Abstracting Soft Constraint

S. Bistarelli[1], P. Codognet[2], Y. Georget[3], F. Rossi[4]

1: Università di Pisa, Dipartimento di Informatica, Corso Italia 40, 56125 Pisa, Italy. E-mail: bista@di.unipi.it
2: University of Paris 6, LIP6, case 169, 4, Place Jussieu, 75 252 Paris Cedex 05, France. E-mail: Philippe.Codognet@lip6.fr
3: INRIA Rocquencourt, BP 105, 78153 Le Chesnay, France. E-mail: Yan.Georget@inria.fr
4: Università di Padova, Dipartimento di Matematica, Via Belzoni 7, 35131 Padova, Italy. E-mail: frossi@math.unipd.it

Abstract. We propose an abstraction scheme for soft constraint problems and we study its main properties. Processing the abstracted version of a soft constraint problem can help us in many ways: for example, to find good approximations of the optimal solutions, or also to provide us with information that can make the subsequent search for the best solution easier. The results of this paper show that the proposed scheme is promising; thus they can be used as a stable formal base for any experimental work specific to a particular class of soft constraint problems.

1 Introduction

Classical constraint satisfaction problems (CSPs) [15] are a very convenient and expressive formalism for many real-life problems, like scheduling, resource allocation, vehicle routing, timetabling, and many others [20]. However, many of these problems are often more faithfully represented as *soft constraint satisfaction problems* (SCSPs), which are just like classical CSPs except that each assignment of values to variables in the constraints is associated to an element taken from a partially ordered set. These elements can then be interpreted as levels of preference, or costs, or levels of certainty, or many other criteria.

There are many formalizations of soft constraint problems. In this paper we consider the one based on semirings [4, 3], where the semiring specifies the partially ordered set and the appropriate operation to use to combine constraints together.

Although it is obvious that SCSPs are much more expressive than classical CSPs, they are also more difficult to process and to solve. Therefore, sometimes it may too costly to find all, or even only one, optimal solution. Also, although formally classical propagation techniques like arc-consistency [14] can be extended to be applied also to SCSPs [4], even such techniques can be too costly to be used, depending on the size and structure of the partial order associated to the SCSP.

For these reasons, it may be reasonable to 1) process an abstracted version of a SCSP, which should be easier to handle, 2) derive some useful information

on the abstract problem, 3) bring back to the original problem some (or possibly all) of this information, and then 4) continue the solution process on the transformed problem. All this with the aim of finding an optimal solution, or an approximation of it, for the original SCSP, within the resource bounds we have.

In this paper we propose an abstraction scheme for SCSPs which follows the classical scheme described in [5] and adapts it to soft constraints: given an SCSP (the *concrete* one), we get an abstract SCSP by just changing the associated semiring, and relating the two structures (the concrete and the abstract one) via a *Galois insertion*. Then, a deep study of the relationship between the concrete SCSP and the corresponding abstract one allows us to prove some results which can help in deriving useful information on the abstract problem and then take some of the derived information back to the concrete problem.

In particular, we can prove the following:

- All optimal solutions of the concrete SCSP are also optimal in the corresponding abstract SCSP (see Theorem 2). Thus, in order to find an optimal solution of the concrete problem, we could find all the optimal solutions of the abstract problem, and then just check their optimality on the concrete SCSP.
- Given any optimal solution of the abstract problem, we can easily find upper and lower bounds for an optimal solution for the concrete problem (see Theorem 3). If we are satisfied with these bounds, we could just take the optimal solution for the abstract problem as a reasonable approximation of an optimal solution for the concrete problem.
- If we apply some constraint propagation technique over the abstract problem, say P, obtaining a new abstract problem, say P', some of the information in P' can be inserted into P, obtaining a new concrete problem which is closer to its solution and thus easier to solve (see Theorem 4 and 6): for example, a branch-and-bound algorithm would explore a smaller (or equal) search tree before finding an optimal solution.

The only other abstraction scheme for soft constraint problems we are aware of is the one in [7], where valued CSPs [18] are abstracted in order to produce good lower bounds for the optimal solutions. The concept of valued CSPs is similar to our notion of SCSPs. In fact, one can pass from one formalization to the other one without changing the solutions, provided that the partial order is total [2]. However, our abstraction scheme is different from the one in [7]. In fact, we are not only interested in finding good lower bounds for the optimum, but also in finding the exact optimal solutions in a shorter time. Moreover, we don't define *ad hoc* abstraction functions but we follow the classical abstraction scheme devised in [5], with Galois insertions to relate the concrete and the abstract domain, and locally correct functions on the abstract side. We think that this is important in that it allows to inherit many properties which have already been proved for the classical case.

When we abstract an SCSP, we only change its semiring, but not its structure (that is, the topology of its graph), nor its variable domains. In the future, we

will also investigate these kinds of abstractions and their possible combination. Some of these were already proposed in the literature [9, 11, 19], but only for classical constraints.

We are aware that this paper is just a first step towards the use of abstraction for helping to find the solution of a soft constraint problem in a shorter time. More properties can probably be investigated and proved, and also an experimental phase is necessary to check the real practical value of our proposal. We plan to perform such a phase within the `clp(fd,S)` system developed at INRIA [13], which can already solve soft constraints in the classical way (branch-and-bound plus propagation via partial arc-consistency).

For reasons of space, this version of the paper does not contain the proofs of the theorems, although some informal discussion and motivations are given in the text of the paper.

2 Background

In this section we recall the main notions about *soft constraints* [4] and *abstract interpretation* [5], that will be useful for the developments and results of this paper.

2.1 Soft constraints

In the literature there are many formalizations of the concept of *soft constraints* [18, 8, 12, 10]. Here we refer to the one described in [4, 3], which however can be shown to generalize and express many of the others [4, 2]. In a few words, a soft constraint is just a classical constraint where each instantiation of its variables has an associated value from a partially ordered set. 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 this formalization is based on the concept of semiring, which is just a set plus two operations.

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. 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.

Let us consider the relation $\leq_S$ over A such that $a \leq_S b$ iff $a + b = b$. Then it is possible to prove that: $\leq_S$ is a partial order; $+$ and $\times$ are monotone on $\leq_S$; $\mathbf{0}$ is its minimum and $\mathbf{1}$ its maximum; $\langle A, \leq_S\rangle$ is a complete lattice and $+$ is its lub. Moreover, if $\times$ is idempotent, then $\langle A, \leq_S\rangle$ is a complete distributive lattice and $\times$ is its glb. Informally, the relation $\leq$ gives us a way to compare (some of the) tuples of values and constraints.

Given a c-semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$, a finite set D (the domain of the variables), and an ordered set of variables V, a *constraint* is a pair $\langle def, con\rangle$ where $con \subseteq V$ and $def : D^{|con|} \to A$. Therefore, a constraint specifies a set

of variables (the ones in con), and assigns to each tuple of values of D of these variables an element of the semiring set A. This element can then be interpreted in several ways: as a level of preference, or as a cost, or as a probability, etc., depending on the choice of the semiring operations.

Constraints can be compared by looking at the semiring values associated to the same tuples: Consider two constraints $c_1 = \langle def_1, con \rangle$ and $c_2 = \langle def_2, con \rangle$, with $|con| = k$. Then $c_1 \sqsubseteq_S c_2$ if for all k-tuples t, $def_1(t) \leq_S def_2(t)$. The relation $\sqsubseteq_S$ is a partial order.

A *soft constraint satisfaction problem* (SCSP) is a pair $\langle C, con \rangle$ where $con \subseteq V$ and C is a set of constraints. Note that a classical CSP is a SCSP where the chosen c-semiring is: $S_{CSP} = \langle \{false, true\}, \vee, \wedge, false, true \rangle$. Fuzzy CSPs [8, 16, 17] can instead be modeled in the SCSP framework by choosing the c-semiring: $S_{FCSP} = \langle [0, 1], max, min, 0, 1 \rangle$. Figure 1 shows a fuzzy CSP. Variables are inside circles, constraints are represented by undirected arcs, and semiring values are written to the right of the corresponding tuples. Here we assume that the domain D of the variables contains only elements a and b.

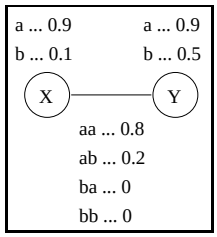

Fig. 1. A fuzzy CSP.

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 $def(t) = def_1(t \downarrow_{con_1}^{con}) \times def(t \downarrow_{con_2}^{con})$. In words, combining two constraints means building a new constraint involving all the variables of the original ones, and which associates to each tuple of domain values for such variables a semiring element which is obtained by multiplying the elements associated by the original constraints to the appropriate subtuples.

Using the properties of $\times$ and $+$, it is easy to prove that: $\otimes$ is associative and commutative, and monotone over $\sqsubseteq_C$. Moreover, if $\times$ is idempotent, $\otimes$ is idempotent as well.

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

The *solution* of a SCSP problem $P = \langle C, con \rangle$ is the constraint $Sol(P) = (\bigotimes C) \Downarrow_{con}$. That is, we combine all constraints, and then project over the variables in *con*. In this way we get the constraint over *con* which is "induced" by the entire SCSP.

For example, each solution of the fuzzy CSP of Figure 1 consists of a pair of domain values (that is, a domain value for each of the two variables) and an associated semiring element (here we assume that *con* contains all variables). Such an element is obtained by looking at the smallest value for all the subtuples (as many as the constraints) forming the pair. For example, for tuple $\langle a, a \rangle$ (that is, $x = y = a$), we have to compute the minimum between 0.9 (which is the value for $x = a$), 0.8 (which is the value for $\langle x = a, y = a \rangle$) and 0.9 (which is the value for $y = a$). Hence, the resulting value for this tuple is 0.8.

Given an SCSP problem P, consider $Sol(P) = \langle def, con \rangle$. An *optimal solution* of P is a pair $\langle t, v \rangle$ such that $def(t) = v$, and there is no t' such that $v <_S def(t')$. Therefore optimal solutions are solutions which have the best semiring element associated to any solution. The set of optimal solutions of an SCSP P will be written as $Opt(P)$.

SCSP problems can be solved by extending and adapting the technique usually used for classical CSPs. For example, to find the best solution we could employ a branch-and-bound search algorithm (instead of the classical backtracking), and also the successfully used propagation techniques, like arc-consistency [14], can be generalized to be used for SCSPs.

The formal definition of propagation algorithms (sometimes called also *local consistency* algorithms) for SCSPs can be found in [4]. For the purpose of this paper, we just need to say that the generalization from classical CSPs concerns the fact that, instead of deleting values or tuples, obtaining local consistency in SCSPs means changing the semiring values associated to some tuples or domain elements. The change always brings these values towards the worst value of the semiring, that is, the $\mathbf{0}$. Thus, it is obvious that, given an SCSP problem P and the problem P' obtained by applying some local consistency algorithm to P, we must have $P' \sqsubseteq_S P$. Another important property of such algorithms is that $Sol(P) = Sol(P')$.

2.2 Abstraction

Abstract interpretation [1, 5, 6] is a theory developed to reason about the relation between two different semantics (the *concrete* and the *abstract* semantics). The idea of approximating program properties by evaluating a program on a simpler domain of descriptions of "concrete" program states goes back to the early 70's. The inspiration was that of approximating properties from the exact (concrete) semantics into an approximate (abstract) semantics, that explicitly exhibits a structure (e.g., ordering) which is somehow present in the richer concrete structure associated to program execution.

The guiding idea is to relate the concrete and the abstract interpretation of the calculus by a pair of functions, the *abstraction* function α and the *concretization* function γ, which form a Galois connection.

Let $(\mathcal{C}, \sqsubseteq)$ (concrete domain) be the domain of the concrete semantics, while $(\mathcal{A}, \leq)$ (abstract domain) be the domain of the abstract semantics. The partial order relations reflect an approximation relation. Since in approximation theory a partial order specifies the precision degree of any element in a poset, it is obvious to assume that if α is a mapping associating an abstract object in $(\mathcal{A}, \leq)$ for any concrete element in $(\mathcal{C} \sqsubseteq)$, then the following holds: if $\alpha(x) \leq y$, then y is also a correct, although less precise, abstract approximation of x. The same argument holds if $x \sqsubseteq \gamma(y)$. Then y is also a correct approximation of x, although x provides more accurate information than $\gamma(y)$. This gives rise to the following formal definition.

Definition 1 (Galois Insertion). *Let $(\mathcal{C}, \sqsubseteq)$ and $(\mathcal{A}, \leq)$ be two posets (the concrete and the abstract domain). A Galois Connection $\langle \alpha, \gamma \rangle : (C, \sqsubseteq) \rightleftharpoons (A, \leq)$ is a pair of maps $\alpha : \mathcal{C} \to \mathcal{A}$ and $\gamma : \mathcal{A} \to \mathcal{C}$ such that*

1. *α and γ are monotonic,*
2. *for each $x \in \mathcal{C}, x \sqsubseteq (\gamma \circ \alpha)(x)$ and*
3. *for each $y \in \mathcal{A}, (\alpha \circ \gamma)(y) \leq y$.*

Moreover, a Galois Insertion *(of $\mathcal{A}$ in $\mathcal{C}$) $\langle \alpha, \gamma \rangle : (\mathcal{C}, \sqsubseteq) \rightleftharpoons (\mathcal{A}, \leq)$ is a Galois connection where $\alpha \circ \gamma = Id_{\mathcal{A}}$.*

The following basic properties are satisfied by any Galois insertion:

1. γ is injective and α is surjective.
2. $\gamma \circ \alpha$ is an upper closure operator in $(\mathcal{C}, \sqsubseteq)$.
3. α is additive and γ is co-additive.
4. Upper and lower adjoints uniquely determine each other. Namely,
$$\gamma = \lambda y. \bigsqcup_{\mathcal{C}} \{x \in \mathcal{C} \mid \alpha(x) \sqsubseteq y\}, \alpha = \lambda x : \bigsqcap_{\mathcal{A}} \{y \in \mathcal{A} \mid x \leq \gamma(y)$$
5. α is an isomorphism from $(\gamma\alpha)(\mathcal{C})$ to $\mathcal{A}$, having γ as its inverse.

An example of a Galois insertion can be seen in Figure 2. Here, the concrete lattice is $\langle [0,1], \leq \rangle$, and the abstract one is $\langle \{0,1\}, \leq \rangle$. Function α maps all real numbers in $[0, 0.5]$ into 0, and all other integers (in $(0.5, 1]$) into 1. Function γ maps 0 into 0.5 and 1 into 1.

Given a concrete semantics and a Galois insertion between the concrete and the abstract domains, we want to define an abstract semantics. The theory requires the concrete semantics to be the least fixpoint of a semantic function $F : \mathcal{C} \to \mathcal{C}$. The abstract semantic function $\tilde{F} : \mathcal{A} \to \mathcal{A}$ is *correct* if $\forall x \in \mathcal{C}. F(x) \sqsubseteq \gamma(\tilde{F}(\alpha(x)))$.

It is often the case that F is in turn defined as composition of "primitive" operators. Let $f : \mathcal{C}^n \to \mathcal{C}$ be one such an operator and assume that $\tilde{f}$ is its abstract counterpart. Then $\tilde{f}$ is (*locally*) *correct w.r.t. f* if $\forall x_1, \ldots, x_n \in \mathcal{C}. f(x_1, \ldots, x_n) \sqsubseteq \gamma(\tilde{f}(\alpha(x_1), \ldots, \alpha(x_n)))$. The local correctness of all the primitive operators implies the global correctness. Hence, we can define an abstract semantics by defining locally correct abstract primitive semantic functions. An abstract computation is then related to the concrete computation, simply by replacing the concrete operators by the corresponding abstract operators.

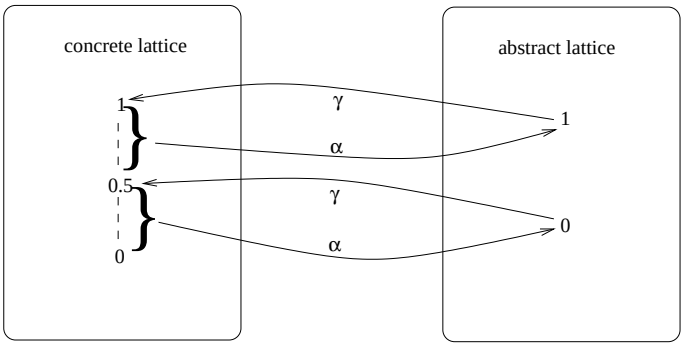

Fig. 2. A Galois insertion.

3 Abstracting soft CSPs

Given the notions of soft constraints and abstraction, recalled in the previous section, we now want to show how to abstract soft constraint problems. The main idea is very simple: we just want to pass, via the abstraction, from an SCSP P over a certain semiring S to another SCSP $\tilde{P}$ over the semiring $\tilde{S}$, where the lattices associated to $\tilde{S}$ and S are related by a Galois insertion as shown above.

More in detail, consider the *concrete* SCSP problem $P = \langle C, con \rangle$ over semiring S, where

- $S = \langle A, +, \times, 0, 1 \rangle$ and
- $C = \{c_0, \ldots, c_n\}$ with $c_i = \langle con_i, def_i \rangle$ and $def_i : D^{|con_i|} \to A$;

we define the *abstract* SCSP problem $\tilde{P} = \langle \tilde{C}, con \rangle$ over the semiring $\tilde{S}$, where

- $\tilde{S} = \langle \tilde{A}, \tilde{+}, \tilde{\times}, \tilde{0}, \tilde{1} \rangle$;
- $\tilde{C} = \{\tilde{c_0}, \ldots, \tilde{c_n}\}$ with $\tilde{c_i} = \langle con_i, \tilde{def_i} \rangle$ and $\tilde{def_i} : D^{|con_i|} \to \tilde{A}$;
- if $L = \langle A, \leq \rangle$ is the lattice associated to S and $\tilde{L} = \langle \tilde{A}, \tilde{\leq} \rangle$ the lattice associated to $\tilde{S}$, then there is a Galois insertion $\langle \alpha, \gamma \rangle$ such that $\alpha : L \to \tilde{L}$.

Notice that the kind of abstraction we consider in this paper does not change the structure of the SCSP problem. That is, C and $\tilde{C}$ have the same number of constraints, and c_i and $\tilde{c_i}$ involve the same variables. The only thing that is changed by abstracting an SCSP is the semiring. Thus P and $\tilde{P}$ have the same graph topology (variables and constraints), but different constraint definitions, since if a certain tuple of domain values in a constraint of P has semiring value a, then the same tuple in the same constraint of $\tilde{P}$ has semiring value $\alpha(a)$. Notice also that α and γ can be defined in several different ways, but all of them have to satisfy the properties of the Galois insertion, from which it derives, among others, that $\alpha(0) = \tilde{0}$ and $\alpha(1) = \tilde{1}$.

As an example, consider any SCSP over the semiring for optimization $S_{WCSP} = \langle \mathcal{R}^+ \cup \{+\infty\}, \min, +, +\infty, 0 \rangle$, and suppose we want to abstract it onto the semiring for fuzzy reasoning $S_{FCSP} = \langle [0,1], max, min, 0, 1 \rangle$. In other words, instead

of computing the minimum of the sum of all costs, we just want to compute the minimum of the maximum of all costs, and we want to normalise the costs over [0..1]. As noted above, the mapping $\alpha : \langle \mathcal{R}^+, \leq_{WCSP} \rangle \rightarrow \langle [0,1], \leq_{FCSP} \rangle$ can be defined in different ways. For example one can decide to map all the reals above some fixed real x onto 1 and then to divide the reals in $[0, x]$ in several intervals and map each of these intervals onto one element of $[0, 1)$.

Another example is the abstraction from the fuzzy semiring to the classical one: $S_{CSP} = \langle \{0,1\}, \vee, \wedge, 0, 1 \rangle$. Here function α maps each element of $[0,1]$ into either 0 or 1. For example, one could map all the elements in $[0, x]$ onto 0, and all those in $(x, 1]$ onto 1, for some fixed x. Figure 2 represents this example with $x = 0.5$.

We have defined Galois insertions between two lattices $\langle L, \leq_S \rangle$ and $\langle \tilde{L}, \tilde{\leq}_{\tilde{S}} \rangle$ of semiring values. However, for convenience, in the following we will often use Galois insertions between lattices of problems $\langle PL, \sqsubseteq_S \rangle$ and $\langle \tilde{PL}, \tilde{\sqsubseteq}_{\tilde{S}} \rangle$ where PL contains problems over the concrete semiring and $\tilde{PL}$ over the abstract semiring. This does not change the meaning of our abstraction, we are just upgrading the Galois insertion from semiring values to problems. Thus, when we will say that $\tilde{P} = \alpha(P)$, it will mean that $\tilde{P}$ is obtained by P via the application of α to all the semiring values appearing in P.

4 Properties and advantages of the abstraction

Let us consider the scheme depicted in Figure 3. Here and in the following pictures, the left box contains the lattice of concrete problems, and the right one the lattice of abstract problems. The partial order in each of these lattices is shown via dashed lines. Connections between the two lattices, via the abstraction and concretization functions, is shown via directed arrows. In the following, we will call $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ the concrete semiring and $\tilde{S} = \langle \tilde{A}, \tilde{+}, \tilde{\times}, \tilde{\mathbf{0}}, \tilde{\mathbf{1}} \rangle$ the abstract one. Thus we will always consider a Galois insertion $\langle \alpha, \gamma \rangle : \langle A, \leq_S \rangle \rightleftharpoons \langle \tilde{A}, \leq_{\tilde{S}} \rangle$.

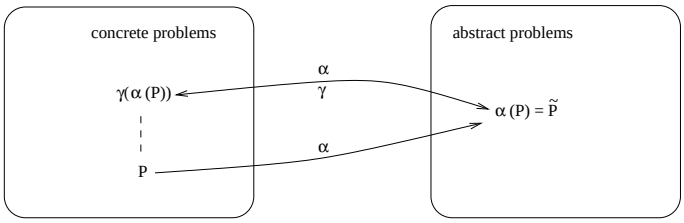

Fig. 3. The concrete and the abstract problem.

In Figure 3, P is the starting SCSP problem. Then with the mapping α we get $\tilde{P} = \alpha(P)$, which is an abstraction of P. By applying the mapping γ to $\tilde{P}$,

we get the problem $\gamma(\alpha(P))$. Let us first notice that these two problems (P and $\gamma(\alpha(P))$) are related by a precise property, as stated by the following theorem.

Theorem 1. *Given an SCSP problem P over S, we have that $P \sqsubseteq_S \gamma(\alpha(P))$.*

We recall that this means that, for each tuple in each constraint, the semiring value associated to such a tuple in P is smaller (w.r.t. $\leq_S$) than the corresponding value associated to the same tuple in $\gamma(\alpha(P))$. This comes from the properties of the Galois insertion $\langle \alpha, \gamma \rangle$, in particular from the fact that, for each x, $x \leq_S \gamma(\alpha(x))$. Notice that this implies that, if a tuple in $\gamma(\alpha(P))$ has semiring value $\mathbf{0}$, then it must have value $\mathbf{0}$ also in P. This holds also for the solutions, whose semiring value is obtained by combining the semiring values of several tuples.

Corollary 1. *Given an SCSP problem P over S, we have that $Sol(P) \sqsubseteq_S Sol(\gamma(\alpha(P))$.*

We recall that $Sol(P)$ is just a constraint, which is obtained as $\bigotimes(C) \Downarrow_{con}$. Thus the statement of this corollary comes just from the monotonicity of $\times$ and $+$.

A consequence of this result is that, by passing from P to $\gamma(\alpha(P))$, we may loose some inconsistency but we will not introduce new ones. That is, if a solution of $\gamma(\alpha(P))$ has value $\mathbf{0}$, then this was true also in P.

There is also an interesting relationship between the set of optimal solutions of P and that of $\alpha(P)$. In fact, if a certain tuple is optimal in P, then this same tuple is also optimal in $\alpha(P)$. This again comes from the properties of a Galois insertion, in particular from the monotonicity of α.

Theorem 2. *Given an SCSP problem P over S, we have that $Opt(P) \subseteq Opt(\alpha(P))$.*

Therefore, the set of optimal solutions of the abstract problem contains all the optimal solutions of the concrete problem. Thus, if we want to find an optimal solution of the concrete problem, we could find all the optimal solutions of the abstract problem, and then use them on the concrete side to find an optimal solution for the concrete problem. Assuming that working on the abstract side is easier than on the concrete side, this method could help us find an optimal solution of the concrete problem by looking at just a subset of tuples in the concrete problem.

Another point to notice is that any optimal solution, say t, of the abstract problem, say with value $\tilde{v}$, has a value among $v_1, \ldots, v_n$ in P, where $\alpha(v_1) = \ldots = \alpha(v_n) = \tilde{v}$. Now, because of properties of the Galois insertion, there exists an optimal solution of P with value among the v_i's. Thus, if we think that approximating the optimal value with a value within the bounds of $v_1, \ldots, v_n$ is satisfactory, we can take t as an approximation of an optimal solution of P. More precisely, we have that $\gamma(\tilde{v})$ is an upper bound of the value of an optimal solution of P, and the value v_i given by P to t is a lower bound.

Theorem 3. *Given an SCSP problem P over S, consider an optimal solution of $\alpha(P)$, say t, with semiring value $\tilde{v}$ in $\alpha(P)$ and v in P. Consider also the set V of all concrete semiring values such that $\alpha(v_i) = \tilde{v}$ for all $v_i \in V$. Then there exists an optimal solution $\bar{t}$ of P, say with value $\bar{v}$, such that:*

- $\bar{v} \in V$;
- $v \leq \bar{v} \leq \gamma(\tilde{v})$.

Consider now what we can do on the abstract problem, $\alpha(P)$. One possibility is to apply an abstract function $\tilde{f}$, which can be, for example, a local consistency algorithm or also a solution algorithm. In the following, we will consider functions $\tilde{f}$ which are always intensive, that is, which bring the given problem closer to the bottom of the lattice. In fact, our goal is to solve an SCSP, thus going higher in the lattice does not help in this task, since solving means combining constraints and thus getting lower in the lattice. Also, the functions $\tilde{f}$ we will consider will always be locally correct with respect to any function f_{sol} which solves the concrete problem. We will call such a property *solution-correctness*. More precisely, given a problem P with constraint set C, $f_{sol}(P)$ is a new problem P' with the same topology as P whose tuples have semiring values possibly lower. Let us call C' the set of constraints of P'. Then, for any constraint $c' \in C'$, $c' = (\bigotimes C) \Downarrow_{var(c')}$. In other words, f_{sol} combines all constraints of P and then projects the resulting global constraint over each of the original constraints.

Definition 2. *Given an SCSP problem P over S, consider a function $\tilde{f}$ on $\alpha(P)$. Then $\tilde{f}$ is solution-correct if, given any f_{sol} which solves P, $\tilde{f}$ is locally correct w.r.t. f_{sol}.*

We will also need the notion of safeness of a function, which just means that it maintains all the solutions.

Definition 3. *Given an SCSP problem P and a function $f : PL \to PL$, f is safe if $Sol(P) = Sol(f(P))$.*

From $\tilde{f}(\alpha(P))$, applying the concretization function γ, we get $\gamma(\tilde{f}(\alpha(P)))$, which therefore is again over the concrete semiring (the same as P). The following property says that, under certain conditions, P and $P \otimes \gamma(\tilde{f}(\alpha(P)))$ are equivalent. Figure 4 describes such a situation. In this figure, we can see that several partial order lines have been drawn. On the abstract side, function $\tilde{f}$ takes any element closer to the bottom, because of its intensiveness. On the concrete side, $P \otimes \gamma(\tilde{f}(\alpha(P)))$ is smaller than both P and $\gamma(\tilde{f}(\alpha(P)))$ because of the properties of $\otimes$. Also, $\gamma(\tilde{f}(\alpha(P)))$ is smaller than $\gamma(\alpha(P))$ because of the monotonicity of γ, $\gamma(\tilde{f}(\alpha(P)))$ is higher than $f_{sol}(P)$ because of the solution-correctness of $\tilde{f}$, and $f_{sol}(P)$ is smaller than P because of the way $f_{sol}(P)$ is constructed. Finally, if $\times$ is idempotent, then it coincides with the glb, thus we have that $P \otimes \gamma(\tilde{f}(\alpha(P)))$ is higher than $f_{sol}(P)$, because by definition the glb is the higher among all the lower bounds of P and $\gamma(\tilde{f}(\alpha(P)))$.

Theorem 4. *Given an SCSP problem P over S, consider a function $\tilde{f}$ on $\alpha(P)$ which is safe, solution-correct, and intensive. Then, if $\times$ is idempotent, $Sol(P) = Sol(P \otimes \gamma(\tilde{f}(\alpha(P))))$.*

This theorem does not say anything about the power of $\tilde{f}$, which could make many modifications to $\alpha(P)$ but could also not modify anything. In this last

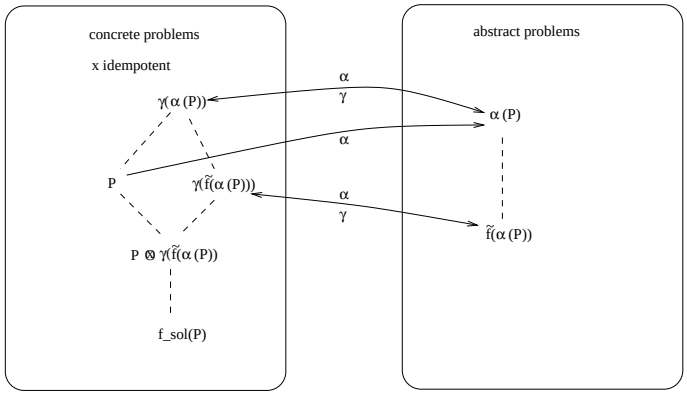

Fig. 4. The general abstraction scheme, with $\times$ idempotent.

case, $\gamma(\tilde{f}(\alpha(P))) = \gamma(\alpha(P)) \sqsupseteq P$ (see Figure 5), so $P \otimes \gamma(\tilde{f}(\alpha(P))) = P$, which means that we have not gained anything in abstracting P. However, we can always use the relationship between P and $\alpha(P)$ (see Theorem 2 and 3) to find an approximation of the optimal solutions and of the inconsistencies of P.

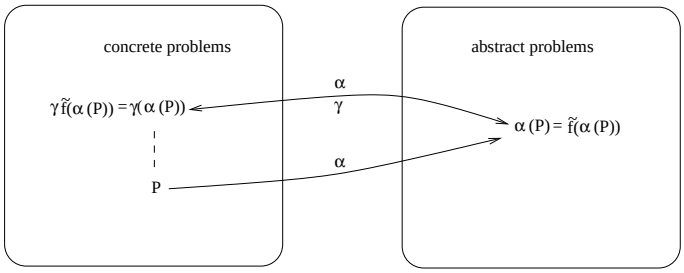

Fig. 5. The scheme when $\tilde{f}$ does not modify anything.

If instead $\tilde{f}$ modifies $\alpha(P)$, then if the order of the concrete semiring is total, we have that $P \otimes \gamma(\tilde{f}(\alpha(P))) = \gamma(\tilde{f}(\alpha(P)))$ (see Figure 6), and thus we can work on $\gamma(\tilde{f}(\alpha(P)))$ to find the solutions of P. In fact, $\gamma(\tilde{f}(\alpha(P)))$ is lower than P and thus closer to the solution.

Theorem 5. *Given an SCSP problem P over S, consider a function $\tilde{f}$ on $\alpha(P)$ which is safe, solution-correct, and intensive. Then, if $\times$ is idempotent, $\tilde{f}$ modifies $\alpha(P)$, and the order of the concrete semiring is total, we have that $P \sqsupseteq_S \gamma(\tilde{f}(\alpha(P))) \sqsupseteq_S f_{sol}(P)$.*

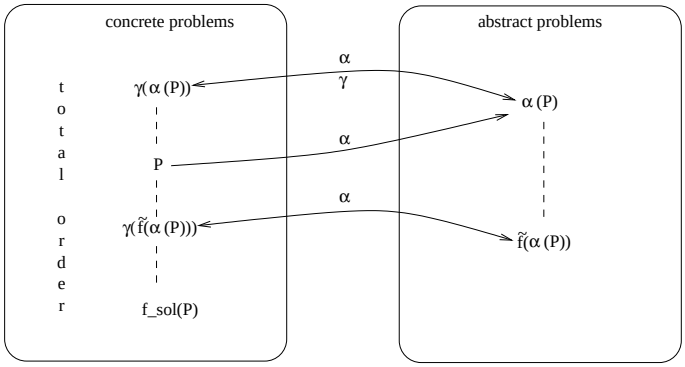

Fig. 6. The scheme when the concrete semiring has a total order.

Notice that we need the idempotence of the $\times$ operator for Theorem 4 and 5. If instead $\times$ is not idempotent, then we can prove something weaker. Figure 7 shows this situation. With respect to Figure 4, we can see that the possible non-idempotency of $\times$ changes the partial order relationship on the concrete side. In particular, we don't have the problem $P \otimes \gamma(\tilde{f}(\alpha(P)))$ any more, nor the problem $f_{sol}(P)$, since these problems would not have the same solutions as P and thus are not interesting to us. We have instead a new problem P', which is contructed in such a way to "insert" the inconsistencies of $\gamma(\tilde{f}(\alpha(P)))$ into P. P' is obviously lower than P in the concrete partial order, since it is the same as P with the exception of some more $\mathbf{0}$'s, but the most important point is that it has the same solutions as P.

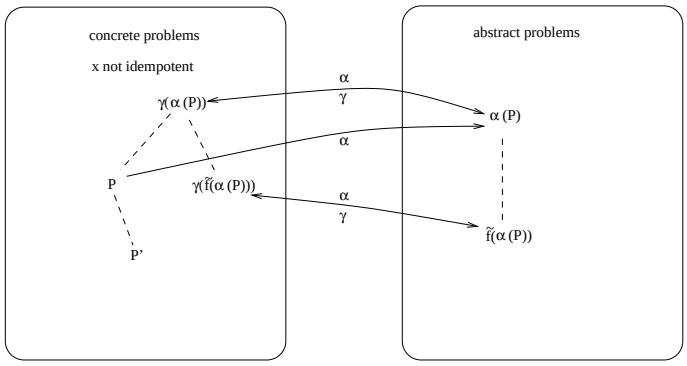

Fig. 7. The scheme when $\times$ is not idempotent.

Theorem 6. *Given an SCSP problem P over S, consider a function $\tilde{f}$ on $\alpha(P)$ which is safe, solution-correct and intensive. Then, if $\times$ is not idempotent, consider P' to be the SCSP which is the same as P except for those tuples which have semiring value $\mathbf{0}$ in $\gamma(\alpha(\tilde{f}(P)))$: these tuples are given value $\mathbf{0}$ also in P'. Then we have that $Sol(P) = Sol(P')$.*

Summarizing, the above theorems can give us several hints on how to use the abstraction scheme to make the solution of P easier: If $\times$ is idempotent, then we can replace P with $P \otimes \gamma(\alpha(\tilde{f}(P)))$, and get the same solutions (by Theorem 4). If instead $\times$ is not idempotent, we can replace P with P' (by Theorem 6). In any case, the point in passing from P to $P \otimes \gamma(\alpha(\tilde{f}(P)))$ (or P') is that the new problem should be easier to solve than P, since the semiring values of its tuples are more explicit, that is, closer to the values of these tuples in a completely solved problem.

More precisely, consider a branch-and-bound algorithm to find an optimal solution of P. Then, once a solution is found, its value will be used to cut away some branches, where the semiring value is worse than the value of the solution already found. Now, if the values of the tuples are worse in the new problem than in P, each branch will have a worse value and thus we might cut away more branches. Therefore, the search tree of the new problem is smaller than that of P.

Another point to notice is that, if using a greedy algorithm to find the initial solution (to use later as a lower bound), this initial phase in the new problem will lead to a better estimate, since the values of the tuples are worse in the new problem and thus close to the optimum. In the extreme case in which the change from P to the new problem brings the semiring values of the tuples to coincide with the value of their combination, it is possible to see that the initial solution is already the optimal one.

Notice also that, if $\times$ is not idempotent, a tuple of P' has either the same value as in P, or $\mathbf{0}$. Thus the initial estimate in P' is the same as that of P (since the initial solution must be a solution), but the search tree of P' is again smaller than that of P, since there may be partial solutions which in P have value different from $\mathbf{0}$ and in P' have value $\mathbf{0}$, and thus the global inconsistency may be recognized earlier.

The same reasoning used for Theorem 2 on $\alpha(P)$ can also be applied to $\tilde{f}(\alpha(P))$. In fact, since $\tilde{f}$ is safe, the solutions of $\tilde{f}(\alpha(P))$ have the same values as those of $\alpha(P)$. Thus also the optimal solution sets coincide. Therefore we have that $Opt(\tilde{f}(\alpha(P)))$ contains all the optimal solutions of P. This means that, in order to find an optimal solution of P, we can find all optimal solutions of $\tilde{f}(\alpha(P))$ and then use such a set to prune the search for an optimal solution of P.

Theorem 7. *Given an SCSP problem P over S, consider a function $\tilde{f}$ on P which is safe, solution-correct and intensive. Then we have that $Opt(P) \subseteq Opt(\tilde{f}(\alpha(P)))$.*

Theorem 3 can be adapted to $\tilde{f}(\alpha(P))$ as well, thus allowing us to use an optimal solution of $\tilde{f}(\alpha(P))$ to find both a lower and an upper bound of an optimal solution of P.

5 Conclusions and future work

We have proposed an abstraction scheme for abstracting soft constraint problems, with the goal of finding an optimal solution is a shorter time. The main idea is to work on the abstract version of the problem and then bring back some useful information to the concrete problem, to make it easier to solve.

Future work will concern the comparison between our properties and the one of [7], the generalization of our scheme to abstract also domains and graph topologies, and an experimental phase to assess the practical value of our proposal.

References

1. G. Birkhoff and S. MacLane. *A Survey of Modern Algebra*. MacMillan, 1965.
2. S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie. Semiring-based CSPs and valued CSPs: Basic properties and comparison. In *Over-Constrained Systems*. Springer-Verlag, LNCS 1106, 1996.
3. S. Bistarelli, U. Montanari, and F. Rossi. Constraint Solving over Semirings. In *Proc. IJCAI95*. Morgan Kaufman, 1995.
4. S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based Constraint Solving and Optimization. *Journal of the ACM*, 44(2):201–236, March 1997.
5. P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Fourth ACM Symp. Principles of Programming Languages*, pages 238–252, 1977.
6. P. Cousot and R. Cousot. Systematic design of program analyis. In *Sixth ACM Symp. Principles of Programming Languages*, pages 269–282, 1979.
7. S. de Givry, G. Verfaille, , and T. Schiex. Bounding The Optimum of Constraint Optimization Problems. In G. Smolka, editor, *Proc. CP97*, pages 405–419. Springer-Verlag, LNCS 1330, 1997.
8. D. Dubois, H. Fargier, and H. Prade. The calculus of fuzzy restrictions as a basis for flexible constraint satisfaction. In *Proc. IEEE International Conference on Fuzzy Systems*, pages 1131–1136. IEEE, 1993.
9. Thomas Ellman. Synthesis of abstraction hierarchies for constraint satisfaction by clustering approximately equivalent objects. In *Proceedings of International Conference on Machine Learning*, pages 104–111, 1993.
10. H. Fargier and J. Lang. Uncertainty in constraint satisfaction problems: a probabilistic approach. In *Proc. European Conference on Symbolic and Qualitative Approaches to Reasoning and Uncertainty (ECSQARU)*, pages 97–104. Springer-Verlag, LNCS 747, 1993.
11. E. C. Freuder and D Sabin. Interchangeability supports abstraction and reformulation for constraint satisfaction. In *Proceedings of Symposium on Abstraction, Reformulation and Approximation (SARA'95)*, 1995.
12. E. C. Freuder and R. J. Wallace. Partial constraint satisfaction. *AI Journal*, 58, 1992.

13. Y. Georget and P. Codognet. Compiling semiring-based constraints with `clp(FD,`S`)`. In M.Maher and J-F. Puget, editors, *Proc. CP98*. Springer-Verlag, LNCS 1520, 1998.
14. A.K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.
15. A.K. Mackworth. Constraint satisfaction. In Stuart C. Shapiro, editor, *Encyclopedia of AI (second edition)*, volume 1, pages 285–293. John Wiley & Sons, 1992.
16. Zs. Ruttkay. Fuzzy constraint satisfaction. In *Proc. 3rd IEEE International Conference on Fuzzy Systems*, pages 1263–1268, 1994.
17. T. Schiex. Possibilistic constraint satisfaction problems, or "how to handle soft constraints?". In *Proc. 8th Conf. of Uncertainty in AI*, pages 269–275, 1992.
18. T. Schiex, H. Fargier, and G. Verfaille. Valued Constraint Satisfaction Problems: Hard and Easy Problems. In *Proc. IJCAI95*, pages 631–637. Morgan Kaufmann, 1995.
19. R. Schrag and D. Miranker. An evaluation of domain reduction: Abstraction for unstructured csps. In *Proceedings of the Symposium on Abstraction, Reformulation, and Approximation*, pages 126–133, 1992.
20. M. Wallace. Practical applications of constraint programming. *Constraints: An International Journal*, 1, 1996.