \{1, \ldots, n\}, X_i \subset \pi_i(C_j \cap X)$, where, for every n-ary relation ρ, $\pi_i(\rho)$ denotes the ith projection of ρ.

Moreover a CSP $E = (S, X)$ is said to be *maximally arc-consistent* with respect to a Cartesian product of domains X' iff X is the greatest Cartesian product (with respect to inclusion) such that E is arc-consistent and $X \subseteq X'$.

These definitions are generally used, in a slightly different formalism, for finite sets and binary constraints, but can be extended, at least theoretically, to infinite sets and n-ary constraints. We give below the definition of weak arc-consistency for ECSPs.

Definition 7 (weak arc-consistency)
Let $E = (\{(C_1, N_1), (C_2, N_2), \ldots, (C_m, N_m)\}, X)$ be an ECSP. E is weakly arc-consistent *iff $X = \overline{E}$.*

The definition for *maximal weak arc-consistency* is expressed as follows:

Definition 8 (maximal weak arc-consistency)
Let $E = (S, X)$ be an ECSP. Let X' be a Cartesian product of domains such that $X \subseteq X'$. Then, if E' is the ECSP (S, X'), E is maximally weak arc-consistent *with respect to X' iff E is weakly arc-consistent and $X = \overline{E'}$.*

We then state without proof, due to size limitations, the following properties: Let $E = (\{(C_1, N_1), \ldots, (C_m, N_m)\}, X)$ be an ECSP over an approximate domain A. Let $E' = (\{C_1, \ldots, C_m\}, X)$ be the CSP associated to E.

1. If for all i in $\{1, \ldots, m\}$, and for all u in A^n, $N_i(u)$ is the Cartesian product of the n projections of the relation $u \cap C_i$, then E is weakly arc-consistent iff E' is arc-consistent.
2. If A is made of all intervals whose bounds are floating point numbers, and if for all i in $\{1, \ldots, m\}$, and for all u in A^n, $N(u) = \underline{\text{apx}}(\rho \cap u)$, where $\underline{\text{apx}}$ is the function that associates to any subset ρ of $\mathbb{R}^n$ the intersection of all the elements of A containing ρ, then E is weakly arc-consistent iff E' is hull-consistent (as defined in [4]) and iff it is arc B-consistent (as defined in [18]).

3. if A is made of all finite unions of intervals whose bounds are floating point numbers and the N_i's are defined as above, then E is weakly arc-consistent iff E' is interval-consistent (as defined in [4]) .
4. weak arc-consistency generalizes also box-consistency [4] and the same type of result as above can be shown as will be developped in section 5.2.

3.1 Filtering algorithm

The algorithm used to compute, from an ECSP E and a Cartesian product of domains X, an equivalent ECSP maximally weak arc-consistent with respect to X is essentially an immediate adaptation of AC-3 [19] and of the filtering algorithms used in interval constraint-based systems like `BNR-Prolog` [26], `CLP(BNR)` [5] or `Newton` [4, 28].

`filtering(` **in** $\{(C_1, N_1), \ldots, (C_m, N_m)\}$; **inout** $X = X_1 \times \ldots \times X_n$ `)`
begin
  `S :=` $\{C_1, \ldots, C_m\}$;
  while $S \neq \emptyset$ **and** $X \neq \emptyset$ **do**
    C `:= choose one` C_i `in` S;
    X' `:=` $N_i(X)$;
    if $X' \neq X$ **then**
      S `:=` $S \cup \{C_j \mid \exists v_k \in var(C_j) \wedge X'_k \neq X_k\}$;
      X `:=` X'
    endif
    S `:=` $S \setminus \{C\}$
  endwhile
end;

Again, we state without proof the main properties of the algorithm:

1. if the computation of the constraint narrowing operators terminates, the algorithm terminates due to the fact that set inclusion is a well-founded ordering for approximate domains.
2. the algorithm is correct in the sense that the solution space defined by the declarative semantics of the initial ECSP is *included* in the computed Cartesian product. In particular, if this Cartesian product is empty, the ECSP is unsatisfiable.
3. the algorithm is confluent in the sense that the output is independant of the order in which the constraints are chosen.
4. given as input E, the output is $\overline{E}$.

4 Symbolic rewriting

Given a CSP over the Reals, it is often very efficient to preprocess it with symbolic rewriting techniques before applying the filtering algorithm, to generate

redundant and/or surrogate constraints. Therefore, we introduce a preprocessing step called *constraint rewriting*.

Constraint rewriting consists in two phases:

1. repeated syntactical transformations of the constraints of a CSP preserving its declarative semantics,
2. association of constraint narrowing operators to the generated set of constraints.

Note that the first phase alone can solve the CSP when the rewriting techniques are complete and computationaly tractable w.r.t. the problem at hand (e.g. cylindrical algebraic decomposition for polynomials, syntactic processing of 0/1 constraints, etc.). Another remark is that, in order to compute a usable ECSP, to each generated narrowing operators must correspond a terminating algorithm for the considered constraint.

The second part of this paper is devoted to an instance of this generic framework addressing the approximation of solutions of multivariate polynomials over the Reals.

5 Combining Gröbner bases and interval Newton methods

We describe in this section an instance of the framework introduced in the previous sections whose purpose is to approximate the solutions of systems of nonlinear polynomial equations. Different programming languages and systems have been designed to handle similar problems such as `Newton` [4, 28] based on interval Newton methods and local consistency techniques, `CLP(BNR)` [5] implementing interval propagation methods, `CoSAc` [22] which combines `Maple`, a Gröbner bases module and a linear solver, the system presented in [20] based on Gröbner bases, the simplex algorithm and interval propagation methods, `TKIB` [14] introducing an approximation notion called tightening and the Krawckzyk operator [16, 17].

In this paper, we propose to combine Gröbner bases computations over subsets of the initial system, used as a preprocessing step, interval Newton methods to compute weak arc-consistency and enumeration techniques used to separate the solutions. This combination was motivated by three main arguments. First, syntactical transformations of the system by means of Gröbner bases computations may speed up the resolution step by adding redundant constraints used to reduce the search space. Second, computing Gröbner bases over subsets of the initial system still permits to generate redundant constraints while keeping the algorithm memory and time consumption in reasonable bounds. Finally, propagation based on interval Newton methods has been preferred to classical interval propagation algorithms for their superior efficiency.

5.1 Constraint rewriting

As mentioned above, the constraint rewriting step of our system consists in computing Gröbner bases for selected subsets of the initial system. We recall

first very briefly some basics on Gröbner bases.

Gröbner bases

The concept of Gröbner bases for polynomial ideals is due to Buchberger [6]. The main idea is to transform a multivariate polynomial set to obtain a new set in a certain normal form which makes possible the resolution of many problems in polynomial ideal theory and in particular the problem of solving polynomial equations (see [7, 9] for more details).

The computation of Gröbner bases essentially depends on the following three notions:

1. Every polynomial is ordered using a *total ordering on monomials* $\succ$ which is Noetherian and compatible with the multiplication on monomials. The maximal monomial of a polynomial p wrt $\succ$ is called the *leading term* of p.
2. The *reduction process* of a polynomial f wrt a set of polynomials F computes an irreducible polynomial obtained as a remainder after iteratively dividing f by the elements of F.
3. The *S-polynomial* of two polynomials f and g, denoted $S(f, g)$, is a combination of two polynomials that cancels their leading terms.

We define Gröbner bases by giving the algorithmic characterization proposed by Buchberger in [7]:

Definition 9 (Gröbner basis)
Let $G = \{f_1, \ldots, f_n\}$ be a set of polynomials. G is a Gröbner basis iff for all pairs $(i, j) \in \{1, \ldots, n\}$ the reduction wrt G of the S-polynomial $S(f_i, f_j)$ is equal to 0.

The basic algorithm computing Gröbner bases naturally follows from the definition. Let F be a set of polynomials. Let F' be a set of polynomials initialized with F. The algorithm consists in successive computations of reduced S-polynomials over F'. These S-polynomials are added to F' at each step. The algorithm stops, after a finite number of steps, when no S-polynomial different from 0 can be computed. The resulting set of polynomials is a Gröbner basis and is denoted GB(F). A Gröbner basis can be further simplified into a *reduced Gröbner basis* (see details in [7]).

Another property which is of interest for this paper is that, given a system of polynomial equations $S = \{f_1 = 0, \ldots, f_n = 0\}$ and $G = \{g_1, \ldots, g_m\}$ a Gröbner basis such that $G = \mathrm{GB}(\{f_1, \ldots, f_n\})$, the systems $\{g_1 = 0, \ldots, g_m = 0\}$ and S have the same solutions.

Rewriting step

The Gröbner bases are used as a preprocessing step to generate redundant equations which may prune the search space. In what follows we choose the total

degree ordering for monomials and the usual lexicographic ordering for variables.

Let $S = \{f_1 = 0, \ldots, f_n = 0\}$ be a constraint system. Let $\{F_0, F_1, \ldots, F_m\}$ be a partition of the set $\{f_1, \ldots, f_n\}$. The rewriting step consists in the computation of reduced Gröbner bases G_i such that for all $i \in \{1, \ldots, m\}$ $G_i = \mathrm{GB}(F_i)$. The set F_0 contains the constraints that are not rewritten. The correctness of this computation is guaranteed by Gröbner bases properties (see the previous section). The termination comes from the termination of Gröbner bases algorithms. The order in which Gröbner bases are computed do not change the resulting system due to the independence of the sets F_i.

In some cases, inconsistency can be detected at this stage. By application of the Weak Nullstellensatz theorem, if there exists $i \in \{1, \ldots, p\}$ such that the constant polynomial 1 belongs to G_i then G_i is inconsistent and so is the whole system.

Consider the set $\{f_1 = x^2 + y^2 - 1, f_2 = x^2 - y\}$. The reduction of the S-polynomial $S(f_1, f_2)$ gives the univariate polynomial $f_3 = y^2 + y - 1$ whose roots can be easily extracted. The elimination of x is possible because the leading terms of f_1 and f_2 are equals. This suggests a heuristic to compute the partition in such a way that its elements contain polynomials whose leading terms share variables. In the current implementation the partition is computed by hand however we believe that this process could be automated as suggested by the previous example.

Example

To illustrate this constraint rewriting step, we present an example taken from [14].

$$S = \begin{cases} \bullet\ x_1^2 + x_2^2 + x_3^2 + x_4^2 - 2 * x_1 - 3 = 0 \\ \bullet\ x_1^2 + x_2^2 + x_3^2 + x_4^2 - 4 = 0 \\ \circ\ x_1 + x_2 + x_3 + x_4 - 1 = 0 \\ \circ\ x_1 + x_2 - x_3 + x_4 - 3 = 0 \end{cases} \qquad S' = \begin{cases} x_2^2 + x_3^2 + x_4^2 - 3.75 = 0 \\ x_1 + x_2 - x_3 + x_4 - 3 = 0 \\ x_3 + 1 = 0 \\ x_1 - 0.5 = 0 \end{cases}$$

The system S is divided in two subsystems S_1 and S_2, materialized by two different kinds of bullets. The resulting set S' is the union of $\mathrm{GB}(S_1)$ and $\mathrm{GB}(S_2)$. In S' the domains of variables x_1 and x_3, which are reduced to a singleton, can be computed immediately. Moreover we remark that three polynomials from S were removed during the computation of reduced Gröbner bases for each subsystems.

5.2 Constraint solving

The approximate resolution of the system after the rewriting step is basically the one proposed in [28] which corresponds to state of the art techniques in constraint solving. We show here how this method can be expressed in terms of computing weak arc-consistency over a certain ECSP.

Basics

We recall first some basic notions from [4, 28]. Due to space restrictions we do not recall the basics of interval computations which can be found for example in [24, 1]. We overload function symbols over the reals with function symbols over the intervals when no confusion is possible. Let D be a set of floating point intervals.

Given a real expression f, the *natural interval extension* F of f is the interval expression which is a copy of f where operation symbols are replaced by interval operations, variables by interval variables and constants by interval constants. Interval constraints are of the form $F = 0$ where F is an expression over intervals. The semantics of interval functions and constraints is quite natural and follows the definitions given in [4]. An interval I is said to be *canonical* w.r.t D iff $\forall J \in D, J \subseteq I$ implies $I = J$.

Let $C = (F = 0)$ be a constraint where F is an interval expression on variables $X_1, \ldots, X_n$ over D and let $I_1 \times \ldots \times I_n$ be a Cartesian product of elements of D. The ith *projection* of C is the constraint $F' = 0$ where F' is obtained from F by replacing the variables X_j by the intervals I_j for all $j \in \{1, \ldots, i-1, i+1, \ldots, n\}$ and is denoted $\pi_i(C)$. We also define the *constraint enclosure* of C, denoted $\mathrm{EN}(C)$, as being the smallest subset of D^n such that, for every n-ary tuple A made of canonical intervals, if A is a solution of C then $A \in \mathrm{EN}(C)$.

Interval Newton methods

Newton methods are numerical algorithms approximating zeros of functions and derived from the mean value theorem. We present here two methods, one based on the Newton interval function and other based on the Taylor series approximation.

Let f be a real function continuously differentiable between x and y and consider that y is a zero of f. It can be deduced from the mean value theorem that $y = x - f(x)/f'(a)$. The Newton method iterates this formula to approximate roots of f. This method has been extended to interval functions [24, 16, 12, 1, 11, 17, 25]. Let X be an interval containing x and y and suppose that F' is the natural interval extension of f', F the natural interval extension of f and $m(X)$ the approximation of the center of X. The Newton interval function is the function

$$N(X) = m(X) - F(m(X))/F^{'}(X)$$

From this definition one can design an *interval Newton method* enclosing roots of interval functions. Given an initial interval X_0 and an interval function F, a sequence of intervals $X_1, X_2, \ldots, X_n$ is computed using the iteration step $X_{i+1} = N(X_i) \cap X_i$. X_n is either empty, which means that X_0 contains no zero of F, or is a fixed point of N.

The centered form is based on the Taylor series approximation. Let $C = (F = 0)$ be an interval constraint, X be a vector of variables, $I = (I_1, \ldots, I_n)$ be a vector of intervals and J be the vector of the continuous partial derivatives

of F. We denote by $M(I)$ the midpoint vector $(m(I_1), \ldots, m(I_n))$. The centered form is the interval expression $F(X) = F(M(I)) + J(I)(X - M(I))$. If X is a zero of F it follows

$$F(M(I)) + J(I)(X - M(I)) = 0$$

In what follows this last constraint will be denoted TAY(C). Let C_i be the ith projection of this constraint. One can remark that C_i is an unary constraint over X_i. After a basic transformation of C_i, X_i can be expressed as a constant interval expression $G(I)$. The method based on the Taylor series approximation computes $X_i = I_i \cap G(I)$.

Generation and resolution of ECSPs

The main idea is to consider, for each non-linear constraint over p variables, $2p$ "copies" of the constraint. To each copy is associated a constraint narrowing operator defined over closed floating point intervals and implementing one of the two methods described above.

More precisely, let $E = (\{C_1, \ldots, C_m)\}, X)$ be a CSP over $\mathbb{R}$. The approximate domain D is the set of all closed intervals whose bounds are floating point intervals.

Then, an ECSP $E' = (S \cup S', X)$ is computed from E by the following algorithm:

```
generation( in E = (S, X) ; out E' = (S' ∪ S'', X) )
begin
    S' := ∅; S'' := ∅;
    while S ≠ ∅ do
        C := choose one constraint in S;
        P := arity(C);
        for i := 1 to P do
            S' := S' ∪ {(C, N_i')};
            S'' := S'' ∪ {(C, N_i'')};
        endfor
        S := S \ {C}
    endwhile
end;
```

First let us remark that, in E', to every constraint narrowing operator N corresponds a variable from the initial set $\{X_1, \ldots, X_n\}$. The index of this variable is denoted ind(N). Then, for every $(C', N') \in S'$, for every $(C'', N'') \in S''$ and for every n-ary Cartesian product u of elements of D, N' and N'' are defined as follows:

$$N'(u) = \mathrm{EN}(\pi_{\mathrm{ind}(N')}(C))$$
$$N''(u) = \mathrm{EN}(\pi_{\mathrm{ind}(N'')}(\mathrm{TAY}(C)))$$

Finally, given such an ECSP E', the (possible) solutions are computed using a branch-and-prune algorithm [28] which is basically an iteration of two steps:

1. a pruning step which computes weak arc-consistency for E' using the filtering algorithm presented in section 3.1. In this case weak arc-consistency is equivalent to box-consistency.
2. a branching step which generates two subproblems by splitting one non-canonical interval, when possible.

The following example (continuing the example developed in the previous section) describes the constraint resolution process. It will give the underlying intuition of the generic branch-and-prune algorithm. Consider that the initial Cartesian product of domains is $[-10, 10]^4$ and the required precision for the resulting intervals is 10^{-12}. Weak arc-consistency is computed, providing the following intervals:

$$\begin{aligned} x_1 \in X_1 &= [+0.5000000000000, +0.5000000000000] \\ x_2 \in X_2 &= [-0.1583186821562, +1.6583186821562] \\ x_3 \in X_3 &= [-1.0000000000000, -1.0000000000000] \\ x_4 \in X_4 &= [-0.1583186821562, +1.6583186821562] \end{aligned}$$

No more pruning can be done and the intervals X_2 and X_4 are not canonicals. A branching step is applied and generates two subintervals (thus two Cartesian products) from X_4 which is split. The pruning step on the first Cartesian product gives immediately the first solution:

$$\begin{aligned} x_1 &\in [+0.5000000000000, +0.5000000000000] \\ x_2 &\in [+1.6513878188659, +1.6513878188660] \\ x_3 &\in [-1.0000000000000, -1.0000000000000] \\ x_4 &\in [-0.1513878188660, -0.1513878188659] \end{aligned}$$

Then the second solution is derived from the second Cartesian product after backtracking:

$$\begin{aligned} x_1 &\in [+0.5000000000000, +0.5000000000000] \\ x_2 &\in [-0.1513878188660, -0.1513878188659] \\ x_3 &\in [-1.0000000000000, -1.0000000000000] \\ x_4 &\in [+1.6513878188659, +1.6513878188660] \end{aligned}$$

6 Experimental results

We have implemented a prototype of a polynomial constraint solver, which we call here `INGB`, integrating the interval Newton method called `IN` described in the previous sections and a Gröbner bases module called `GB`. In order to compare constraint solvers we propose two parameters. The computation time is the most used computational parameter and permits to compare behaviours of rather different systems. This parameter is used here to compare our system with `CoSAc` [22] and `Newton` [28]. The number of branchings is the second parameter and illustrates the use of Gröbner bases as a preprocessing step. In particular, `INGB` is compared to `TKIB` [14].

The following computational results are given for a SUN Sparc20 and the width of the resulting intervals are smaller than 10^{-12}. The results of `TKIB` were computed on a Silicon Graphics workstation with a 50 MHz processor MIPS 4000/4010. The results of `CoSAc` and `Newton` were computed on a SUN Sparc10. The Gröbner bases are computed using different orderings for monomials and the usual lexicographic ordering for the variables. Derivatives are computed symbolically.

Example 1 *This example comes from [14] and concerns the intersection of a circle and a parabola. We compare* `INGB` *with the solver* `TKIB` *which implements an approximation method called tightening, the Krawckzyk operator and enumeration methods. A reduced Gröbner basis S' is computed for the whole system S.*

$$S = \begin{cases} x^2 + y^2 - 1 = 0 \\ x^2 - y = 0 \end{cases} \qquad S' = \begin{cases} x^2 - y = 0 \\ y^2 + y - 1 = 0 \end{cases}$$

The initial intervals for x, y are $[-1.5, 1.5]$. The experimental results show that `INGB` *has the best possible behaviour on the number of enumeration steps (one branching for two solutions).*

Example 2 *This example was developed in the previous section and is also taken from [14]. The initial intervals for the variables are $[-10, 10]$. The table 1 shows again an improvement w.r.t. the number of branchings and a good runtime behaviour.*

Example 3 *This example is taken from [21, 22] and consists in moving a rectangle through a right angled corridor. The system S is divided in three parts materialized by different bullets for the last two subsystems.* `GB` *is applied on each of the last two parts.*

$$S = \begin{cases} \quad y - b = 0 \\ \quad b - r * t = 0 \\ \quad w^2 - 1 + t^2 = 0 \\ \bullet\ x - l * t^3 - L * w = 0 \\ \bullet\ y - L * t - l * w^3 = 0 \\ \bullet\ L - 1 = 0 \\ \bullet\ l - 2 = 0 \\ \circ\ x - a = 0 \\ \circ\ 2 * a - 3 = 0 \end{cases} \qquad S' = \begin{cases} t^2 + w^2 - 1 = 0 \\ -r * t + b = 0 \\ L - 1 = 0 \\ l - 2 = 0 \\ a - x = 0 \\ -b + y = 0 \\ t^3 + 0.5 * w - 0.5 * x = 0 \\ w^3 + 0.5 * t - 0.5 * y = 0 \\ x - 1.5 = 0 \end{cases}$$

The initial intervals are $[-10^6, 10^6]$ for x, y, l, t, L, b, r, a and $[0, 10^6]$ for w. The experimental results show that `INGB` *has a better behaviour than* `CoSAc` *which is due for the most part to interval Newton methods and to the partial use of Gröbner bases in our system.*

Example 4 *This example comes from [28] and describes an economics modelling problem for which we consider the case of four variables. A reduced Gröbner basis S' is computed for the whole system S.*

$$S = \begin{cases} \bullet\ x_1 * x_4 + x_1 * x_2 * x_4 - 0.35 = 0 \\ \bullet\ x_2 * x_4 + x_1 * x_3 * x_4 - 1.086 = 0 \\ \bullet\ x_3 * x_4 - 2.05 = 0 \\ \bullet\ x_1 + x_2 + x_3 + 1 = 0 \end{cases}$$

$$S' = \begin{cases} x_1 + x_2 + x_3 + 1 = 0 \\ x_2 + \frac{850}{2593} * x_3 + \frac{500}{2593} * x_4 + \frac{3311}{2593} = 0 \\ x_3 + \frac{10000}{71463} * x_4^2 + \frac{15240}{23821} * x_4 + \frac{514433}{595525} = 0 \\ x_4^3 + \frac{1143}{250} * x_4^2 + \frac{1543299}{250000} * x_4 + \frac{2929983}{200000} = 0 \end{cases}$$

The initial intervals are $[-10^8, 10^8]$ for each variables. The experimental results in the table 1 show that the computation of Gröbner bases speeds up interval Newton methods. The reason is that the roots of the last polynomial can be computed immediately using Newton methods. Then the propagation in S' of the values of x_4 allows to solve the first three equations using the same process.

Benchs	Methods	CPU time (in seconds)			Branching(s)	Solution(s)
		GB	IN	Total		
Ex1	INGB	0.01	0.02	0.03	1	2
	TKIB	-	-	0.03	3	2
Ex2	INGB	0.02	0.11	0.13	1	2
	TKIB	-	-	0.78	10	2
Ex3	INGB	0.05	0.19	0.24	0	1
	CoSAc	2	-	2	-	1
Ex4	INGB	0.13	0.07	0.20	0	1
	Newton	-	0.60	0.60	?	1

Table 1. Experimental results.

7 Conclusion

In this paper, we have proposed an extension of the notion of Constraint Satisfaction Problems, called Extended Constraint Satisfaction Problem, to formalize

the collaboration of different solvers over continuous domains. We have defined an extended notion of arc-consistency, called weak arc-consistency and shown that it generalizes previous local consistency notions for finite and continuous CSPs. Weak arc-consistency is characterized by a fixed point semantics over the lattice of approximate domains. To illustrate this framework we have proposed a novel combination of the use of Gröbner Bases as a preprocessing step applied to reasonably small subsets of the initial constraint set and of state of the art local consistency techniques based on Interval Newton methods. Finally, we have reported experimental results and compared with other recent implementations. We intend to continue these experiments and notably to automatize the process which partition the initial system in the preprocessing step. We envisage also to instantiate this framework to other combinations of symbolic and numeric methods.

Acknowledgements

We are grateful to Alain Colmerauer, Gérard Ferrand and Pascal Van Hentenryck for fruitful discussions on these topics.

References

1. G. Alefeld and J. Herzberger. *Introduction to Interval Computations.* Academic Press, 1983.
2. P. Barth and A. Bockmayr. Finite domain and cutting plane techniques in CLP($\mathcal{PB}$). In L. Sterling, editor, *Proceedings of ICLP'95*, pages 133–147, Kanagawa, Japan, 1995. MIT Press.
3. 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.
4. F. Benhamou, D. McAllester, and P. Van Hentenryck. CLP(Intervals) Revisited. In *Proceedings of ILPS'94*, pages 124–138, Ithaca, NY, USA, 1994.
5. F. Benhamou and W. J. Older. Applying Interval Arithmetic to Real, Integer and Boolean Constraints. *Journal of Logic Programming*, 1996. forthcoming.
6. B. Buchberger. *An Algorithm for Finding a Basis for the Residue Class Ring of a Zero-Dimensional Polynomial Ideal.* PhD thesis, University of Innsbruck, 1965. (in German).
7. B. Buchberger. Gröbner Bases: an Algorithmic Method in Polynomial Ideal Theory. In *Multidimensional Systems Theory*, pages 184–232. D. Reidel Publishing Company, 1985.
8. J. G. Cleary. Logical Arithmetic. *Future Computing Systems*, 2(2):125–149, 1987.
9. D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties and Algorithms.* Springer-Verlag, 1992.
10. E. Davis. Constraint Propagation with Interval Labels. *Artificial Intelligence*, (32), 1987.
11. E. R. Hansen and R. I. Greenberg. An Interval Newton Method. *Applied Mathematics and Computation*, 12:89–98, 1983.

12. E. R. Hansen and S. Sengupta. Bounding Solutions of Systems of Equations using Interval Analysis. *BIT*, 21:203–211, 1981.
13. H. Hong. Confluency of Cooperative Constraint Solving. Technical Report 94-08, RISC-Linz, Johannes Kepler University, Linz, Austria, 1994.
14. H. Hong and V. Stahl. Safe Start Region by Fixed points and Tightening. *Computing*, 53(3-4):323–335, 1994.
15. E. Hyvönen. Constraint Reasoning based on Interval Arithmetic. The Tolerance Propagation Approach. *Artificial Intelligence*, 58:71–112, 1992.
16. R. Krawczyk. Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. *Computing*, 4:187–201, 1969.
17. R. Krawczyk. A Class of Interval Newton Operators. *Computing*, 37:179–183, 1986.
18. O. Lhomme. Consistency techniques for numeric CSPs. In R. Bajcsy, editor, *Proceedings of the 13th IJCAI*, pages 232–238, Chambéry, France, 1993. IEEE Computer Society Press.
19. A. Mackworth. Consistency in Networks of Relations. *Artificial Intelligence*, 8(1):99–118, 1977.
20. P. Marti and M. Rueher. A Distributed Cooperating Constraint Solving System. *International Journal on Artificial Intelligence Tools*, 4(1):93–113, 1995.
21. E. Monfroy. Gröbner Bases: Strategies and Applications. In *Proceedings of AISMC'92*, volume 737 of *LNCS*, pages 133–151. Springer-Verlag, 1992.
22. E. Monfroy, M. Rusinowitch, and R. Schott. Implementing Non-Linear Constraints with Cooperative Solvers. In K. George, J. Carroll, D. Oppenheim, and J. Hightower, editors, *Proceedings of ACM Symposium on Applied Computing*, pages 63–72, February 1996.
23. U. Montanari. Networks of Constraints: Fundamental Properties and Applications to Picture Processing. *Information Science*, 7(2):95–132, 1974.
24. R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1966.
25. A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge University Press, 1990.
26. W. Older and A. Vellino. Constraint Arithmetic on Real Intervals. In F. Benhamou and A. Colmerauer, editors, *Constraint Logic Programming: Selected Research*. MIT Press, 1993.
27. I. Shvetzov, A. Semenov, and V. Telerman. Constraint Programming based on Subdefinite Models and its Applications. In *ILPS'95 post-conference workshop on Interval Constraints*, Portland, Oregon, USA, 1995.
28. P. Van Hentenryck, D. McAllester, and D. Kapur. Solving Polynomial Systems Using a Branch and Prune Approach. *SIAM Journal on Numerical Analysis*. (To appear).
29. P. Van Hentenryck, H. Simonis, and M. Dincbas. Constraint Satisfaction Using Constraint Logic Programming. *Artificial Intelligence*, 58(1-3):113–159, Dec. 1992.
30. D. L. Waltz. Generating Semantic Descriptions from Drawings of Scenes with Shadows. In P. H. Winston, editor, *The Psychology of Computer Vision*. McGraw Hill, 1975.

This article was processed using the LaTeX macro package with LLNCS style