Constraint Propagation for Soft Constraints: Generalization and Termination Conditions

S.Bistarelli[1], R.Gennari[2], F. Rossi[3]

1: Università di Pisa, Dipartimento di Informatica, Corso Italia 40, 56125 Pisa. E-mail: bista@di.unipi.it
2: ILLC, Institute of Logic, Language and Computation, University of Amsterdam, N. Doelenstraat 15, 1012 CP Amsterdam, The Netherlands. E-mail: gennari@hum.uva.nl
3: Università di Padova, Dipartimento di Matematica Pura ed Applicata, Via Belzoni 7, 35131 Padova. E-mail: frossi@math.unipd.it

Abstract. Soft constraints based on semirings are a generalization of classical constraints, where tuples of variables' values in each soft constraint are uniquely associated to elements from an algebraic structure called semiring. This framework is able to express, for example, fuzzy, classical, weighted, valued and over-constrained constraint problems.
Classical constraint propagation has been extended and adapted to soft constraints by defining a schema for *soft local consistency* [BMR97]. On the other hand, in [Apt99a,Apt99b] it has been proved that most of the well known constraint propagation algorithms for classical constraints can be cast within a single schema.
In this paper we combine these two schema and we show how the framework of [Apt99a,Apt99b] can be used for soft constraints. In doing so, we generalize the concept of soft local consistency, and we prove some convenient properties about its termination.

1 Introduction

Soft constraints allow to model faithfully many real-life problems, especially those which possess features like preferences, uncertainties, costs, levels of importance, and absence of solutions. Formally, a soft constraint problem (SCSP) is just like a classical constraint problem (CSP), except that each assignment of values to variables in the constraints is associated to an element taken from a set (usually ordered). These elements will then directly represent the desired features.

There are many formalizations of soft constraint problems. In this paper we consider the one based on semirings [BMR97], where the semiring specifies the partially ordered set and the appropriate operation to use to combine constraints together. This formalism has been shown to have many interesting instances, like classical, fuzzy [DFP93], weighted, valued [SFV95], probabilistic [FL93] and partial [FW92] constraints.

The propagation techniques usually used for classical CSPs have been extended and adapted to deal with soft constraints, provided that certain conditions are met. This has led to a general framework for soft constraint propagation, where at each step a subproblem is solved, as in classical constraint propagation [BMR97]. By studying the properties of this schema, it has been proved that such steps can be seen as applications of functions which are monotonic, inflationary, and idempotent over a certain partial order.

On an orthogonal line of research, the concept of constraint propagation over classical constraints has been studied in depth in [Apt99a,Apt99b], and a general algorithmic schema (called GI) has been developed. In such a schema, constraint propagation is achieved whenever we have a set of functions which are monotonic and inflationary over a partial order with a bottom.

By studying these two frameworks and comparing them, we noticed that the GI schema can be applied to soft constraints (see Section 5.1), since the functions and the order used for soft constraints have all the necessary properties for GI. This is proved in this paper by defining an appropriate partial order over soft constraint problems (see Section 3).

By analyzing the features of the GI algorithm, we also realized (see Section 5.2) that indeed soft constraint propagation can be extended to deal with functions which are not necessarily idempotent. Notice that this is a double generalization: we don't require any longer that each step has to solve a subproblem (it could do some other operation over the problem), nor that it is idempotent. This allows us to model several forms of "approximate" constraint propagation which were instead not modeled in [BMR97]. Example are: bounds-consistency for classical constraints [MS98], and partial soft arc-consistency for soft constraints [BCGR00].

These two results allow us to use the GI algorithm schema for performing a generalized form of soft constraint propagation. What is important to study, at this point, is when the resulting GI schema terminates. In fact, if we work with classical constraints over finite domains, it is easy to see that the GI algorithm always terminates. However, when moving to soft constraints over a semiring, even if the variable domain is finite, we could have an infinite behavior due to an infinite number of elements in the semiring. For example, fuzzy constraints have a semiring containing all reals between 0 and 1, and the semiring of weighted constraints contains all the reals, or all the naturals.

In Section 6 we identify some sufficient conditions for the termination of the GI algorithm over soft constraints. The first, predictable, condition that we consider is the well-foundedness of the partial order over soft constraint problems: if the partial order over which the GI algorithm works has chains of finite length, since constraint propagation never goes from one chain to another one, obviously the whole algorithm terminates.

The second condition is in some sense more precise, although less general: when the propagation steps are defined via the two semiring operations, then we can just consider the sub-order over semiring elements obtained by taking the elements initially appearing in the given problem, and closing it under the two operations. In fact, in this case the GI algorithm cannot reach other elements. Therefore, if such a set (or a superset of it) is well-founded, the GI algorithm terminates.

Each of these two conditions is sufficient for termination. However, they could be difficult to check, unless the partial order has a well-known structure of which we know the well-foundedness. Nevertheless, in a special case we can formally prove that there exists a well-founded set of the shape required by the second condition above, and thus we can automatically deduce termination. This special case is related to the idempotence of the multiplicative operation of the semiring, the one that we use to combine constraints: if this operation is idempotent, then GI terminates. For example, in classical constraints the multiplicative operation is logical and, and in fuzzy constraints it is the minimum, thus we can formally prove that the algorithm GI over *any* classical or fuzzy constraint problem *always* terminates, provided that the functions are defined via the two semiring operations.

We believe that the generalizations and termination conditions that we have developed and proved will make soft constraints more widely applicable, and soft constraint propagation more practically usable.

2 Soft Constraints

In the semiring-based formalism for soft constraints [BMR97], a soft constraint is just a constraint where each instantiation of its variables has an associated value. Combining constraints will then have to take into account such additional values, and thus the formalism has also to provide suitable operations for combination ($\times$) and comparison ($+$) of tuples of values and constraints. This is why the formalization adopted in [BMR97] uses a semiring structure, which is just a set plus two operations.

Semirings and SCSPs. A *semiring* is a tuple $\langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ such that: A is a set and $\mathbf{0}, \mathbf{1} \in A$; $+$ is commutative, associative and $\mathbf{0}$ is its unit element; $\times$ is associative, distributes over $+$, $\mathbf{1}$ is its unit element and $\mathbf{0}$ is its absorbing element. Further, we enforce some additional properties, leading to the notion of c-semiring: a *c-semiring* is a semiring $\langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ such that $+$ is idempotent with $\mathbf{1}$ as its absorbing element and $\times$ is commutative.

The partial order $\leq_S$ over the set A, defined as $a \leq_S b$ iff $a + b = b$, is used to compare elements in the semiring: $a \leq_S b$ means that b is "better" than a.

A *constraint system* is a tuple $CS = \langle S, D, V \rangle$ where S is a c-semiring, D is a finite set (the domain of the variables) and V is a finite ordered set of variables.

Given a semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ and a constraint system $CS = \langle S, D, V \rangle$, a *constraint* is a pair $\langle def, con \rangle$ where $con \subseteq V$ and $def : D^{|con|} \rightarrow A$. Therefore a constraint specifies a set con of variables and assigns each tuple of values of these variables an element of the semiring.

A *Soft Constraint Satisfaction Problem* (SCSP) on a constraint system CS is a pair $P = \langle C, con \rangle$, where $con \subseteq V$ and C is a set of constraints: con represents the set of variables of interest for the constraint set C, which however may contain constraints defined on variables not in con.

Combining and projecting soft constraints. Given two constraints $c_1 = \langle def_1, con_1\rangle$ and $c_2 = \langle def_2, con_2\rangle$, their *combination* $c_1 \otimes c_2$ is the constraint $\langle def, con\rangle$ defined by $con = con_1 \cup con_2$ and $def(t) = def_1(t \downarrow^{con}_{con_1}) \times def(t \downarrow^{con}_{con_2})$, where $t \downarrow^X_Y$ denotes the projection of tuple t from X onto Y. In words, combining two constraints means building a new constraint involving all the variables of the original one; the new constraint associates to each tuple of domain values a semiring element obtained by multiplying the elements associated by the original constraints to the appropriate subtuples.

Given a constraint $c = \langle def, con\rangle$ and a subset I of V, the *projection* of c onto I, written $c \Downarrow_I$, is the constraint $\langle def', con'\rangle$ where $con' = con \cap I$ and $def'(t') = \sum_{t | t\downarrow^{con}_{I\cap con} = t'} def(t)$. Thus, projecting means eliminating some variables. This is done by associating each tuple t over the remaining variables a semiring element; the last one is the sum of the elements associated by the original constraint to all the extensions of the tuple t over the eliminated variables.

In brief: combination is performed via the multiplicative operation of the semiring, and projection via the additive one.

Examples. Classical CSPs are SCSPs where the chosen c-semiring is $Bool = \langle\{false, true\}, \vee, \wedge, false, true\rangle$. By means of $Bool$ we can associate each tuple of elements in D a Boolean value, $false$ or $true$, then project and combine constraints via the Boolean connectives.

Fuzzy CSPs [DFP93] can instead be modeled by choosing the c-semiring $Fuzzy = \langle[0, 1], max, min, 0, 1\rangle$. In fact, there each tuple has a value between 0 and 1; constraints are combined via the min operation and compared via the max operation.

Solutions. The *solution* of an SCSP $P = \langle C, con\rangle$ is the constraint $Sol(P) = (\bigotimes C) \Downarrow_{con}$: it is obtained by combining all constraints of P and then projecting over the variables in con. In this way we get the constraint over con that is "induced" by the entire problem P.

Two problems $P_1 = \langle C_1, con\rangle$ and $P_2 = \langle C_2, con\rangle$ (notice that they have the same con) are considered equivalent, and written as $P_1 \equiv_P P_2$, when $Sol(P_1) = Sol(P_2)$.

Soft local consistency. Most of the traditional *local consistency* (also called *propagation*) algorithms can be generalized to SCSPs [BMR97]. In order to define local consistency algorithms for SCSPs, the notion of *local consistency rule* is introduced. The application of one of such rules consists of solving a subproblem of the given problem.

To model this, we use the notion of *typed location*. Informally, a typed location is just a location l (as in ordinary store-based programming languages) which has a set of variables con as type, and thus can only be assigned a constraint

$c = \langle def, con \rangle$ with the same type. In the following we assume to have a location for every set of variables, and thus we identify a location with its type.

Given an SCSP $P = \langle C, con \rangle$, the *value* $[l]_P$ *of the location* l in P is defined as the constraint $\langle def, l \rangle \in C$ if it exists, as $\langle \mathbf{1}, l \rangle$ otherwise. Given n locations $l_1, \ldots, l_n$, the value $[\{l_1, \ldots, l_n\}]_P$ of this set of locations in P is defined as the set of constraints $\{[l_1]_P, \ldots, [l_n]_P\}$.

An *assignment* is a pair $l := c$ where $c = \langle def, l \rangle$. Given an SCSP $P = \langle C, con \rangle$, the result of the assignment $l := c$ is the problem $[l := c](P)$ defined as: $[l := c](P) = \langle \{\langle def', con' \rangle \in C \mid con' \neq l\} \cup c, con \rangle$.

Thus an assignment $l := c$ is seen as a function from constraint problems to constraint problems, that modifies a given problem by changing just one constraint, namely the one with type l. The change consists in substituting such a constraint with c. If there is no constraint of type l, then the constraint c is added to the given problem. In other words, the assignment $l := c$ in P produces a new problem P' which is the same as P, except that it has an additional constraint c over the variables in l, and that the old constraints over l are removed. Note also that when $\mid l \mid = 1$ we are able to modify domains, since a domain can be seen as a unary constraint.

Consider a constraint system $CS = \langle S, D, V \rangle$, a location l and a set of locations L, where $l \in L$; a *local consistency rule* r_l^L is a function r_l^L which, taken any problem P over CS, returns $r_l^L(P) = [l := Sol(\langle [L]_P, l \rangle)](P)$.

Intuitively, the application of r_l^L to P adds the constraint $Sol(\langle [L]_P, l \rangle)$ over the variables in l to P. This constraint, by definition of Sol, is obtained by combining all constraints of P identified by L and then projecting the resulting constraint over l.

Since a local consistency rule is a function from problems to problems, the application of a sequence S of rules to a problem is easily provided by function composition: we write $[r; S](P) = [S]([r_1](P))$ and mean that the problem $[r; S](P)$ is obtained applying *first* the rule r and *then* the rules of the sequence S in the specified order.

An infinite sequence T of rules of a set R is called a *strategy*. A strategy is *fair* if each rule of R occurs in it infinitely often.

We are now ready to define *local consistency algorithms*. Given a problem P, a set of rules R and a fair strategy T for R, a *local consistency algorithm* applies to P the rules in R in the order given by T. The algorithm stops when the current problem is a fixpoint of all functions from R. In that case, we write $lc(P, R, T)$ to denote the resulting problem.

3 Some useful orderings over semiring constraints

We now review and modify some of the orderings among semiring elements, constraints, and problems, which have been introduced in [BMR97]; moreover, we also define new orderings that will be used in the next sections.

3.1 Semiring order

All the orderings we will consider in this section are derived from the partial order $\leq_S$ over semiring elements, which, we recall, is defined as follows: $a \leq_S b$ iff $a + b = b$. This intuitively means that b is "better" than a.

Definition 1. *Consider any partial ordering $\langle D, \sqsubseteq \rangle$ and the component-wise ordering $\langle D^n, \sqsubseteq_n \rangle$, with $n \geq 1$, where $\langle d_1, \ldots, d_n \rangle \sqsubseteq_n \langle d'_1, \ldots, d'_n \rangle$ iff $d_i \sqsubseteq d'_i$ for each $i = 1, \ldots, n$. Let f be a function from D^n to D. Then:*

- *f is monotonic iff $\langle d_1, \ldots, d_n \rangle \sqsubseteq_n \langle d'_1, \ldots, d'_n \rangle$ implies $f(\langle d_1, \ldots, d_n \rangle) \sqsubseteq f(\langle d'_1, \ldots, d'_n \rangle)$;*
- *f is inflationary w.r.t. $\sqsubseteq$ iff $d_i \sqsubseteq f(\langle d_1, \ldots, d_n \rangle)$ for every $i = 1, \ldots, n$.*

Given the definition above, it is easy to see that the following results hold when D is the semiring set A and the order considered is $\leq_S$:

- $\leq_S$ is a partial order;
- $\mathbf{0}$ is the minimum;
- $\mathbf{1}$ is the maximum.
- if $\times$ is idempotent, then $\langle A, \leq_S \rangle$ is a distributive lattice where $+$ is the lub and $\times$ is the glb.
- $+$ and $\times$ are monotonic with respect to $\leq_S$;
- $+$ is inflationary with respect to $\leq_S$; instead $\times$ is inflationary with respect to $\geq_S$.

3.2 Constraint order

From the ordering $\leq_S$ over A, we can also define a corresponding order between constraints. Before introducing the new order we define its domain, namely the set of all possible constraints over a constraint system.

Definition 2. *Given a semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ and a constraint system $CS = \langle S, D, V \rangle$, we define the* Constraint Universe *related to the constraint system CS as follows: $\mathcal{C}_{CS} = \bigcup_{con \subseteq V} \{\langle def, con \rangle \mid def : D^{|con|} \rightarrow A\}$.*

We will write $\mathcal{C}$ (instead of C_{CS}) when the constraint system CS is clear from the context.

Definition 3. *Consider two constraints c_1, c_2 over a constraint system CS; assume that $con_1 \subseteq con_2$ and $\mid con_2 \mid = k$. Then we write $c_1 \sqsubseteq_S c_2$ if and only if, for all k-tuples t of values from D, $def_2(t) \leq_S def_1(t \downarrow^{con_2}_{con_1})$.*

Loosely speaking, a constraint c_1 is smaller than c_2 in the order $\sqsubseteq_S$ iff it constrains possibly less variables and assigns to each tuple a greater value with respect to $\leq_S$ than c_2 does.

Theorem 1 ($\sqsubseteq_S$ is a po). *Given a semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ with $\times$ idempotent and a constraint system $CS = \langle S, D, V \rangle$, we have the following:*

- *the relation $\sqsubseteq_S$ is a partial order over the set $\mathcal{C}_{CS}$;*
- *its bottom is $\langle \mathbf{1}, \emptyset \rangle$, where the 0-arity function $\mathbf{1} : \emptyset \to A$ is the constant $\mathbf{1}$ of the semiring.*

Proof. We prove our first claim. We need to demonstrate that $\sqsubseteq_S$ is a reflexive, antisymmetric and transitive relation. Reflexivity holds trivially. To prove antisymmetry, suppose that $c_1 \sqsubseteq_S c_2$ and $c_2 \sqsubseteq_S c_1$; this yields that $con_1 = con_2$. Now, for all $t \in D^{|con_1|}$, we have both $def_1(t) \leq_S def_2(t)$ and $def_2(t) \leq_S def_1(t)$, hence $def_1(t) = def_2(t)$ and so $c_1 = c_2$. The transitivity of $\sqsubseteq_S$ follows from the transitivity of $\leq_S$. The other claim immediately follows from the definition of $\sqsubseteq_S$. □

3.3 Constraint set order

We can easily extend the order $\sqsubseteq_S$ over constraints to a new order over *constraint sets* as follows.

Definition 4. *Consider two sets of constraints C_1, C_2 over a constraint system CS. Suppose furthermore that $C_1 = \{c_i^1 : i \in I\}$, $C_2 = \{c_j^2 : j \in J\}$, $I \subseteq J$ and that, for every $i \in I$, the relation $c_i^1 \sqsubseteq_S c_i^2$ holds. Then we write $C_1 \sqsubseteq_C C_2$.*

The intuitive reading of $C_1 \sqsubseteq_C C_2$ is that C_2 is a problem generally "more constraining" than C_1 is, because C_2 has (possibly) a larger number of "more restrictive" constraints than C_1 has.

Theorem 2 ($\sqsubseteq_C$ is a partial order). *Given a semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$, and a constraint system $CS = \langle S, D, V \rangle$, we have that:*

- *the relation $\sqsubseteq_C$ is a partial order over $\wp(\mathcal{C})$;*
- *the bottom of the relation is $\emptyset$.*

Proof. We only prove the first claim, the other one being straightforward. Reflexivity trivially holds. As far as antisymmetry is concerned, suppose that $C_1 = \{c_i^1\}_{i \in I}$, $C_2 = \{c_j^2\}_{j \in J}$ and both $C_1 \sqsubseteq_C C_2$ and $C_2 \sqsubseteq_C C_1$ hold; this means that $I = J$. Moreover, the following relations hold for every $i \in I$: $c_i^1 \sqsubseteq_S c_i^2$ and $c_i^2 \sqsubseteq_S c_i^1$. Hence $c_i^1 = c_i^2$ for every $i \in I$, because $\sqsubseteq_S$ is a partial order relation, cf. Theorem 1. Transitivity follows similarly, by exploiting the transitivity of $\sqsubseteq_S$. □

3.4 Problem order

So far, we have introduced two partial orders: one between constraints ($\sqsubseteq_S$) and another one between constraint sets ($\sqsubseteq_C$). However, local consistency algorithms take constraint problems as input; therefore we need an ordering relation between problems if we want the `GI` algorithm to be used for soft local consistency.

First we define the set of all problems that can be built over a constraint system CS.

Definition 5. *Given a semiring* $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ *and a constraint system* $CS = \langle S, D, V \rangle$*, we define the* Problem Set Universe *related to the constraint system* CS *as* $\mathcal{P}_{CS} = \{\langle C, con \rangle \mid C \subseteq \mathcal{C}_{CS}, con \subseteq var(C)\}$*. When no confusion can arise, we shall simply write* $\mathcal{P}$ *instead of* $\mathcal{P}_{CS}$*.*

Definition 6. *Given a constraint system* CS*, consider two problems* $P_1 = \langle C_1, con_1 \rangle$ *and* $P_2 = \langle C_2, con_2 \rangle$ *in* $\mathcal{P}_{CS}$*. We write* $P_1 \sqsubseteq_P P_2$ *iff* $C_1 \sqsubseteq_C C_2$ *and* $con_1 \subseteq con_2$*.*

We now need to define a partially ordered structure that contains all SCSPs that can be generated by enforcing local consistency, starting from a given problem.

Definition 7. *Consider a constraint system* CS *and an SCSP* P *over it. The* up-closure *of* P*, briefly* $P\uparrow$*, is the class of all problems* P' *on* CS *such that* $P \sqsubseteq_P P'$*.*

Proposition 1. *Consider a constraint system* CS*, an SCSP* P *over it and its up-closure* $P\uparrow$*. Then the following statements hold:*

1. *if* $P_1 \sqsubseteq_P P_2$ *and* $P_1 \in P\uparrow$*, then* $P_2 \in P\uparrow$*;*
2. *if* $P_1 \sqsubseteq_P P_2$*, then* $P_2\uparrow \subseteq P_1\uparrow$*.*

The proof of the above proposition is an immediate consequence of the previous definition and of the fact that $\sqsubseteq_C$ is transitive, cf. Theorem 2.

Theorem 3 ($\sqsubseteq_P$ is a po). *Given a constraint system* $CS = \langle S, D, V \rangle$ *and a problem* P *on it, we have:*

- *the relation* $\sqsubseteq_P$ *is a partial order over* $\mathcal{P}_{CS}$*;*
- *in particular* $\langle P\uparrow, \sqsubseteq_{P|P\uparrow} \rangle$ *is a partial ordering, where* $\sqsubseteq_{P|P\uparrow}$ *is the restriction of* $\sqsubseteq_P$ *to* $P\uparrow$*; when no confusion can arise, we simply write* $\langle P\uparrow, \sqsubseteq_P \rangle$*;*
- *the bottom* $\perp_{CS}$ *of* $\langle P\uparrow, \sqsubseteq_P \rangle$ *is* P*.*

Proof. We prove the first claim, the other ones following immediately from the definition of $P\uparrow$ and Proposition 1. As usual, we only prove that the relation is antisymmetric, because transitivity can be proved similarly and reflexivity trivially holds. Hence, suppose that both $P_1 \sqsubseteq_P P_2$ and $P_2 \sqsubseteq_P P_1$ hold. This means that we have the following relations: $con_1 \subseteq con_2$, $C_1 \sqsubseteq_C C_2$, $con_2 \subseteq con_1$, $C_2 \sqsubseteq_C C_1$. From the two previous relations and Theorem 2, it follows that $con_1 = con_2$ and $C_1 = C_2$; hence $P_1 = P_2$. □

3.5 Order-related properties of soft local consistency rules

We recall, from Section 2, that two problems P_1 and P_2, that share the same set of variables, are equivalent if they have the same solution set, and we write $P_1 \equiv_P P_2$.

Now we can list some useful properties of soft local consistency rules, which are related to equivalence and to problem ordering [BMR97]. Here we assume that we are given a constraint system C and a rule r on CS:

– (equivalence) $P \equiv_P r_l^L(P)$ if $\times$ is idempotent.
– (inflationarity) $P \sqsubseteq_P r_l^L(P)$. This means that the new semiring values assigned to tuples by the rule application are always smaller than or equal to the old ones with respect to $\leq_S$.
– (monotonicity) Consider two SCSPs $P_1 = \langle C_1, con_1 \rangle$ and $P_2 = \langle C_2, con_2 \rangle$ over CS. If $P_1 \sqsubseteq_P P_2$, then $r(P_1) \sqsubseteq_P r(P_2)$.

It is easy to prove that all the results about local consistency rules hold also for a whole local consistency algorithm. Moreover, we can prove also that the strategy does not influence the result, if it is fair [BMR97].

4 The Generic Iteration algorithm

In [Apt99b] the Generic Iteration (`GI`) algorithm is introduced to find the least fixpoint of a finite set of functions defined on a partial ordering with bottom. This was then used as an algorithmic schema for classical constraint propagation: each step of constraint propagation was seen as the application of one of these functions. Our idea is to compare this schema with the one used for soft constraints, with the aim of obtaining a new schema which is the most general (that is, it can be applied both to classical and to soft constraints) and has the advantages of both of them.

Given a partial ordering with bottom, say $\langle D, \sqsubseteq, \perp \rangle$, consider now a set of functions $F := \{f_1, \ldots, f_k\}$ on D. The following algorithm can compute the least common fix point of the functions in F.

GENERIC ITERATION ALGORITHM (`GI`)

$d := \perp$;
$G := F$;
while $G \neq \emptyset$ **do**
  choose $g \in G$;
  $G := G - \{g\}$;
  $G := G \cup update(G, g, d)$;
  $d := g(d)$
od

where for all G, g, d the set of functions $update(G, g, d)$ from F is such that:

A. $\{f \in F - G \mid f(d) = d \wedge f(g(d)) \neq g(d)\} \subseteq update(G, g, d)$;
B. $g(d) = d$ implies $update(G, g, d) = \emptyset$;
C. $g(g(d)) \neq g(d)$ implies $g \in update(G, g, d)$.

Assumption **A** states that $update(G, g, d)$ at least contains all the functions from $F - G$ for which d is a fix point but $g(d)$ is not. So at each loop iteration such functions are added to the set G. In turn, assumption **B** states that no functions are added to G in case the value of d did not change. Note that, even though after the assignment $G := G - \{g\}$ we have $g \in F - G$, still $g \notin \{f \in F - G \mid f(d) = d \wedge f(g(d)) \neq g(d)\}$ holds. So assumption **A** does not provide any information when g is to be added back to G. This information is provided in assumption **C**. On the whole, the idea is to keep in G at least all functions f for which the current value of d is not a fix point.

We now recall the results which state the (partial) correctness of the `GI` algorithm, cf. [Apt99b]:

i. Every terminating execution of the `GI` algorithm computes in d a common fixpoint of the functions from F.
ii. Suppose that all functions in F are monotonic. Then every terminating execution of the `GI` algorithm computes in d the least common fixpoint of all the functions from F.
iii. Suppose that all functions in F are inflationary and that D is finite. Then every execution of the `GI` algorithm terminates. □

5 Generalized local consistency for soft constraints via algorithm GI

In this section we will try to combine the two formalisms described so far (soft constraints and the GI algorithm). Our goal is to exploit the `GI` algorithm to perform local consistency over soft constraint problems.

5.1 GI for standard local consistency over soft constraints

The functions that `GI` needs in input are defined on a partial ordering with bottom. In the case of local consistency rules for SCSPs, the partial ordering is $\langle P\uparrow, \sqsubseteq_P\rangle$, and the bottom is the problem P itself, cf. Theorem 3. Moreover, the local consistency rules (and also the more general local consistency functions) have all the "good" properties that GI needs. Namely, those functions are monotonic and inflationary. Thus algorithm GI can be used to perform constraint propagation over soft constraint problems. More precisely, we can see that algorithm GI and the local consistency algorithm schema for soft constraints obtain the same result.

Theorem 4 (GI for soft local consistency rules). *Given an SCSP P over a constraint system CS, consider the SCSP $lc(P, R, T)$ obtained by applying to P a local consistency algorithm using the rules in R and with a fair strategy T. Consider also the partial order $\langle P\uparrow, \sqsubseteq_P\rangle$, and the set of functions R, and apply algorithm GI to such input. Then the output of GI coincides with $lc(P, R, T)$.*

5.2 GI for generalized local consistency over soft constraints

While all local consistency rules are idempotent (since they solve a subproblem), algorithm GI does not need this property. This means that we can define a generalized notion of local consistency rules for soft constraints, by dropping idempotence.

Definition 8 (local consistency functions). *Consider an SCSP problem P over a semiring S. A* local consistency function *for P is a function $f : P\uparrow \rightarrow P\uparrow$ which is monotonic and inflationary over $\sqsubseteq_P$.*

With this definition of a local consistency function we relax two conditions about a local consistency step:

- that it must solve a subproblem;
- that it must be idempotent.

The second generalization has been triggered by the results about the GI algorithm, which have shown that idempotence is not needed for the desired results. Moreover, many practical local consistency algorithms do not exactly solve subproblems, but generate an approximation of the solution (see for example the definition of *bounds consistency* in [MS98] or the notion of partial soft arc-consistency in [BCGR00]). Thus the first extension allows one to model many more practical propagation algorithms.

Theorem 5 (GI for soft local consistency functions). *Given a constraint system CS and an SCSP problem P on it, let us apply the* `GI` *algorithm to the partial order $\langle P\uparrow, \sqsubseteq_P\rangle$ and a finite set R of local consistency functions. Then every terminating execution of the* `GI` *algorithm computes in the output problem P' the least common fixpoint of all the functions from R.*

What is now important to investigate is when the algorithm terminates. This is particularly crucial for soft constraints, since, even when the variable domains are finite, the semiring may contain an infinite number of elements, which is obviously a source of possible non-termination.

6 Termination of the GI algorithm over soft constraints

As noted above, the presence of a possibly infinite semiring may lead to a constraint propagation algorithm which does not terminate. In the following we will give several independent conditions which guarantee termination in some special cases.

The first condition is a predictable extension of the one given in Section 4: instead of requiring the finiteness of the domain of computation, we just require that its chains have finite length, since it is easy to see that constraint propagation moves along a chain in the partial order.

Theorem 6 (termination 1). *Given a constraint system CS and an SCSP problem P on it, let us instantiate the* `GI` *algorithm with the po $\langle P\uparrow, \sqsubseteq_P\rangle$ and a finite set R of local consistency functions. Suppose that the order $\sqsupset_{CS}$ restricted to $P\uparrow$ is well founded. Then every execution of the* `GI` *algorithm terminates.*

This theorem can be used to prove termination in many case. For example, classical constraints over finite domains generate a partial order which is finite (an thus trivially well-founded), so the above theorem guarantees termination. Another example occurs when dealing with weighted soft constraints, where we deal with the naturals. Here the semiring is $\langle N, min, +, 0, +\infty \rangle$. Thus we have an infinite order, but well-founded.

However, there are also many interesting cases in which the ordering $\langle P \uparrow, \sqsubset_P \rangle$ is not well-founded. Consider for instance the case of fuzzy or probabilistic CSPs. For fuzzy CSPs, the semiring is $\langle [0,1], max, min, 0, 1 \rangle$. Thus the partially ordered structure containing all problems that are smaller than the given one, according to the semiring partial order, is not well-founded, since we have all the reals between 0 and a certain element in [0,1]. Thus the above theorem cannot say anything about termination of GI in this case. However, this does not mean that GI does not terminate, but only that the theorem above cannot be applied. In fact, later we will give another sufficient condition which will guarantee termination in the last two cases as well.

In fact, if we restrict our attention to local consistency functions defined via $+$ and $\times$, we can define another condition on our input problem that guarantees the termination of the GI algorithm; this condition exploits the fact that the local consistency functions are defined by means of the two semiring operations, and the properties of such operations.

Definition 9 (semiring closure). *Consider a constraint system CS with semiring $S = \langle A, +, \times, 0, 1 \rangle$, and an SCSP P on CS. Consider also the set of semiring values appearing in P: $Cl(P) = \bigcup_{\langle def', con' \rangle \in C} \{def'(d) \mid d \in D^{|con'|}\}$. Then, a* semiring closure *of P is any set B such that: $Cl(P) \subseteq B \subseteq A$; B is closed with respect to $+$ and $\times$; $<_S$ restricted to B is well founded.*

Theorem 7 (termination 2). *Consider a constraint system CS with semiring $S = \langle A, +, \times, 0, 1 \rangle$, an SCSP P on it and a finite set of local consistency functions R defined via $+$ and $\times$. Assume also there exists a semiring closure of P. Then every execution of the* `GI` *algorithm terminates.*

Proof. The proof is similar to the one in [Apt99b] for the termination theorem; just replace the order $\sqsupset_{CP}$ with $<_S$ and accordingly the set D with B, where B is a semiring closure of P. □

Notice that this theorem is similar to the one in [BMR97] about termination; however in [BMR97] they force the set B to be finite in order to guarantee the termination of a local consistency algorithm (cf. Theorem 4.14 of [BMR97]); a hypothesis that is implied by ours.

If we have a fuzzy constraint problem, then we can take B as the set of all semiring values appearing in the initial problem. In fact, this set is closed with respect to min and max, which are the two semiring operations in this case. Moreover, it is well-founded, since it is finite. Another example is constraint optimization over the reals: if the initial problem contains only natural numbers, then the set B can be the set of all naturals, which is is a well-founded subset of the reals and it is closed w.r.t. $+$ (min) and $\times$ (sum).

Therefore, by using Theorem 7 we can prove that also constraint propagation over fuzzy constraint problems always terminates, provided that each step of the algorithm uses a local consistency function which is defined in terms of the two semiring operations only.

However, it is not always easy to find a semiring closure of a given SCSP P, mainly because we should check that the order restricted to a tentative set B, closed and containing $Cl(P)$, is well-founded. Nevertheless, there is a special case in which we don't have to find such a set, because we can prove that it always exists (which is what Theorem 7 requires). This special case occurs when the multiplicative operation of the semiring is idempotent. In fact, we can prove that in this case there exists always a finite (and thus well-founded) semiring closure of any given problem over that semiring. This obviously is very convenient, since it provides us with an easy way to check whether Theorem 7 can be applied.

Theorem 8 (idempotence of $\times$ and termination). *Consider a constraint system CS, an SCSP P on it and a finite set of local consistency functions R defined via $+$ and $\times$. Assume also that $\times$ is idempotent. Then there exists a finite semiring closure of P, and thus every execution of the GI algorithm terminates.*

Proof. Consider the set $Cl(P)$ of all semiring elements appearing in P. If we combine any subset of them via the $+$ operation, we generate a set of elements, which contains $Cl(P)$, that we denote by $Cl+$. Notice that this set is finite, because the number of subsets of $Cl(P)$ is finite.

Let us now combine any subset of elements of $Cl+$ via the $\times$ operation: in this way we generate a larger set of elements, containing $Cl+$, which we denote by $Cl + \times$. Again, this set is finite. Therefore $Cl + \times$ is a finite semiring closure of P. □

Consider again the fuzzy constraint example. Here $\times$ is min, thus it is idempotent. Therefore, by Theorem 8, GI over such problems always terminates. This is an alternative, and easier, way (to Theorem 7) to guarantee that soft constraint propagation over fuzzy constraints terminates. In fact, we don't have to find a semiring closure of the problem, but just check that the multiplicative operation is idempotent.

Considering all the above results, we can devise the following steps towards proving the termination of algorithm GI on a soft constraint problem P over a semiring S:

- If the local consistency functions are defined via the two operations of S, and the multiplicative operation of S is idempotent, then GI terminates (by Theorems 8).
- If instead $\times$ is not idempotent, but we still have local consistency functions defined via the two semiring operations, we can try to find a semiring closure of P. If we find it, then GI terminates (by Theorem 7).
- If we cannot find a semiring closure of P, or the local consistency functions are more general, then we can try to prove that the partial order of problems is well-founded. If it is so, the GI terminates (by Theorem 6).

While Theorem 8 applies in a special case of the hypothesis of Theorem 7, it is interesting to investigate the relationship between the hypothesis of Theorem 6 and 7. What can be proved is that these two conditions, namely, the well-foundedness of the partial order of problems and the existence of a semiring closure, are independent. In other words, there are cases in which one holds and not the other one, and vice versa. To prove this result, we need the following definition.

Definition 10. *Let $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ be a semiring and B a subset of A. The set B is a* down-set *(or an* order ideal*) if, whenever $a \in B$, $a' \in A$ and $a' \leq_S a$, then $a' \in B$. Given any subset B of A, the downward closure of B is*

$$B \downarrow := \{d' \in A \,:\, \exists d\,(d \in B \text{ and } d' \leq_S d)\}.$$

Observe that the class $\mathcal{F}$ of down-sets containing a subset B of A is not empty, since A itself is such a set. Moreover, it is easy to check that the downward closure of B is the smallest down-set of $\mathcal{F}$; hence the downward closure of a set is well defined.

Given a semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$, the following result links the upward closure of a problem P with the downward closure of $Cl(P)$, thus allowing us to compare the conditions in Theorem 6 and 7.

Proposition 2. *Given a constraint system CS and a problem P defined on it, consider the set $B := \{def(t) \in A \,:\, \exists P' \in P \uparrow\ (c := \langle def, con \rangle \in P',\ t \in D^{|con|})\}$. Then $B = Cl(P) \downarrow$.*

Proof. It follows immediately from the definition of $\sqsubseteq_P$, of $P \uparrow$ and $Cl(P) \downarrow$. □

Now we can notice that, given a subset B of a semiring, if $<_S$ restricted to B is well founded, then so is $<_S$ restricted to $\hat{B} := B \cup \{\mathbf{0}, \mathbf{1}\}$. Furthermore, if B is finite, so is $\hat{B}$. In fact it is sufficient to check that the following identities hold because of the fact that $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ is a c-semiring: if $a \in B$ then $a + 0 = a \in B$; if $a \in B$ then $a \times 1 = a \in B$; if $a \in B$ then $a + 1 = 1 \in \hat{B}$; if $a \in B$ then $a \times 0 = 0 \in \hat{B}$. Hence, in Theorem 7, we can replace the hypothesis "$Cl(P) \subseteq B$ and B a semiring closure of $Cl(P)$" with the condition that "Cl(P) is a subset of a well founded sub-c-semiring B of $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$".

Moreover, $+$ is the least upper bound operation and, if $\times$ is idempotent, $\times$ is the greatest lower bound operation, cf. Theorem 2.9 and 2.10 of [BMR97]. Hence a sub-c-semiring is also a sub-lattice of $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ and vice versa if $\times$ is idempotent. Thus a subset B of a semiring can be a down-set and yet it may be not closed with respect to $\times$ and $+$. For instance, the set of negative real numbers augmented with $-\infty$ is a down-set in the lattice $\boldsymbol{R}^{\infty}$ of reals extended with $\{+\infty, -\infty\}$ and the usual linear ordering; however it is not a sub-lattice itself. Vice versa, there are sub-lattices of $\boldsymbol{R}^{\infty}$ - hence sets that are closed with respect to the least upper bound and the greatest lower bound operations - that are not down-sets. For instance, the extended interval $[0, 1] \cup \{+\infty, -\infty\}$. Therefore the two conditions that guarantee the termination of algorithm `GI` in Theorems 6 and 7 are independent.

7 Conclusions

The results of this paper show that indeed it is possible to treat in a uniform way hard and soft constraints. This was already shown in [BMR97], but restricted to a notion of local consistency where each step solves a subproblem. Here instead we have extended this result also to more general propagation algorithms, which approximate the classical notion in order to be more efficient. Moreover, we have also studied the conditions under which constraint propagation terminates. Some results were already present in [BMR97] and [Apt99b], but here we have generalized and extended them, providing more general, and also more easily checkable, sufficient conditions.

References

[Apt99a] K.R. Apt, The Essence of Constraint Propagation, *Theoretical Computer Science*, 221(1-2), pp. 179-210, 1999.

[Apt99b] K.R. Apt, The Rough Guide to Constraint Propagation, *Proc. of the 5th International Conference on Principles and Practice of Constraint Programming (CP'99)*, (invited lecture), Springer-Verlag Lecture Notes in Computer Science 1713, pp. 1-23.

[B94] C. Bessière. Arc-consistency and Arc-consistency again. *Artificial Intelligence*, 65(1), 1994.

[BCGR00] S. Bistarelli, P. Codognet, Y. Georget and F. Rossi Labeling and Partial Local Consistency for Soft Constraint Programming *Proc. of the 2nd International Workshop on Practical Aspects of Declarative Languages (PADL'00)*, Springer-Verlag Lecture Notes in Computer Science 1753, 2000.

[BMR95] S. Bistarelli, U. Montanari and F. Rossi. Constraint Solving over Semirings. *Proceedings of IJCAI'95*, Morgan Kaufman, 1995.

[BMR97] S. Bistarelli, U. Montanari and F. Rossi. Semiring-based Constraint Solving and Optimization. *Journal of ACM*, vol. 44, no. 2, March 1997.

[DFP93] D. Dubois, H. Fargier and H. Prade. The calculus of fuzzy restrictions as a basis for flexible constraint satisfaction. *Proc. IEEE International Conference on Fuzzy Systems*, IEEE, pp. 1131–1136, 1993.

[FL93] H. Fargier and J. Lang Uncertainty in Constraint Satisfaction Problems: a Probabilistic Approach *Proc. European Conference on Symbolic and Qualitative Approaches to Reasoning and Uncertainty (ECSQARU)*, Springer-Verlag, LNCS 747, pp. 97–104, 1993.

[FW92] E. C. Freuder and R. J. Wallace. Partial Constraint Satisfaction. *AI Journal*, 1992, 58.

[AC5] P. Van Hentenryck, Y. Deville and C-M. Teng. A generic arc-consistency algorithm and its specializations. *Artificial Intelligence 57 (1992)*, pp 291-321.

[MS98] K. Marriott and P. Stuckey. Programming with Constraints. MIT Press, 1998.

[SFV95] T. Schiex and H. Fargier and G. Verfaille. Valued Constraint Satisfaction Problems: Hard and Easy Problems. *Proc. IJCAI95*, Morgan Kaufmann, pp. 631–637, 1995.