Consistency Techniques for Numeric CSPs

Olivier Lhomme

Dassault Electronique
55, Quai Marcel Dassault,
92214 Saint-Cloud, France

also at: I3S, University de Nice
250, avenue Einstein,
06560 Valbonne, France

Email : lhomme@dassault-elec.fr

Abstract

Many problems can be expressed in terms of a numeric constraint satisfaction problem over finite or continuous domains (numeric CSP). The purpose of this paper is to show that the consistency techniques that have been developed for CSPs can be adapted to numeric CSPs. Since the numeric domains are ordered the underlying idea is to handle domains only by their bounds. The semantics that have been elaborated, plus the complexity analysis and good experimental results, confirm that these techniques can be used in real applications.

1 Introduction

Artificial intelligence, operational research and, more recently, logic programming often use the concept of constraint in order to express and solve a problem. Bit by bit a theoreucal framework has evolved, that of Constraint Satisfaction Problems (CSP) [Waltz, 1972; Montanari, 1974; Mackworth, 1977]. A CSP is defined by a set of variables each with an associated domain of possible values and a set of constraints on the variables.

This paper deals with CSPs where the constraints are numeric relations and where the domains are either finite integer domains or continuous domains (numeric CSPs). [Davis, 1987] offers a good insight into this kind of problem. Methods exist to solve numeric constraint systems in certain special cases (Simplex algorithm, Grtibncr bases, Newton's method, etc.) but no general method exists.

Consistency techniques have been successfully applied to general CSPs and could be very useful for numeric constraint systems, but we shall sec that even a very simple consistency technique such as arc consistency may turn out to be unrealistic (although the characteristics of numeric CSPs can be used to greatly reduce the complexity of such techniques: (1) numeric constraints are expressed intensionally and (2) the domains are ordered and can also be expressed intensionally).

Interval propagation is another technique, used for solving numeric CSPs. It consists in propagating the bounds of the domains and is often used in implementations either on finite integer domains such as CHIP [Dincbas *et al.,* 1988] and OSL [IBM, 1991] or continuous ones such as BNR-Prolog [Older and Vellino, 1990] and Interlog [Tosello, 1990; Dassault Electronique, 1991]. But interval propagation has never been formalized by a concept of consistency and no complexity analysis has ever been performed.

This paper is a presentation of two new partial consistencies, their specificity being that they only consider the bounds of the domains. The first one formalizes interval propagation, the second, which is stronger than the first, has given rise to a new algorithm (implemented in Interlog). An extension of these consistencies enables the running lime of algorithms to be tuned.

The paper is organized as follows. Chapter 2 describes the problem. Section 2.1 considers the usefulness of consistency techniques in numeric CSPs. Section 2.2 introduces the abstract concept of *numeric value,* thanks to which it *is* possible to provide definitions and algorithms that are valid for both finite and continuous domains. Section 2.3 defines the formalism that has been adopted.
Chapter 3 discusses solving numeric CSPs and first shows that are consistency may be unrealistic on large size domains. Section 3.1 defines arc B-consistency (where *B* stands for *bounds)* which formalizes the interval propagation technique. Section 3.2 introduces 3-B-consistency which is a stronger consistency on domain bounds than arc B-consistency. The algorithms that perform these two consistencies are given and their complexity is analyzed. Section 3.3 contains some experimental results.

The whole paper has been written with continuous domains in mind (the examples always involve real variables) but it is obvious that everything is equally valid for finite integer domains. The proof of the propositions and the complexity analysis can be found in [Lhomme, 1992].

2 Numeric CSPs

2 .1 Why consistency techniques are useful

Numeric CSPs can be used to express a large number of problems, in particular physical models involving inaccurate data or partially defined parameters. Here it is a question of what solving such problems implies. These problems are generally under-constrained, i.e. a very large number of solutions exists (infinitely large in the case of continuous domains). Although it is impossible to enumerate all of them, it is often possible to express the set of solutions.

Example: Let three real variables be linked by a constraint U = R * I. If we know that I varied between 1 and 2 and that R varies between 10 and 11, it is not so much one solution that is worth knowing (e.g. R-10.53,

I=1, U=10.53) at the exact range of values of U (i.e. the interval [10, 72)).

In this example the domains are continuous and so it is impossible to enumerate all the solutions. The constraint U = R * I is non linear, the Simplex cannot apply. The Grobner bases method can be used to solve polynomial equations but cannot take inequations into account.

The difficulty comes from the fact that we are trying to reason on the range of values of variables. A natural method would therefore be to attribute a dynamic domain to each variable and to propagate this domain through the constraints. This is exacdy what the CSP consistency techniques [Mackworth, 1977] do. We shall use these techniques to try to determine the exact range of values of a set of constrained variables.

Traditionally, domains and consistency techniques in CSPs are only used to simplify a problem before going on to enumerate the solutions. In numeric CSPs, however, one of the main ideas is to consider that the domain of a variable is an approximation of the exact range of values of the variable and may itself be of use. This interpretation is similar to Hyvonen's *tolerance propagation* [Hyvonen, 1989].

Independently of the very large, or even infinitely large number of solutions, another advantage of working on ranges of values rather than on single values is that it allows *dynamic* numeric CSPs to be handled, i.e. numeric CSPs to which constraints may be added. Incrementality is therefore crucial and requires delaying choice points (which list the solutions to an intermediate numeric CSP) as long as possible. Incrementality is a key feature when integrating consistency techniques in a programming language (see [Cleary, 1987; Dincbas et al., 1988; Older and Vcllino, 1990; Sidcbottom and Havens, 1991; Lee and Van Emden, 1992; Benhamou and Older, 1992]).

22 The concept of numeric values

In order to be able to use the same language both for finite and for continuous domains, the abstract concept of *numeric value* has been adopted. It will be defined separately for finite domains and continuous ones.

When a domain is finite, a numeric value is simply defined by:

Definition Ia: numeric values in finite domains
For a finite numeric domain D a numeric value is an element ofD.

In the case of continuous domains, the domain associated with a variable represents an infinite set of real numbers. However, since a computer handles real numbers through floating-point numbers it is important to have a formalism that allows for this, without which no valid result could ever be obtained. In floating-point representation the set of values that a numeric variable can take is the set of floating-point numbers $\{f_0,...f_n\}$. A real number between two consecutive floating-point numbers is usually approximated. In order to avoid approximations and guarantee correct results, a numeric value in a continuous domain is defined as follows:

Definition Ib: numeric values in continuous domains
A numeric value is an element of the set $\{ (-\infty f_0) , f_0 , (f_0 f_1) , f_1 , (f_1 f_2) , f_2 , ... , f_n , (f_n,+\infty) \}$.

A numeric value is either a floating-point number *f* or an open interval whose bounds are two consecutive floating point numbers[1] $(f, f+)$ or one of the terms $(-\infty f_0)$ or $(f_n,+\infty)$ representing the infinities. The interval $(f, f+)$ represents an infinite set of real numbers but will be considered as a *single* value. This definition enables us to consider the domain associated with a variable as a finite set while preserving the continuity of the domain. Note that an open interval will also be represented by a finite set (and will therefore be closed).

***Example*: The half-open interval [2,3) shall be considered as the finite set { 2 , (2,2+) , 2+ , ..., 3- , (3-,3) } and shall be represented in our notation by a closed interval [a,b] with a=2 and b=(3-,3).**

Interval arithmetic [Moore, 1966; Alefeld and Herzberger, 1983] provides computing methods that respect this formalism (and in particular allow the symbolic handling of infinities). Whereas computations performed using a floating-point representation are a mere estimate of the correct result, interval-based computing methods provide an interval and guarantee[2] bounds for this correct result. The advantage of interval arithmetic is not just a matter of checking computing errors; it also offers ways to reason on the range of values of variables.

2.3 Definitions

By NCSP we mean a Numeric Constraint Satisfaction Problem. The corresponding definitions are almost the same as for CSPs [Mackworth, 1977] except that the constraints cannot be given extensionally.

Definition 2: an NCSP
An NCSP P = (V,D,C) is defined by
- a set of numeric variables $V = \{X_1,... X_n\}$
- a set of domains $D = \{D_1,... D_n\}$ *where* D_i *, a set of numeric values, is the domain associated with the variable* X_i
- a set of constraints $C = \{C_1,... C_m\}$ *where a constraint* C_i *is defined by any numeric relation linking a set of variables.*

***Example*: Figure 1 defines three NCSPs that will be used later (P_1, P_2 and P_3 differ only by their domains).**

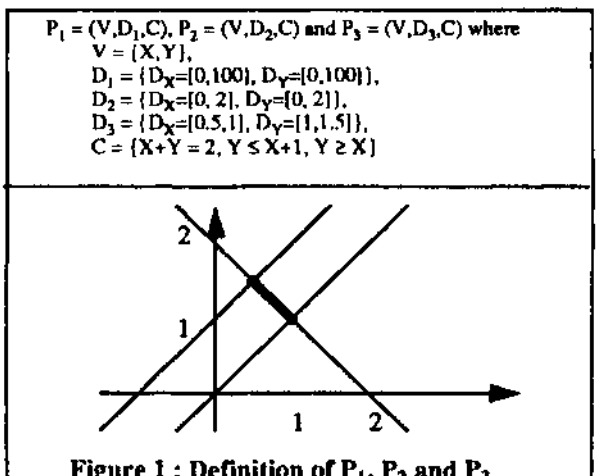

Figure 1 : Definition of P_1, P_2 and P_3

[1] Notation: if *f* is a floating-point number, *f-* and *f+* are respectively the two floating-point numbers immediately below and immediately above *f*.

[2] This guarantee comes from the basic property of interval arithmetic [Moore, 1966]: let I be an interval, f a function and F the extension of f to the intervals, so that, $\forall x \in I, f(x) \in F(I)$.

Definition 3: solution to an NCSP
A solution to an NCSP P = (V,D,C) is an instantiation of the variables of V for which both inclusion in the associated domains and all the constraints of C are satisfied.

Example: {X=1,Y=1} and {X=(1-,1), Y=(1,1+)} are two solutions[1] to P_1.

Definition 4: global consistency of an NCSP
Let P = (V,D,C) an NCSP, P is globally consistent iff[2] $\forall X \in V, \forall a \in D_X$, *X = a belongs to a solution to P.*

Global consistency reflects the fact that for any variable X of the problem, the domain D_X is the projection of the set of solutions to *P* on X. Unlike Freuder's global consistency for CSPs [Freuder, 1988], this definition allows constraints that are given intensionally.

Example: It is easy to verify that P_3 is globally consistent.

Definition 5: equivalence of two NCSPs
P and P' are equivalent (P ≡ P') iff they have the same set of solutions.

Example: $P_1 \equiv P_2 \equiv P_3$.

We will use the following terminology when talking about consistency techniques (see [Jégou, 1991]). A *partial consistency* λ (e.g. λ = arc or path consistency) is a property of a CSP. Let *P* be a CSP. *Closure by λ of P*, denoted $\Phi_\lambda(P)$, is informally the largest CSP included in *P* for which λ consistency holds. A λ *filtering* algorithm computes $\Phi_\lambda(P)$ and is usually called a consistency technique.

Let us now look at a number of concepts that will be used later on.

If D_X is the domain of a variable X, we say that D_X is *convex* iff all the numeric values between min(D_X) and max(D_X) belong to D_X. If D_1 ...., D_k are convex then $E = D_1 \times ... \times D_k$ shall be called a *box*.

Definition 6: projection of a constraint
Let X_1,..., X_k be k variables, let $E = D_1 \times ... \times D_k$. The projection over X_i of the constraint $C(X_1,..., X_k)$ restricted to domains D_1,..., D_k that shall be denoted $\Pi_i(C(X_1,...,X_k), E)$ is the set: $\{ v_i \in D_i / \exists (v_1,..., v_{i-1}, v_{i+1},...,v_k) \in D_1 \times ... \times D_{i-1} \times D_{i+1} \times ... \times D_k$ such that $C(v_1, ..,v_k)$ is satisfied$\}$.

Example: Given a constraint $Y = X^2$. Let $D_X = [-2,+2]$, $D_Y = [1,10]$, $E = D_X \times D_Y$,
so $\Pi_X(\{Y = X^2\}, E) = [-2,-1] \cup [1,2]$ and $\Pi_Y(\{Y = X^2\}, E) = [1,4]$.

For reasons of efficiency of implementations the concept of *basic constraint* now needs to be introduced.

Definition 7: basic constraint
A constraint $C(X_1,..., X_k)$ is said to be basic iff $\forall i \in 1 .. k$, *it is possible to show two functions F_i^{min} and F_i^{max} such that, for any box $E = D_1 \times ... \times D_k$, $F_i^{min}(E)$ and $F_i^{max}(E)$ are the min and max values of $\Pi_i(C(X_1,..., X_k), E)$.*

Please note that the rest of this paper assumes that the constraints are basic.

Remark 1: a small set of basic constraints enables a large number of basic constraints to be expressed (through the conjunction[3] of constraints and addition of intermediate variables). In Interlog, over continuous domains, given X, Y and Z distinct variables, the following set { $X = Y$, $X \leq Y$, $Z = X+Y$, $Z = X*Y$, $Y = -X$, $Y = \sin X$, $Y = \cos X$, $Y = e^X$, $Y = abs(X)$, $Z = X^Y$, $Z = \min(X,Y)$, $Z = \max(X,Y)$ } is a set of basic constraints, thanks to interval arithmetic. The constraint $Z = X*X+Y$ is basic but it is not in the set; it may be expressed by $(Z = V1+Y) \& (V1 = X^2)$.

Remark 2: when a constraint is not basic it is often possible to transform[4] it into a conjunction of basic constraints (e.g. $e^X = \sin(X+Y)$ becomes $(e^X = V1) \& (V1 = \sin V2) \& (V2=X+Y)$).

A constraint is said *disjunctive* if it "destroys" the convexity of its domains. This concept depends on the domain of variables: the same constraint can be disjunctive for certain domains but non disjunctive for others.

Definition 8: disjunctive constraints
A constraint $C(X_1,..., X_k)$ is disjunctive for the box $E = D_1 \times ... \times D_k$ iff
$\exists i \in 1 .. k, \Pi_i(C(X_1,..., X_k), E)$ *is non convex.*

Examples: The constraint $X^2 = Y$ is non disjunctive for $D_X=[-2, 2]$ and $D_Y = [0, 4]$, is disjunctive for $D_X=[-2,2]$ and $D_Y = [1, 4]$, is non disjunctive for $D_X=[1, 2]$ and $D_Y = [1,4]$.

3 How to solve an NCSP

The goal here is neither to enumerate all the solutions nor to algebraically solve a system of constraints but to determine the exact ranges of values of the different variables. If it were possible to find an NCSP that was equivalent and globally consistent the goal would be reached. But the search for global consistency is an NP-hard problem[5].

A natural approach, that has proved its worth on CSPs, is to begin by trying to reach partial consistency. Arc consistency[6] could seem a good beginning. Unfortunately, if there are any disjunctive constraints in the system arc consistency can rapidly lead to combinatorial explosion. If we look at the example below we shall soon see that this approach is unsuitable.

Example: $D_X = [0, 31416]$, $D_Y = [0, 31416]$, $D_Z = (-\infty, +\infty)$, $\sin X = 0.2$, $\cos Y = 0.1$, $Z = X * Y$.
Arc consistency would lead to the union of 10^4 intervals for D_x and D_Y (10^4 = the number of monotonic parts of sine and cosine in [0,31426)) and to 10^8 intervals for D_z. Such a combinatorial explosion is totally unreasonable.

The above example is an extreme case but there are only three constraints! Even without this kind of extreme case, as soon as the number of constraints becomes rather large there is a risk of combinatorial explosion due to the representation of the domains.

In NCSPs the domains that are handled are finite but of a very particular kind: they are by definition very large[7] and are

Remember that operations are performed by interval arithmetic methods: so (1-,1) + (1,1+)= [a,b] and [a,b] is such that [a,b] contains 2.

[2] if and only if.

[3] The conjunction of basic constraints is not always a basic constraint.

[4] Under such transformations, global consistency is invariant but partial consistencies are generally not invariant.

[5] It is an uncomputable problem over the real numbers (because of the transcendental functions) but in finite precision it becomes an NP-hard problem.

[6] Let P be an NCSP, X a variable of P. D_X is arc consistent iff
$\forall C(X, X_1, ..., X_k)$ a constraint over X, $\forall v \in D_X$,
$\exists v_1, ..., v_k \in D_1 \times ... \times D_k \mid C(v, v_1, ..., v_k)$ is satisfied.
An NCSP is arc consistent iff all the domains are arc consistent.

[7] The number of 64-bit floating-point numbers between 0.0 and 1000.0 is of the order of $4.6 * 10^{18}$ on an IBM 3090.

ordered. A very economical way of representing a convex domain is to do so intensionally by just one interval (i.e. two bounds). Accordingly, if we only handle convex domains, the combinatorial explosion due to the representation of domains can be avoided. This brings us on to the definition of new partial consistencies. Whereas partial consistencies arising from CSPs guarantee conditions over all the elements of a domain, the new consistencies will guarantee conditions only over the bounds of the domain and will thus preserve convexity.

Moreover the constraints are also given intensionally. Interval arithmetic will thus make it possible to give very efficient filtering algorithms.

Subsequently only convex domains will be handled and the domain associated with a variable will simply be an interval. In addition, thanks to our representation of open intervals (see section 2.2), only closed intervals have to be handled.

3.1 Arc B-consistency and interval propagation

Arc B-consistency (*B* for *bounds*) is a partial consistency defined as follows:

Definition 9: arc B-consistency
Let P be an NCSP, X a variable of P, D_X = [a,b]. D_X is arc B-consistent iff $\forall$ *C(X, X_1, ... ,X_k) a constraint over X*
$\exists\ v_1, ..., v_k \in D_1 \times ... \times D_k$ / C(a, v_1, ..., v_k) *is satisfied,*
$\exists\ v_1, ..., v_k \in D_1 \times ... \times D_k$ / C(b, v_1, ..., v_k) *is satisfied.*
An NCSP is arc B-consistent iff all the domains are arc B-consistent.

Informally arc B-consistency is a form of arc consistency that is restricted to the bounds of the domain.

Example: It is easy to verify that P_2 (see Figure 1) is arc B-consistent.

The difference between arc B-consistency and arc consistency comes from the way the disjunctive constraints are handled. Arc B-consistency encompasses in one single interval the domain of a variable; some values of a domain may be locally inconsistent.

Example 1: Let P be defined by
D_X = [1,4], D_Y = [-2,+2],
$X = Y^2$.
P is arc B-consistent but not arc consistent (the value 0 of D_Y does not correspond to any value of D_X such that the constraint can be satisfied).

Closure by arc B-consistency is defined as follows:

Definition 10: closure by arc B-consistency
Given an NCSP P where P = (V,D,C). Closure by arc B-consistency of P, denoted $\Phi_{AB}(P)$, *is an NCSP P' where P' = (V, D',C) and P' such that:*
P' ≡ P,
P' is arc B-consistent,
$\forall\ D_i' \in D', D_i' \subseteq D_i$,
there is no P"=(V, D", C) such that
P" is arc B-consistent, P" ≡ P, P"≠ P', $\forall\ D_i' \in D', D_i' \subseteq D_i$".

Example: We have $P_2 = \Phi_{AB}(P_1)$.

It can be proved that $\Phi_{AB}(P)$ always exists and is unique. $\Phi_{AB}(P)$ may be the empty NCSP (i.e. the domains are equal to $\emptyset$), in which case *P* has no solution.

By denoting closure by arc consistency of *P* as $\Phi_{AC}(P)$, the relations between arc B-consistency and arc consistency can be expressed by the following theorem.

Theorem 1
Given an NCSP P where P = (V,D,C) and D = {D_1, ... , D_n}. Let P' = $\Phi_{AB}(P)$ *and P" =* $\Phi_{AC}(P)$, *then* $\forall\ i \in 1 .. n, D_i'$ *is the smallest interval encompassing* D_i".

Corollary 1
Let P be an NCSP that contains no disjunctive constraints over the domains being considered. So P is arc B-consistent iff P is arc consistent.

Example: The NCSP P_2 is arc B-consistent, all the constraints of P_2 are non disjunctive, therefore P_2 is arc consistent. However, the NCSP of example 1 above (with the constraint $X = Y^2$) is not arc consistent.

Figure 2 shows an algorithm (called IP_1 for *interval propagation*) to compute closure by arc B-consistency. This algorithm looks very much like that of Waltz [1972] and also like AC-3 [Mackworth, 1977]. The difference lies in the way disjunctive constraints are processed, which is by bounding the result within a single interval (i.e. a constraint will not delete an impossible value between possible values).

```
procedure IP_1(inout P)
Precondition: P0 is an NCSP.
Postcondition: either P is the closure by arc B-consistency of P0
               or exit with failure.
Begin
1  Agenda := {<C, X> | C is a constraint of P, X is a variable of C}
2  while Agenda ≠ ∅
3     select and delete <C, X> from Agenda
4     REVISE( <C, X>, Result )
5     if Result = fail then exit with failure
6     if Result =changed then
7        Agenda:= Agenda ∪ {<C', X'> | C'≠C, X in C' and X'≠X }
8  end while
end IP_1

procedure REVISE( <C,X>, Result )
Begin
9  let F^min and F^max be the functions of definition 7 for C and X
10 m := F^min over D1 ... Dn ; M := F^max over D1 ... Dn
11 if m and M do not exist then Result := fail
12 elsif DX ≠ [m,M] then      % [m,M] is always included in DX
13    DX := [m,M]
14    Result := changed
15 else Result := nothing
end REVISE
```

Figure 2: IP_1

Description of IP_1. Lines 3 and 4 remove a pair <C, X> from the agenda and call the REVISE procedure which tries to restrict D_X. If D_X has been modified (line 6), the pairs <C', X'> that are likely to narrow a domain are added to the agenda (line 7).
The REVISE procedure uses the particularities of basic constraints: it is possible to compute the min and max of the projection of the constraint over the variable X (lines 9 and 10).

Terminating the IP_1 algorithm. Over continuous domains, IP_1 can lead to problems of termination.

Example: Let P be defined by:
(a) Y = X+1,
(b) Y = 2 * X,
D_X = [0,10], D_Y = (-∞,+∞).
D_X = [0,10] leads, according to (a), to D_Y = [1,11]
D_Y = [1,11] leads, according to (b), to D_X = [0.5 , 5.5]
D_X = [0.5 , 5.5] leads, according to (a), to D_Y = [1.5 , 6.5]
D_Y = [1.5 , 6.5] leads, according to (b), to D_X = [0.75 , 3.25]
...
A phenomenon of iteration and asymptotic convergence towards the solution (X=1, Y=2) occurs. In twenty iterations the degree of precision

obtained is less than 10^{-6} and gives the result $D_X = [0.999999, 1.000001]$, $D_Y = [1.999999, 2.000001]$.

Termination of this algorithm over continuous domains could naively give rise to two types of discourse.

- The first one says that since the number of floating point numbers is finite, termination can easily be shown (a domain can only be restricted). But this attitude is of very little practical interest and the algorithm can take a long time terminating.
- The second one is directly linked to the implementation. It involves interrupting propagation before normal termination (e.g. by giving a time limit or, when a domain is restricted to less than ε - absolute or relative - by not reactivating the constraints concerned). The problem this time is that, when the algorithm terminates, it is not possible to describe the status of the domains (the NCSP is neither arc B-consistent nor anything like it).

What could be done is a third way that avoids the drawbacks of the first two. This requires defining arc B(w)-consistency, which generalizes arc B-consistency, and then modifying IP_1 so as to compute filtering by arc B(w)-consistency. The w (as *width*) characterizes authorized imprecision at the bounds.

Notation: *if* v *is a numeric value and* k *a positive integer,* v^{+k} *(resp.* v^{-k}*) is the* k^{th} *greater (resp. smaller) value than* v *if it exists. If it does not exist,* v^{+k} *(resp.* v^{-k}*) will be the maximum (resp. minimum) numeric value. Note that if* k *equals* 0, $v^{+0} = v^{-0} = v$.

Definition 11: arc B(w)-consistency
Let P be an NCSP, X a variable of P, $D_X = [a,b]$, w *a positive integer.* D_X *is arc B(w)-consistent iff*
$\forall$ *C(X,* X_1, ... X_k*) a constraint over X*
$\exists v \in [a, a^{+w}], \exists v_1, ..., v_k \in D_1 \times ... \times D_k$ / *C(v,* v_1, ..., v_k*) is satisfied,*
$\exists v \in [b^{-w}, b], \exists v_1, ..., v_k \in D_1 \times ... \times D_k$ / *C(v,* v_1, ..., v_k*) is satisfied.*
An NCSP is arc B(w)-consistent iff all its domains are arc B(w)-consistent.

Arc B(0)-consistency is equivalent to arc B-consistency.

Proposition 1
Let P be an arc B(w)-consistent NCSP, let $w' \geq w$, *hence P is arc B(w')-consistent.*

It is possible to define a closure by arc B(w)-consistency. Unlike closure by arc B-consistency, this one is not unique and depends on the order in which the constraints are evaluated. The IP_2 algorithm shown in Figure 3 computes one of these closures, denoted $\Phi_{AB(w)}(P)$.

Description of IP_2. The IP_2 procedure differs from IP_1 in the parameter w and the function *SUFFICIENT_CHANGE*. The function *SUFFICIENT_CHANGE* is true if at least one of the bounds of D_X is more than w distant from the corresponding bound of [m,M]. When it can be seen that a constraint will move the two bounds of a domain by less than or equal to w, then (line 15) it is not applied (in fact it is satisfied in terms of arc B(w)-consistency). If this constraint were to be applied systematically[1] (if the assignment of line 13 were to be performed before the test of line 12), it would be impossible to define a strict semantics. Obviously, if $w = 0$, IP_2 is equivalent to IP_1.

Complexity. Let A be the maximum domain size, let $a = A/(w+1)$ (a characterizes the number of packets of $(w+1)$ values in a domain) and let m be the total number of constraints of P. The complexity analysis (see [Lhomme, 1992]) shows that the worst case running time of IP_2 is bounded below by $\Omega(m)$ and above by $O(am)$. The lower bound of the worst case running time of IP_2 is independent of A. This is all the more useful as the domains being handled are large.

The complexity of IP_1 follows immediately from this (by taking w=0, and therefore A=a). It is bounded by $\Omega(m)$ and $O(Am)$. For *non disjunctive constraints*, arc B-consistency is equivalent to arc consistency and IP_1 can be compared with arc consistency algorithms. Mackworth and Freuder [1985] have given the lower and upper bounds of the worst case running time of AC-3: $\Omega(A^2m)$ and $O(A^3m)$. AC-4 [Mohr and Henderson, 1986] does not apply here because the domain size is too large (AC-4 associates a data structure with each value of the domain). Deville and Van Hentenryck [1991] and Perlin [1992] suggest taking the constraint semantics into account. For certain classes of constraints, the worst case running time of their algorithms is also $O(Am)$.

```
procedure IP_2(inout P, in w)
Precondition: P0 is an NCSP.
Postcondition: either P is a closure by arc B(w)-consistency of P0
               or exit with failure
Begin
1  Agenda := {<C,X> | C is a constraint of P, X is a variable of C}
2  While Agenda ≠ Ø
3     select and delete <C, X> from Agenda
4     REVISE( <C, X>, w, Result )
5     if Result = fail then exit with failure
6     if Result =changed then
7        Agenda:= Agenda ∪ {<C', X'> | C'≠C, X in C' and X'≠X}
8  end while
end IP_2

procedure REVISE( <C,X>, w, Result )
Begin
9  let F^min and F^max be the functions of definition 7 for C and X
10 m := F^min over D1 ... Dn ; M := F^max over D1 ... Dn
11 if m and M do not exist then Result := fail
12 elsif SUFFICIENT_CHANGE([m, M], DX, w) then
13    DX := [m, M]
14    Result := changed
15 else Result := nothing
end REVISE
```

Figure 3: IP_2

3.2 Stronger consistency: 3-B-consistency

Neither arc B-consistecy nor arc consistency are always sufficient:

Example: P_2 is arc B consistent (and even arc consistent) but it does not allow the range of values of the variables to be found i.e. Dx=[0.5, 1], D_Y=[1.1.5].

In the case of CSPs over finite domains, an efficient way of finding solutions is often to perform an interleaved enumeration with arc consistency filtering. One of the ways of adapting this method to NCSPs is domain splitting [Cleary, 1987]: if P is arc B-consistent, the domain of a variable is split in two and the two resulting NCSPs are explored separately. This

[1] This is the case for BNR-Prolog, for example.

process introduces choice points and can rapidly lead to combinatorial explosion.

A different approach would be the definition of a stronger consistency than arc B-consistency (and that shall be called 3-B-consistency) and the description of a filtering algorithm that computes closure by 3-B-consistency. In order to avoid combinatorial explosion, 3-B-consistency guarantees conditions only on the bounds of the domains. Intuitively, 3-B-consistency can be seen as a form of strong 3-consistency [Freuder, 1988] restricted to the bounds of the domains.

Notation: *The union of an NCSP $P = (V,D,C)$ and a constraint K (which only constrains variables that are already present in P) is defined by: $P' = P \cup \{K\} = (V,D,C \cup \{K\})$*

Definition 12: 3-B-consistency
Let P be an NCSP, let X be a variable of P, and $D_X = [a,b]$. D_X is 3-B-consistent iff P' and P'' are both non empty
where $P' = \Phi_{AB}(P \cup \{X = a\})$ and $P'' = \Phi_{AB}(P \cup \{X = b\})$.
An NCSP is 3-B-consistent iff all its domains are 3-B-consistent.

Notation: $\Phi_{3\text{-}BC}(P)$ *represents the closure of P by 3-B-consistency.*

Example: P_1 and P_2 are not 3-B-consistent but P_3 is 3-B-consistent. In addition $P_3 = \Phi_{3\text{-}BC}(P_1) = \Phi_{3\text{-}BC}(P_2)$.

Before giving a 3-B-consistency filtering algorithm the concept of 3-B-consistency is generalized by 3-B($w1$,$w2$)--consistency so as to give the bounds a width (authorized imprecision).

Definition 13: 3-B(w1,w2)-consistency
Let P be an NCSP, X a variable of P, $D_X = [a,b]$, and $w1 \geq w2 \geq 0$.
D_X is 3-B($w1$,$w2$)-consistent iff P' and P'' are both non empty
where $P' = \Phi_{AB(w2)}(P \cup \{X \in [a, a^{+w1}]\})$
and $P'' = \Phi_{AB(w2)}(P \cup \{X \in [b^{-w1}, b]\})$.
An NCSP is 3-B($w1$,$w2$)-consistent iff all its domains are 3-B($w1$,$w2$)-consistent.

3-B-consistency is equivalent to 3-B(0,0)-consistency. The following proposition corresponds to proposition 1 for 3-B-consistency.

Proposition 2
Let P be a 3-B($w1$,$w2$)-consistent NCSP, $w1' \geq w1$, $w2' \geq w2$, hence P is 3-B($w1'$,$w2'$)-consistent.

A 3-B-consistency filtering algorithm is shown in Figure 4. It has been implemented in Interlog. The principle underlying this algorithm is based on proof by refutation, hence its name, *Ref_filtering*. Let us assume P to be an arc B-consistent NCSP, X a variable of P, and $D_X = [a,b]$. We are going to try to restrict D_X by increasing its lower bound. Let it be a point $c \in (a,b)$. Now add the constraint $X \in [a,c)$. If interval propagation detects a contradiction we can affirm that the range of values of X is included in $[c,b]$. If, on the other hand, no contradiction is detected, we cannot be sure of anything, but we can repeat the process with a point c' that is closer to a than c (for instance we could take c' midway between $[a,c]$).

Description. The index i specifies a pair (X, *bound*) where X is a variable and *bound* is either LOW or UP. The index i therefore varies between 1 and $2n$. The procedure IP_3 is an incremental version of IP_2: the first parameter is an NCSP P which is arc B($w2$)-consistent, and the second parameter is a constraint to add to P. The function *assign(i,size)* returns a constraint that corresponds to the instantiation of X at its bound enlarged to *size* elements. The function *delete_elements(i, size)* returns the complementary constraint of *assign(i,size)*. For instance, if $D_X = [a,b]$, and if *bound* equals LOW, *assign* returns the constraint "$X \in [a, a^{+size}]$" and *delete_elements* returns the constraint "$X \in [a^{+(size+1)}, b]$".

Complexity. Remember that m is the number of constraints, n is the number of variables, and A is the domain size. Note $a_1 = A/(w1+1)$ and $a_2 = A/(w2+1)$. The complexity analysis (see [Lhomme, 1992]) shows that the worst case running time of *Ref_filtering* is bounded below by $\Omega(n \log_2 a1)$ and above by $O(mn^2a_1a_2)$. Over finite integer domains, if we take $w1 = w2 = 0$ (i.e. $a_1 = a_2 = A$), these values become $\Omega(n \log_2 A)$ and $O(mn^2A^2)$.

```
procedure Ref_filtering(Inout P, In w1, In w2)
Precondition: P0 is an NCSP which is arc B(w2)-consistent.
Postcondition: either P is a closure by 3-B(w1,w2)-consistency of P0
               or exit with failure
begin
1  size := A /* A is an upper bound for the size of domains */
2  repeat
3     size := size div 2
4     repeat
5        fixed_point := true
6        i := 1
7        while i <= 2 * n
8           P' := P
9           Result := IP_3(P', assign(i,size), w2)
10          if Result = failure then
11             Result_2:=IP_3( P, delete_elements(i,size), w2)
12             if Result_2 = failure then exit with failure
13             fixed_point:= false
14          else
15             i := i+1
16    end {while}
17 until fixed_point
18until size <= w1
end Ref_filtering
```

Figure 4: Ref_filtering

It should be observed that this algorithm can be improved: corollary 2 defines for an NCSP $P = (V,D,C)$ a partition of V corresponding to equivalence classes. A closure by 3-B($w1$,$w2$)-consistency of P could thus be computed by considering just one representative of each class.

Theorem 2
Let P be an arc B(w2)-consistent NCSP, let X and Y be two variables of P linked by a binary, one-to-one monotonic relation over D_X and D_Y. If D_X is 3-B($w1$,$w2$)-consistent then D_Y is 3-B($w1'$,$w2$)-consistent with $w1' = constant * w1$.

The constant linking $w1$ and $w1'$ is a bound of the derivative of the relation over the domains D_X and D_Y.

Corollary 2
Let P be an arc B-consistent NCSP, let X and Y be two variables of P such that there exists a sequence $X_0, X_1, \ldots, X_k$ (with $X_0 = X$, $X_k = Y$) and X_i and X_{i+1} linked by a binary, one-to-one monotonic relation over D_i and D_{i+1}. If X is 3-B($w1$,$w2$)-consistent then Y is 3-B($w1'$,$w2$)-consistent with $w1' = constant * w1$.

Example: Let P be defined by
$X = 4*Z + 3$, $Z = 3*\log(Y-1)$, $T = Y\hat{}2$,
$D_X = D_Y = D_Z = D_T = [2,1000]$.

The four variables of P are in the tame equivalence class. All that needs to be done is to compute 3-B(w1,w2)-consistency on one of these variables in order to obtain 3-B(w1',w2)-consiftency of P.

33 Experimental results

Arc B-consistency filtering (i.e. interval propagation) is a technique that has already been used and validated experimentally on both finite domains (CHIP, OSL) and continuous domains (BNR-Prolog, Interlog). 3-B-consistency filtering, however, is new and deserves in-depth experimentation. Here, suffice it to say that in the case of P_1 and P_2 , *Ref filtering* would allow the range of values of each variable to be found rapidly (thus giving P_3), whereas the domain splitting technique (see section 3.2) would give a very huge number of contiguous solutions. In addition, even in more complicated cases such as that given in the example below, computation times have in practice turned out to be considerably lower than those predicted from the complexity analysis.

Example: Let P be the NCSP:

$D_X=[0,1000]$, $D_Y=[0,1000]$, $D_Z=[0,3.1416]$, $D_T=[0,3.1416]$,

$X*Y + T - 2*Z = 4$

$X*\sin Z + Y*\cos T = 0$

$X - Y + \cos^2 Z = \sin^2 T$

$X*Y*Z = 2*T$

with the solution $X = Y = 2$, $Z = \pi/2$, $T = \pi$. P is arc B-consistent and therefore interval propagation cannot narrow any of the domains.

Ref filtering, with *w1* and *w2* corresponding to a relative precision[1] of 10^{-4} gives the following result after roughly 10^4 elementary operations:

$D_X = [1.9999 , 2.0001]$,

$D_Y = [1.9999 , 2.0001]$,

$D_Z = [1.5636 , 1.5701]$,

$D_T = [3.1273 , 3.1416]$.

This system of constraints decomposes into a system of 22 basic constraints and 22 variables. On an IBM 3090 the number of numeric values in [0,1000] is roughly 10^{19}, and a relative precision of 10^{-4} corresponds to $w1 = w2 = 10^{12}$. Complexity analysis predicts a number of elementary operations between 500 and 10^{18}. Experimental results are relatively close to the lower bound.

The drawback of domain splitting in the above example is less serious than for P_1 and P_2 because the solution here is a single value. But it gives many quadruplets (D_x, D_y, D_z, D_T) that are close to the solution and is incapable of determining which one actually contains the solution.

4 Conclusion

CSP consistency techniques can be used in NCSPs to determine the range of values of variables. However their complexity remains too high, even though it can be greatly reduced by the structure specific to NCSPs.

The advantage of the two new partial consistencies, arc B-consistency and 3-B-consistency, is that they are well adapted to NCSPs. Their distinguishing feature is that they only consider the bounds of the domains. As for *non disjunctive* constraints, it has been shown that arc B-consistency is equivalent to arc consistency. An extension of these B-consistencies introduces the concept of bound width and enables the complexity of algorithms (even on continuous domains) to be tuned. In [Lhomme, 1993] these partial consistencies are generalized by defining k-B-consistency.

[1] On continuous domains, k $(\text{in } v^{+\infty})$ characterizes a *relative* imprecision on v (for $v \neq 0$), whereas on finite domains k characterizes an *absolute* imprecision.

Acknowledgments

Patrick Talllibert introduced me to the problems derscribed here and gave invaluable help in preparing this paper. My thanks also go to Michel Rueher, my PhD supervisor, for his useful suggestions and constructive criticism. I would also like to thank Bernard Botella, Philippe Jegou, Jimmy Lee, Philippe Marguerie and Franck Porcher for their useful comments on previous drafts of this paper. Finally, my thanks go to Rosalind Greenstein for the English version of this paper.

References

[Alefeld and Herzberger, 1983] G. Alefeld, J. Herzberger, Introduction to Interval Computations, Academic Press, 1983.

[Benhamou and Older, 1992] F. Benhamou, W.J. Older, "Applying Interval Arithmetic to Integer and Boolean Constraints", Technical Report, Bell Northern Research, 1992

[Cleary, 1987] J.C. Cleary, "Logical Arithmetic", Future Computing Systems, Vol. 2, Number 2, p. 125-149, 1987.

(Davis, 1987] E. Davis, "Constraint Propagation With Interval Labels", Artificial Intelligence 32. pp 281-331,1987.

[Dassault Electronique, 1991] "INTERLOG 1.0: User Guide" (in French), Dassault Electroniquc, 55 Quai M. Dassault, 92214 Saint Cloud, France, 1991.

[Deville and Van Hentenryck, 1991] Y. Deville, P. Van Hcntenryck, "An efficient Arc Consistency Algorithm for a Class of CSP Problems", in Proceedings of the 12th IJCAf, Sydney, 1991.

[Dincbas *et al,* 1988] M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf, F. Berthier, "The Constraint Logic Programming Langage CHIP", in Proceedings of the International Conference on Fifth Generation Computer Systems, Tokyo, Japan, 1988.

[Freuder, 1978] E. C. Freudcr, "Synthesizing Constraint Expressions", CACM, 21-11, p. 958-966, 1978.

[IBM, 1991]"IBM Optimization Subroutine Library Release 2 : User Guide", 1991.

[Hyvonen, 1989] E. Hyvonen, "Constraint Reasoning Based on Interval Arithmetic", Proceedings of the 11th LJCAI, Detroit, 1989.

[J6gou, 1991] P. Jegou, "Contribution to the Study of Constraint Satisfaction Problems" (in French), PhD Thesis, University Montpellier II, 1991.

[Lee and Van Emden, 1992] J.H.M. Lee, M.H. Van Emden, "Adapting CLP(R) to Floating-Point Arithmetic", Proceedings of the Fifth Generation Computer Systems Conference, 1992.

[Lhomme, 1992] O. Lhomme, "Numeric CSPs and Consistency Techniques" (in French), working document August 1992.

[Lhomme, 1993] O. Lhomme, "K-consistency like methods for Numeric CSPs", Technical Report NE 595 954, Dassault Electroniquc, 1993.

[Mack worth, 1977] A. K. Mack worth, "Consistency in Network of Relations", Artificial Intelligence 8, p. 99-118, 1977.

[Mackworth and Freuder, 1985] A. K. Mackworth, E. C. Freuder, "The Complexity of some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems", Artificial Intelligence 25, p. 65-74, 1985.

[Mohr and Henderson, 1986] R. Mohr, T.C. Henderson, "Arc and Path Consistency Revisited", Artificial Intelligence 28, p. 225-233, 1986.

[Montanari, 1974] U. Montanari, "Networks of Constraints: Fundamental Properties and Applications to Picture Processing", Information Sciences, 7, p. 95-132, 1974.

[Moore, 1966] R.E. Moore, Interval Analysis, Prentice Hall, New Jersey, 1966.

[Older and Vellino, 1990] W. Older, A. Vellino, "Extending Prolog With Constraint Arithmetic on Real Intervals", IEEE Canadian conf. on Electrical and Computer Engineering, 1990.

[Perlin, 1992] M. Perlin, "Arc Consistency for Factorable Relations", Artificial Intelligence 53, p. 329-342, 1992.

[Sidebottom and Havens, 1991] G. Sidebottom, W.S. Havens, "Hierarchical Arc Consistency Applied to Numeric Constraint Processing in Logic Programming", Technical Report CSS-IS TR 91-06, Centre for Systems Science, Simon Fraser University, Bumaby, B.C, Canada, 1991.

[Tosello, 1990] O. Tosello, "Constraints over Intervals in Prolog" (in French), Postgraduate Dissertation, Pierre and Marie Curie University, 1990.

[Waltz, 1972] D.L Waltz, "Generating semantic descriptions from drawings of scenes with shadows". Tech. Rept. AI-TR-271, MIT, Cambridge, MA, 1972.