Abstracting Soft Constraints

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 Roquencourt, 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) [18] are a very convenient and expressive formalism for many real-life problems, like scheduling, resource allocation, vehicle routing, timetabling, and many others [23]. 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 be too costly to find all, or even only one, optimal solution. Also, although classical propagation techniques like arc-consistency [17] can be extended 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 work on a simplified version of the given problem, trying however to not loose too much information. We propose to define this simplified version by means of the notion of abstraction, which takes an SCSP and returns a new one which is simpler to solve. Here, as in many other works on abstraction [16, 15], "simpler" may mean many things, like the fact that a certain solution algorithm finds a solution, or an optimal solution, in a fewer number of steps, or also that the abstracted problem can be processed by a machinery which is not available in the concrete context.

There are many formal proposals to describe the process of abstracting a notion, be it a formula, or a problem [16], or even a classical [11] or a soft CSP [21]. Among these, we chose to use one based on Galois insertions [5], mainly to refer to a well-know theory, with many results and properties that can be useful for our purposes. This made our approach compatible with the general theory of abstraction in [16]. Then, we adapted it to work on 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*. Note that this way of abstracting constraint problems does not change the structure of the problem (the set of variables remains the same, as well as the set of constraints), but just the semiring values to be associated to the tuples of values for the variables in each constraint.

Once we get the abstracted version of a given problem, we propose to

1. process the abstracted version: this may mean either solving it completely, or also applying some incomplete solver which may derive some useful information on the abstract problem;
2. bring back to the original problem some (or possibly all) of the information derived in the abstract context;
3. continue the solution process on the transformed problem, which is a concrete problem equivalent to the given one.

All this process has the main aim of finding an optimal solution, or an approximation of it, for the original SCSP, within the resource bounds we have. The hope is that, by following the above three steps, we get to the final goal faster than just solving the original problem.

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:

- If the abstraction satisfies a certain property, all optimal solutions of the concrete SCSP are also optimal in the corresponding abstract SCSP (see Theorem 4). 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 find upper and lower bounds for an optimal solution for the concrete problem (see Theorem 5). If we are satisfied with these bounds, we could just take the optimal solution of 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 6 and 8).
  This however can be done only if the semiring operation which describes how to combine constraints on the concrete side is idempotent (see Theorem 6).
- If instead this operation is not idempotent, still we can bring back some information from the abstract side. In particular, we can bring back the inconsistencies (that is, tuples with associated the worst element of the semiring), since we are sure that these same tuples are inconsistent also in the concrete SCSP (see Theorem 8).

In both the last two cases, the new concrete problem is easier to solve, in the sense, for example, that a branch-and-bound algorithm would explore a smaller (or equal) search tree before finding an optimal solution.

The paper is organized as follows. First, in Section 2 we give the necessary notions about soft CSPs and abstraction. Then, in Section 3 we define our notion of abstraction, and in Section 4 we prove properties of our abstraction scheme and discuss some of their consequences. Finally, in Section 7 we summarize our work and give hints about future directions.

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* [21, 8, 13, 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.

Definition 1 (semirings and c-semirings). *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 (see [4]):

- $\leq_S$ 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, for all $a, b \in A$, $a + b = lub(a, b)$.

Moreover, if $\times$ is idempotent, then $\langle A, \leq_S \rangle$ is a complete distributive lattice and $\times$ is its glb. Informally, the relation $\leq_S$ gives us a way to compare (some of the) tuples of values and constraints. In fact, when we have $a \leq_S b$, we will say that *b is better than a*.

Definition 2 (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. The correct way to interpret such elements depends 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. In the following we will also use the obvious extension of this relation to sets of constraints, and also to problems (seen as sets of constraints). Therefore, given two SCSPs P_1 and P_2 with the same graph topology, we will write $P_1 \sqsubseteq_S P_2$ if, for each constraint c_1 in P and the corresponding constraint c_2 in P_2, we have that $c_1 \sqsubseteq_S c_2$.

Definition 3 (soft constraint problem). *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, 19, 20] 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.

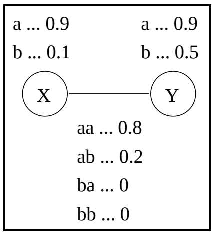

Fig. 1. A fuzzy CSP.

Definition 4 (combination). *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 $def(t) = def_1(t \downarrow^{con}_{con_1}) \times def_2(t \downarrow^{con}_{con_2})$*[1]*. The combination operator $\otimes$ can be straightforward extended also to sets of constraints: when applied to a set of constraints C, we will write $\bigotimes C$.*

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, commutative, and monotone over $\sqsubseteq_S$;
- if $\times$ is idempotent, $\otimes$ is idempotent as well.

Definition 5 (projection). *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 $def'(t') = \sum_{t/t\downarrow^{con}_{I\cap con}=t'} def(t)$.*

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.

Definition 6 (solution). *The* solution *of a SCSP problem $P = \langle C, con\rangle$ is the constraint $Sol(P) = (\bigotimes C) \Downarrow_{con}$.*

That is, to obtain the solution of an SCSP, 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).

[1] By $t \downarrow^X_Y$ we mean the projection of tuple t, which is defined over the set of variables X, over the set of variables $Y \subseteq X$.

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.

Definition 7 (optimal solution). *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 among those associated to solutions. The set of optimal solutions of an SCSP P will be written as $Opt(P)$.

Definition 8 (problem ordering and equivalence). *Consider two problems P_1 and P_2. Then $P_1 \sqsubseteq_P P_2$ if $Sol(P_1) \sqsubseteq_S Sol(P_2)$. If $P_1 \sqsubseteq_P P_2$ and $P_2 \sqsubseteq_P P_1$, then they have the same solution, thus we say that they are equivalent and we write $P_1 \equiv P_2$.*

The relation $\sqsubseteq_P$ is a preorder. Moreover, $P_1 \sqsubseteq_S P_2$ implies $P_1 \sqsubseteq_P P_2$. Also, $\equiv$ is an equivalence relation.

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 [17], can be generalized to be used for SCSPs.

The detailed formal definition of propagation algorithms (sometimes called also *local consistency* algorithms) for SCSPs can be found in [4]. For the purpose of this paper, what is important to say is 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 $Sol(P) = Sol(P')$, that is, local consistency algorithms do not change the set of solutions.

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 9 (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 $x \in \mathcal{C}, x \sqsubseteq \gamma(\alpha(x))$ and*
3. *for each $y \in \mathcal{A}, \alpha(\gamma(y)) \leq y$.*

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 $\gamma \cdot \alpha = Id_{\mathcal{A}}$.

Property 2 is called *extensivity* of $\alpha \cdot \gamma$. The map α (γ) is called the *lower* (*upper*) *adjoint* or *abstraction* (*concretization*) in the context of abstract interpretation.

The following basic properties are satisfied by any Galois insertion:

1. γ is injective and α is surjective.
2. $\alpha \cdot \gamma$ 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,
$$\gamma = \lambda y. \bigcup_{\mathcal{C}} \{x \in \mathcal{C} \mid \alpha(x) \sqsubseteq y\}, \alpha = \lambda x. \bigcap_{\mathcal{A}} \{y \in \mathcal{A} \mid x \leq \gamma(y)$$
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.

One property that will be useful later relates to a precise relationship between the ordering in the concrete lattice and that in the abstract one.

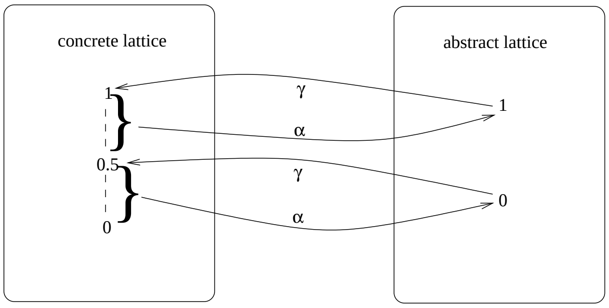

Fig. 2. A Galois insertion.

Theorem 1 (total ordering). *Consider a Galois insertion from* $(\mathcal{C}, \sqsubseteq)$ *to* $(\mathcal{A}, \leq)$. *Then, if* $\sqsubseteq$ *is a total order, also* $\leq$ *is so.*

Proof. It easy follows from the monotonicity of α (that is, $x \sqsubseteq y$ implies $\alpha(x) \leq \alpha(y)$, and from its surjectivity (that is, there is no element in $\mathcal{A}$ which is not the image of some element in $\mathcal{C}$ via α). □

Usually, both the concrete and the abstract lattice have some operators that are used to define the corresponding semantics. Most of the times it is useful, and required, that the abstract operators show a certain relationship with the corresponding concrete ones. This relationship is called *local correctness*.

Definition 10 (local correctness). *Let* $f : \mathcal{C}^n \rightarrow \mathcal{C}$ *be an operator over the concrete lattice, and assume that* $\tilde{f}$ *is its abstract counterpart. Then* $\tilde{f}$ *is locally correct w.r.t.* f *if* $\forall x_1, \ldots, x_n \in \mathcal{C}. f(x_1, \ldots, x_n) \sqsubseteq \gamma(\tilde{f}(\alpha(x_1), \ldots, \alpha(x_n)))$.

3 Abstracting soft CSPs

Given the notions of soft constraints and abstraction, recalled in the previous sections, 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.

Definition 11 (abstracting SCSPs). *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* $def_i : D^{|con_i|} \rightarrow A$;

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* $\alpha : L \to \tilde{L}$;
- $\tilde{\times}$ *is locally correct with respect to* $\times$.

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 $\alpha(1) = \tilde{1}$.

Example 1. As an example, consider any SCSP over the semiring for optimization

$$S_{WCSP} = \langle \mathcal{R}^- \cup \{-\infty\}, \max, +, -\infty, 0 \rangle$$

(where costs are represented by negative reals) 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 maximum of the sum of all costs, we just want to compute the maximum of the minimum of all costs, and we want to normalize the costs over $[0..1]$. Notice that the abstract problem is in the FCSP class and it has an idempotent $\times$ operator (which is the min). This means that in the abstract framework we can perform local consistency over the problem in order to find inconsistencies. As noted above, the mapping $\alpha : \langle \mathcal{R}^-, \leq_{WCSP} \rangle \to \langle [0,1], \leq_{FCSP} \rangle$ can be defined in different ways. For example one can decide to map all the reals below some fixed real x onto 0 and then to map the reals in $[x, 0]$ into the reals in $[0, 1]$ by using a normalization function, for example $f(r) = \frac{x-r}{x}$.

Example 2. 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.

An important property of our notion of abstraction is that the composition of two abstractions is still an abstraction. This allows to build a complex abstraction by defining several simpler abstractions to be composed.

Theorem 2 (abstraction composition). *Consider an abstraction from the lattice corresponding to a semiring S_1 to that corresponding to a semiring S_2, denoted by the pair $\langle \alpha_1, \gamma_1 \rangle$. Consider now another abstraction from the lattice corresponding to the semiring S_2 to that corresponding to a semiring S_3, denoted by the pair $\langle \alpha_2, \gamma_2 \rangle$. Then the pair $\langle \alpha_1 \cdot \alpha_2, \gamma_2 \cdot \gamma_1 \rangle$ is an abstraction as well.*

Proof. We first have to prove that $\langle \alpha, \gamma \rangle = \langle \alpha_1 \cdot \alpha_2, \gamma_2 \cdot \gamma_1 \rangle$ satisfies the four properties of a Galois insertion:

- since the composition of monotone functions is again a monotone function, we have that both α and γ are monotone functions;
- given a value $x \in S_1$, from the first abstraction we have that $x \sqsubseteq_1 \gamma_1(\alpha_1(x))$. Moreover, for any element $y \in S_2$, we have that $y \sqsubseteq_2 \gamma_2(\alpha_2(y))$. This holds also for $y = \alpha_1(x)$, thus by monotonicity of γ_1 we have $x \sqsubseteq_1 \gamma_1(\gamma_2(\alpha_2(\alpha_1(x))))$.
- a similar proof can be used for the third property;
- the composition of two identities is still an identity .

To prove that $\times_3$ is locally correct w.r.t. $\times_1$, it is enough to consider the local correctness of $\times_2$ w.r.t. $\times_1$ and of $\times_3$ w.r.t. $\times_2$, and the monotonicity of γ_1. □

4 Properties and advantages of the abstraction

In this section we will define and prove several properties that hold on abstractions of soft constraint problems. The main goal here is to point out some of the advantages that one can have in passing through the abstracted version of a soft constraint problem instead of solving directly the concrete version.

4.1 Relating a soft constraint problem and its abstract version

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 $\langle \alpha, \gamma \rangle : \langle A, \leq_S \rangle \rightleftharpoons \langle \tilde{A}, \leq_{\tilde{S}} \rangle$.

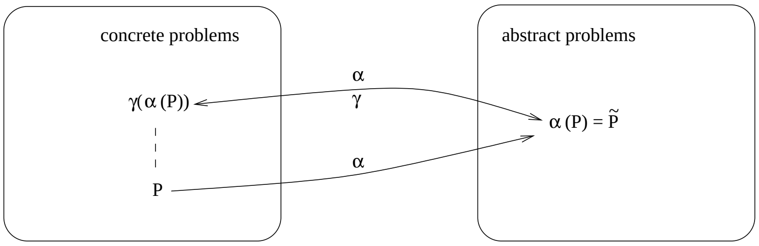

Fig. 3. The concrete and the abstract problem.

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

Proof. Immediately follows from the properties of a Galois insertion, in particular from the fact that $x \leq_S \gamma(\alpha(x))$ for any x in the concrete lattice. In fact, $P \sqsubseteq_S \gamma(\alpha(P))$ means that, for each tuple in each constraint of P, 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 $\gamma(\alpha(P))$. □

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 $Sol(P) \sqsubseteq_S Sol(\gamma(\alpha(P)))$.*

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

Therefore, by passing from P to $\gamma(\alpha(P))$, no new inconsistencies are introduced: if a solution of $\gamma(\alpha(P))$ has value $\mathbf{0}$, then this was true also in P. However, it is possible that some inconsistencies are forgotten (that is, they appear to be consistent after the abstraction process).

If the abstraction preserves the semiring ordering (that is, applying the abstraction function and then combining gives elements which are in the same ordering as the elements obtained by combining only), then 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 $\alpha(P)$. Let us first investigate the meaning of the *order-preserving* property.

Definition 12 (order-preserving abstraction). *Consider two sets I_1 and I_2 of concrete elements. Then an abstraction is said to be order-preserving if*

$$\tilde{\prod_{x \in I_1}} \alpha(x) \leq_{\tilde{S}} \tilde{\prod_{x \in I_2}} \alpha(x) \Rightarrow \prod_{x \in I_1} x \leq_S \prod_{x \in I_2} x$$

where the products refer to the multiplicative operations of the concrete ($\prod$) and the abstract ($\tilde{\prod}$) semirings.

In words, this notion of order-preservation means that if we first abstract and then combine, or we combine only, we get the same ordering (but on different semirings) among the resulting elements.

Example 3. An abstraction which is not order-preserving can be seen in Figure 4. Here, the concrete and the abstract sets, as well as the additive operations of the two semirings, can be seen from the picture. For the multiplicative operations, we assume they coincide with the glb of the two semirings.

In this case, the concrete ordering is partial, while the abstract ordering is total. Functions α and β are depicted in the figure by arrows going from the concrete semiring to the abstract one and vice versa. Assume that the concrete problem has no solution with value 1. Then all solutions with value a or b are optimal. Suppose a solution with value b is obtained by computing $b = 1 \times b$, while we have $a = 1 \times a$. Then the abstract counterparts will have values $\alpha(1) \times' \alpha(b) = 1 \times' 0 = 0$ and $\alpha(1) \times' \alpha(a) = 1 \times' 1 = 1$. Therefore the solution with value a, which is optimal in the concrete problem, is not optimal anymore in the abstract problem.

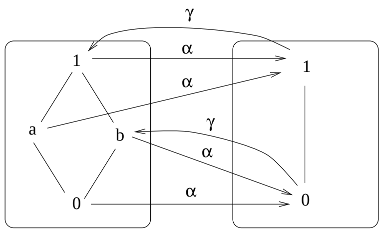

Fig. 4. An abstraction which is not order-preserving.

Example 4. The abstraction in Figure 2 is order-preserving. In fact, consider two abstract values which are ordered, that is $0 \leq' 1$. Then $1 = 1 \times' 1 = \alpha(x) \times' \alpha(y)$, where both x and y must be greater than 0.5. Thus their concrete combination (which is the min), say v, is always greater than 0.5. On the other hand, 0 can be obtained by combining either two 0's (therefore the images of two elements smaller than or equal to 0.5, whose minimum is smaller than 0.5 and thus smaller than v), or by combining a 0 and a 1, which are images of a value greater than 0.5 and one smaller than 0.5. Also in this case, their combination (the min) is smaller than 0.5 and thus smaller than v. Thus the order-preserving property holds.

Example 5. Another abstraction which is not order-preserving is the one that passes from the semiring $\langle N \cup \{+\infty\}, min, sum, 0, +\infty\rangle$, where we minimize the sum of values over the naturals, to the semiring $\langle N \cup \{+\infty\}, min, max, 0, +\infty\rangle$, where we minimize the maximum of values over the naturals. In words, this abstraction maintains the same domain of the semiring, and the same additive operation (min), but it changes the multiplicative operation (which passes from sum to max). Notice that the semiring orderings are the opposite as those usually used over the naturals: if i is smaller than j then $j \leq_S i$, thus the best element is 0 and the worst is $+\infty$. The abstraction function α is just the identity, and also the concretization function γ.

In this case, consider in the abstract semiring two values and the way they are obtained by combining other two values of the abstract semiring: for example, $8 = max(7, 8)$ and $9 = max(1, 9)$. In the abstract ordering, 8 is higher than 9. Then, let us see how the images of the combined values (the same values, since α is the identity) relate to each other: we have $sum(7, 8) = 15$ and $sum(1, 9) = 10$, and 15 is lower than 10 in the concrete ordering. Thus the order-preserving property does not hold.

Notice that, if we reduce the sets I_1 and I_2 to singletons, say x and y, then the order-preserving property says that $\alpha(x) \leq_{\tilde{S}} \alpha(y)$ implies that $x \leq_S y$. This means that if two abstract objects are ordered, then their concrete counterparts have to be ordered as well, and in the same way. Of course they could never be ordered in the opposite sense, otherwise α would not be monotonic; but they could be incomparable. Therefore, if we choose an abstraction where incomparable objects are mapped by α onto ordered objects, then we don't have the order-preserving property. A consequence of this is that if the abstract semiring is totally ordered, and we want an order-preserving abstraction, then the concrete semiring must be totally ordered as well.

On the other hand, if two abstract objects are not ordered, then the corresponding concrete objects can be ordered in any sense, or they can also be not comparable. Notice that this restriction of the order-preserving property to singleton sets always holds when the concrete ordering is total. In fact, in this case, if two abstract elements are ordered in a certain way, then it is impossible that the corresponding concrete elements are ordered in the opposite way, because, as we said above, of the monotonicity of the α function.

Theorem 4. *Consider an abstraction which is order-preserving. Given an SCSP problem P over S, we have that $Opt(P) \subseteq Opt(\alpha(P))$.*

Proof. Let us take a tuple t which is optimal in the concrete semiring S, with value v. Then v has been obtained by multiplying the values of some subtuples. Suppose, without loss of generality, that the number of such subtuples is two (that is, we have two constraints): $v = v_1 \times v_2$. Let us then take the value of this tuple in the abstract problem, that is, the abstract combination of the abstractions of v_1 and v_2: this is $v' = \alpha(v_1) \times' \alpha(v_2)$. We have to show that if v is optimal, then also v' is optimal.

Suppose then that v' is not optimal, that is, there exists another tuple t'' with value v'' such that $v' \leq_{S'} v''$. Assume $v'' = v_1'' \times' v_2''$. Now let us see the value of tuple t'' in P. If we set $v_i'' = \alpha(\bar{v_i})$, then we have that this value is $\bar{v} = \bar{v_1} \times \bar{v_2}$. Let us now compare v with $\bar{v}$. Since $v' \leq_{\tilde{S}} v''$, by order-preservation we get that $v \leq_S \bar{v}$. But this means that v is not optimal, which was our initial assumption. Therefore v' has to be optimal. □

Therefore, in case of order-preservation, the set of optimal solutions of the abstract problem contains all the optimal solutions of the concrete problem. In other words, it is not allowed that an optimal solution in the concrete domain becomes non-optimal in the abstract domain. However, some non-optimal solutions could become optimal by becoming incomparable with the optimal solutions.

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 important property, which holds for any abstraction, concerns computing bounds that approximate an optimal solution of a concrete problem. In fact, any optimal solution, say t, of the abstract problem, say with value $\tilde{v}$, can be used to obtain both an upper and a lower bound of an optimum in P. In fact, we can prove that there is an optimal solution in P with value between $\gamma(\tilde{v})$ and the value of t in P. Thus, if we think that approximating the optimal value with a value within these two bounds is satisfactory, we can take t as an approximation of an optimal solution of P.

Theorem 5. *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. Then there exists an optimal solution $\bar{t}$ of P, say with value $\bar{v}$, such that $v \leq \bar{v} \leq \gamma(\tilde{v})$.*

Proof. By local correctness of the multiplicative operation of the abstract semiring, we have that $v \leq_S \gamma(\tilde{v})$. Since v is the value of t in P, either t itself is optimal in P, or there is another tuple which has value better than v, say $\bar{v}$. We will now show that $\bar{v}$ cannot be greater than $\gamma(\tilde{v})$.

In fact, assume by absurd that $\gamma(\tilde{v}) \leq_S \bar{v}$. Then, by local correctness of the multiplicative operation of the abstract semiring, we have that $\alpha(\bar{v})$ is smaller than the value of $\bar{t}$ in $\alpha(P)$. Also, by monotonicity of α, by $\gamma(\tilde{v}) \leq_S \bar{v}$ we get that $\tilde{v} \leq_{\tilde{S}} \alpha(\bar{v})$. Therefore by transitivity we obtain that $\tilde{v}$ is smaller than the value of $\bar{t}$ in $\alpha(P)$, which is not possible because we assumed that $\tilde{v}$ was optimal.

Therefore there must be an optimal value between v and $\gamma(\tilde{v})$. □

Thus, given a tuple t with optimal value $\tilde{v}$ in the abstract problem, instead of spending time to compute an exact optimum of P, we can do the following:

- compute $\gamma(\tilde{v})$, thus obtaining an upper bound of an optimum of P;
- compute the value of t in P, which is a lower bound of the same optimum of P;
- If we think that such bounds are close enough, we can take t as a reasonable approximation (to be precise, a lower bound) of an optimum of P.

Notice that this theorem does not need the order-preserving property in the abstraction, thus any abstraction can exploit its result.

4.2 Working on the abstract problem

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, functions $\tilde{f}$ 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 13. *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 14. *Given an SCSP problem P and a function $f : PL \to PL$, f is safe if $Sol(P) = Sol(f(P))$.*

It is easy to see that any local consistency algorithm, as defined in [4], can be seen as a safe, intensive, and solution-correct function.

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 5 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, we have that

- $P \otimes \gamma(\tilde{f}(\alpha(P)))$ is smaller than both P and $\gamma(\tilde{f}(\alpha(P)))$ because of the properties of $\otimes$;
- $\gamma(\tilde{f}(\alpha(P)))$ 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 $\tilde{f}$;
- $f_{sol}(P)$ is smaller than P because of the way $f_{sol}(P)$ is constructed;
- 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 $\gamma(\tilde{f}(\alpha(P)))$.

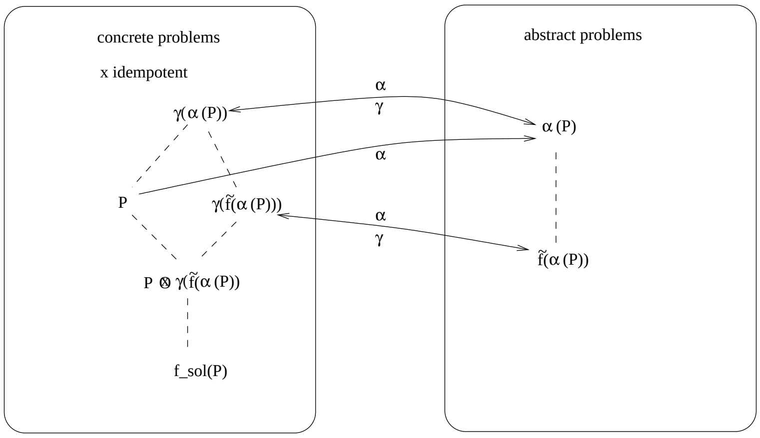

Fig. 5. The general abstraction scheme, with $\times$ idempotent.

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 idempotent, $Sol(P) = Sol(P \otimes \gamma(\tilde{f}(\alpha(P))))$.*

Proof. Take any tuple t with value v in P, which is obtained by combining the values of some subtuples, say two: $v = v_1 \times v_2$. Let us now consider the abstract versions of v_1 and v_2: $\alpha(v_1)$ and $\alpha(v_2)$. Function $\tilde{f}$ changes these values by lowering them, thus we get $\tilde{f}(\alpha(v_1)) = v'_1$ and $\tilde{f}(\alpha(v_2)) = v'_2$.

Since $\tilde{f}$ is safe, we have that $v'_1 \times' v'_2 = \alpha(v_1) \times' \alpha(v_2) = v'$. Moreover, $\tilde{f}$ is solution-correct, thus $v \leq_S \gamma(v')$. By monotonicity of γ, we have that $\gamma(v') \leq_S \gamma(v'_i)$ for $i = 1, 2$. This implies that $\gamma(v') \leq_S (\gamma(v'_1) \times \gamma(v'_2))$, since $\times$ is idempotent by assumption and thus it coincides with the glb. Thus we have that $v \leq_S (\gamma(v'_1) \times \gamma(v'_2))$.

To prove that P and $P \otimes \gamma(\tilde{f}(\alpha(P)))$ give the same value to each tuple, we now have to prove that $v = (v_1 \times \gamma(v'_1)) \times (v_2 \times \gamma(v'_2))$. By commutativity of $\times$, we can write this as $(v_1 \times v_2) \times (\gamma(v'_1) \times \gamma(v'_2))$. Now, $v_1 \times v_2 = v$ by assumption, and we have shown that $v \leq_S \gamma(v'_1) \times \gamma(v'_2)$. Therefore $v \times (\gamma(v'_1) \times \gamma(v'_2)) = v$. $\square$

This theorem does not say anything about the power of $\tilde{f}$, which could make many modifications to $\alpha(P)$, or it could also not modify anything. In this last case, $\gamma(\tilde{f}(\alpha(P))) = \gamma(\alpha(P)) \sqsupseteq P$ (see Figure 6), 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 4 and 5) to find an approximation of the optimal solutions and of the inconsistencies of P.

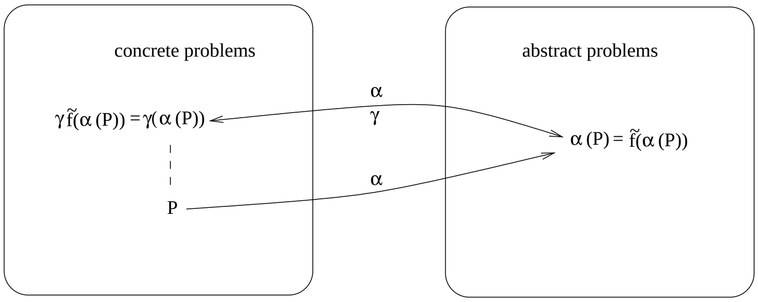

Fig. 6. The scheme when $\tilde{f}$ does not modify anything.

If instead $\tilde{f}$ modifies all semiring elements in $\alpha(P)$, 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 7), 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 7. *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 every semiring element in $\alpha(P)$, and the order of the concrete semiring is total, we have that $P \sqsupseteq_S \gamma(\tilde{f}(\alpha(P))) \sqsupseteq_S f_{sol}(P)$.*

Proof. Consider any tuple t in any constraint of P, and let us call v its semiring value in P and v_{sol} its value in $f_{sol}(P)$. Obviously, we have that $v_{sol} \leq_S v$. Now take $v' = \gamma(\alpha(v))$. By monotonicity of α, we cannot have $v <_S v'$. Also, by solution-correctness of $\tilde{f}$, we cannot have $v' <_S v_{sol}$. Thus we must have $v_{sol} \leq_S v' \leq_s v$, which proves the statement of the theorem. $\square$

Notice that we need the idempotence of the $\times$ operator for Theorem 6 and 7. If instead $\times$ is not idempotent, then we can prove something weaker. Figure 8 shows this situation. With respect to Figure 5, we can see that the possible non-idempotence 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

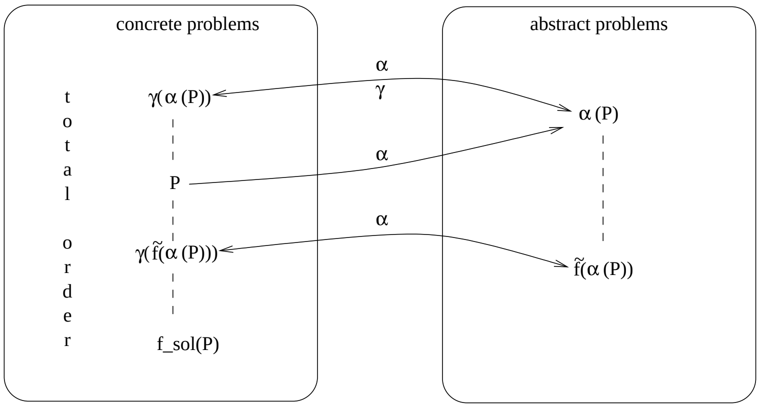

Fig. 7. The scheme when the concrete semiring has a total order.

and thus are not interesting to us. We have instead a new problem P', which is constructed 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.

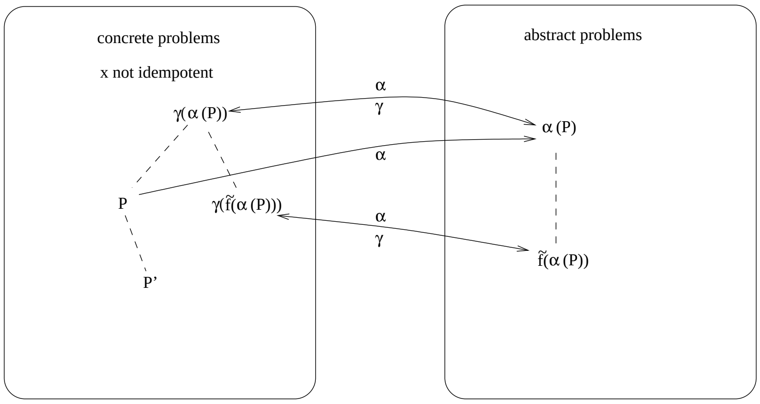

Fig. 8. The scheme when $\times$ is not idempotent.

Theorem 8. *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* $Sol(P) = Sol(P')$*.*

Proof. Take any tuple t with value v in P, which is obtained by combining the values of some subtuples, say two: $v = v_1 \times v_2$. Let us now consider the abstract versions of v_1 and v_2: $\alpha(v_1)$ and $\alpha(v_2)$. Function $\tilde{f}$ changes these values by lowering them, thus we get $\tilde{f}(\alpha(v_1)) = v_1'$ and $\tilde{f}(\alpha(v_2)) = v_2'$.

Since $\tilde{f}$ is safe, we have that $v_1' \times' v_2' = \alpha(v_1) \times' \alpha(v_2) = v'$. Moreover, $\tilde{f}$ is solution-correct, thus $v \leq_S \gamma(v')$. By monotonicity of γ, we have that $\gamma(v') \leq_S \gamma(v_i')$ for $i = 1, 2$. Thus we have that $v \leq_S \gamma(v_i')$ for $i = 1, 2$.

Now suppose that $\gamma(v_1') = \mathbf{0}$. This implies that also $v = \mathbf{0}$. Therefore, if we set $v_1 = \mathbf{0}$, again the combination of v_1 and v_2 will result in v, which is $\mathbf{0}$. □

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 6). If instead $\times$ is not idempotent, we can replace P with P' (by Theorem 8). 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. For example, consider the fuzzy semiring (that is, we want to maximize the minimum of the values of the subtuples): if the solution already found has value 0.6, then each partial solution of P with value smaller than or equal to 0.6 can be discarded (together with all its corresponding subtree in the search tree), but all partial solutions with value greater than 0.6 must be considered; if instead we work in the new problem, the same partial solution with value greater than 0.6 may now have a smaller value, possibly also smaller than 0.6, and thus can be disregarded. 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 4 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 if the abstraction is order-preserving. 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 9. *Given an SCSP problem P over S, consider a function $\tilde{f}$ on P which is safe, solution-correct and intensive, and let us assume the abstraction is order-preserving. Then we have that $Opt(P) \subseteq Opt(\tilde{f}(\alpha(P)))$.*

Proof. Easy follows from Theorem 4 and from the safeness of $\tilde{f}$. □

Theorem 5 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 Some abstraction mappings

In this section we will list some semirings and several abstractions between them, in order to provide the reader with a scenario of possible abstractions that he/she can use, starting from one of the semirings considered here. Some of these semirings and/or abstractions have been already described in the previous sections of the paper, however here we will re-define them to make this section self-contained. Of course many other semirings could be defined, but here we focus on the ones for which either it has been defined, or it is easy to imagine, a system of constraint solving. The semirings we will consider are the following ones:

- the classical one, which describes classical CSPs via the use of logical and and logical or:

$$S_{CSP} = \langle \{T, F\}, \vee, \wedge, F, T \rangle$$

- the fuzzy semiring, where the goal is to maximize the minimum of some values over $[0, 1]$:

$$S_{fuzzy} = \langle [0,1],\ \max,\ \min, 0, 1 \rangle$$

- the extension of the fuzzy semiring over the naturals, where the goal is to maximize the minimum of some values of the naturals:

$$S_{fuzzyN} = \langle N \cup \{+\infty\},\ \max,\ \min, 0, +\infty \rangle$$

- the extension of the fuzzy semiring over the positive reals:

$$S_{fuzzyR} = \langle R^+ \cup \{+\infty\},\ \max,\ \min, 0, +\infty \rangle$$

– the optimization semiring over the naturals, where we want to maximize the sum of costs (which are negative integers):

$$S_{optN} = \langle Z^- \cup \{-\infty\}, \max, +, -\infty, 0\rangle$$

– the optimization semiring over the negative reals:

$$S_{optR} = \langle R^- \cup \{-\infty\}, \max, +, -\infty, 0\rangle$$

– the probabilistic semiring, where we want to maximize a certain probability which is obtained by multiplying several individual probabilities. The idea here is that each tuple in each constraint has associated the probability of being allowed in the real problems we are modeling, and different tuples in different constraints have independent probabilities (so that their combined probability is just the multiplication of their individual probabilities) [10]. The semiring is:

$$S_{prob} = \langle [0,1],\ \max, \times, 0, 1\rangle$$

– the subset semiring, where the elements are all the subsets of a certain set, the operations are set intersection and set union, the smallest element is the empty set, and the largest element is the whole given set:

$$S_{sub} = \langle \mathcal{P}(A), \bigcup, \bigcap, \emptyset, A\rangle.$$

We will now define several abstractions between pairs of these semirings. The result is drawn in Figure 9, where the dashed lines denote the abstractions that have not been defined but can be obtained by abstraction composition. In reality, each line in the figure represents a whole family of abstractions, since each $\langle\alpha, \gamma\rangle$ pair makes a specific choice which identifies a member of the family. Moreover, by defining one of this families of abstractions we do not want to say that there do not exist other abstractions between the two semirings.

It is easy to see that some abstractions focus on the domain, by passing to a given domain to a smaller one, others change the semiring operations, and others change both:

1. from fuzzy to classical CSPs: this abstraction changes both the domain and the operations. The abstraction function is defined by choosing a threshold within the interval $[0,1]$, say x, and mapping all elements in $[0,x]$ to F and all elements in $(x,1]$ to T. Consequently, the concretization function maps T to 1 and F to x. See Figure 2 as an example of such an abstraction. We recall that all the abstractions in this family are order-preserving, so Theorem 4 can be used.
2. from fuzzy over the positive reals to fuzzy CSPs: this abstraction changes only the domain, by mapping the whole set of positive reals into the $[0,1]$ interval. This means that the abstraction function has to set a threshold, say x, and map all reals above x into 1, and any other real, say r, into $\frac{r}{x}$. Then, the concretization function will map 1 into $+\infty$, and each element of $[0,1)$,

say y, into $y \times x$. It is easy to prove that all the members of this family of abstractions are order-preserving.

3. from probabilistic to fuzzy CSPs: this abstraction changes only the multiplicative operation of the semiring, that is, the way constraints are combined. In fact, instead of multiplying a set of semiring elements, in the abstracted version we choose the minimum value among them. Since the domain remains the same, both the abstraction and the concretization functions are the identity (otherwise they would not have the properties required by a Galois insertion, like monotonicity). Thus this family of abstractions contains just one member.

   It is easy to see that this abstraction is not order-preserving. In fact, consider for example the elements 0.6 and 0.5, obtained in the abstract domain by $0.6 = min(0.7, 0.6)$ and $0.5 = min(0.9, 0.5)$. These same combinations in the concrete domain would be $0.7 \times 0.6 = 0.42$ and $0.9 \times 0.5 = 0.45$, thus resulting in two elements which are in the opposite order with respect to 0.5 and 0.6.
4. from optimization-N to fuzzy-N CSPs: here the domain remains the same (the negative integers) and only the multiplicative operation is modified. Instead of summing the values, we want to take their minimum. As noted in a previous example, these abstractions are not order-preserving.
5. from optimization-R to fuzzy-R CSPs: similar to the previous one but on the negative reals.
6. from optimization-R to optimization-N CSPs: here we have to map the negative reals into the negative integers. The operations remain the same. A possible example of abstraction is the one where $\alpha(x) = \lceil x \rceil$ and $\gamma(x) = x$. It is not order-preserving.
7. from fuzzy-R to fuzzy-N CSPs: again, we have to map the positive reals into the naturals, while maintaining the same operations. The abstraction could be the same as before, but in this case it is order-preserving (because of the use of min instead of sum).
8. from fuzzy-N to classical CSPs: this is similar to the abstraction from fuzzy CSPs to classical ones. The abstraction function has to set a threshold, say x, and map each natural in $[0, x]$ into F, and each natural above x into T. The concretization function maps T into $+\infty$ and F into x. All such abstractions are order-preserving.
9. from subset CSPs to any of the other semirings: if we want to abstract to a semiring with domain A, we start from the semiring with domain $\mathcal{P}(A)$. The abstraction mapping takes a set of elements of A and has to choose one of them by using a given function, for example min or max. The concretization function will then map an element of A into the union of all the corresponding sets in $\mathcal{P}(A)$. For reason similar to those used in Example 3, some abstractions of this family may be not order-preserving.

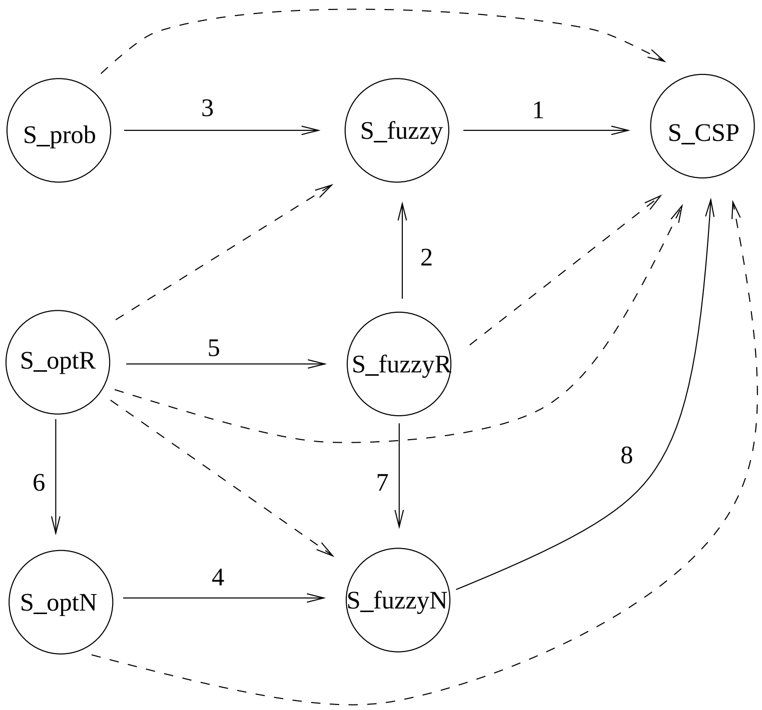

Fig. 9. Several semiring and abstractions between them.

6 Related work

We will compare here our work to other abstraction proposals, more or less related to the concepts of constraints.

Abstracting valued CSPs. The only other abstraction scheme for soft constraint problems we are aware of is the one in [7], where *valued CSPs* [21] 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, in valued CSPs, the goal is to minimize the value associated to a complete assignment. In valued CSPs, each constraint has one associated element, not one for each tuple of domain values of its variables. However, our notion of soft CSPs and that in valued CSPs are just different formalizations of the same idea, since 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 proven for the classical case. It is also worth noticing that our notion of an order-preserving abstraction is related to their concept of aggregation compatibility, although generalized to deal with partial orders.

Abstracting classical CSPs. Other work related to abstracting constraint problems proposed the abstraction of the domains [11, 12, 22], or of the graph topology (for example to model a subgraph as a single variable or constraint) [9]. We did not focus on these kinds of abstractions for SCSPs in this paper, but we believe that they could be embedded into our abstraction framework: we just need to define the abstraction function in such a way that not only we can change the semiring but also any other feature of the concrete problem. The only difference will be that we cannot define the concrete and abstract lattices of problems by simply extending the lattices of the two semirings.

A general theory of abstraction. A general theory of abstraction has been proposed in [16]. The purpose of this work is to define a notion of abstraction that can be applied to many domains: from planning to problem solving, from theorem proving to decision procedures. Then, several properties of this notion are considered and studied. The abstraction notion proposed consists of just two formal systems Σ_1 and Σ_2 with languages L_1 and L_2 and an effective total function $f : L_1 \rightarrow L_2$, and it is written as $f : \Sigma_1 \Rightarrow \Sigma_2$. Much emphasis is posed in [16] onto the study of the properties that are preserved by passing from the concrete to the abstract system. In particular, one property that appears to be very desirable, and present in most abstraction frameworks, is that *what is a theorem* *in the concrete domain, remains a theorem in the abstract domain* (called the **TI** property, for *Theorem Increasing*).

It is easy to see that our definition of abstraction is an instance of this general notion. Then, to see whether our concept of abstraction has this property, we first must say what is a theorem in our context. A natural and simple notion of a theorem could be an SCSP which has at least one solution with a semiring value different from the 0 of the semiring. However, we can be more general that this, and say that a theorem for us is *an SCSP which has a solution with value greater than or equal to k, where $k \geq 0$*. Then we can prove our version of the TI property:

Theorem 10 (our TI property). *Given an SCSP P which has a solution with value $v \geq k$, then the SCSP $\alpha(P)$ has a solution with value $v' \geq \alpha(k)$.*

Proof. Take any tuple t in P with value $v > k$. Assume that $v = v_1 \times v_2$. By abstracting, we have $v' = \alpha(v_1) \times' \alpha(v_2)$. By solution correctness of $\times'$, we have that $v \leq_S \gamma(v')$. By monotonicity of α, we have that $\alpha(v) \leq_{S'} \alpha(\gamma(v')) = v'$. again by monotonicity of α, we have $\alpha(k) \leq_{S'} \alpha(v)$, thus by transitivity $\alpha(k) \leq_S v'$. □

Notice that, if we consider the boolean semiring (where a solution has either value true or false), this statement reduces to saying that if we have a solution in the concrete problem, then we also have a solution in the abstract problem, which is exactly what the TI property says in [16]. Thus our notion of abstraction, as defined in the previous sections, on one side can be cast within the general theory proposed in [16], while on the other side it generalizes it to concrete and abstract domains which are more complex than just the boolean semiring. This is predictable, because, while in [16] formulas can be either true (thus theorems) or false, here they may have any level of satisfaction which can be described by the given semiring.

Notice also that, in our definition of abstraction of an SCSP, we have chosen to have a Galois insertion between the two lattices $\langle A, \leq \rangle$ (which corresponds to the concrete semiring S) and $\langle \tilde{A}, \tilde{\leq} \rangle$ (which corresponds to the abstract semiring $\tilde{S}$). This means that the ordering in the two lattices coincide with those of the semirings. We could have chosen differently: for example, that the ordeing of the lattices in the abstraction be the opposite of those of those in the semirings. In that case, we would not have had property **TI**. However, we would have the dual property (called **TD** in [16]), which states that abstract theorems remain theorems in the concrete domain. It has been shown that such a property can be useful in some application domains, such as databases.

7 Conclusions and future work

We have proposed an abstraction scheme for abstracting soft constraint problems, with the goal of finding an optimal solution, or a good approximation of it, in 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.

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 [14], which can already solve soft constraints in the classical way (branch-and-bound plus propagation via partial arc-consistency).

Another line for future research concerns the generalization of our approach to include also domain and topological abstractions, as already considered for classical CSPs.

Acknowledgments

This work has been partially supported by Italian MURST project TOSCA.

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. Eliminating interchangeable values in constraint satisfaction subproblems. In *Proc. AAAI-91*, 1991.

12. 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. http://www.cs.unh.edu/csprojects/csp/csp.html.
13. E. C. Freuder and R. J. Wallace. Partial constraint satisfaction. *AI Journal*, 58, 1992.
14. 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.
15. F. Giunchiglia, A. Villafiorita, and T. Walsh. Theories of abstraction. *AI Communication*, 10(3-4):167–176, 1997.
16. F. Giunchiglia and T. Walsh. A theory of abstraction. *Artificial Intelligence*, 56(2-3):323–390, 1992.
17. A.K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.
18. A.K. Mackworth. Constraint satisfaction. In Stuart C. Shapiro, editor, *Encyclopedia of AI (second edition)*, volume 1, pages 285–293. John Wiley & Sons, 1992.
19. Zs. Ruttkay. Fuzzy constraint satisfaction. In *Proc. 3rd IEEE International Conference on Fuzzy Systems*, pages 1263–1268, 1994.
20. T. Schiex. Possibilistic constraint satisfaction problems, or "how to handle soft constraints?". In *Proc. 8th Conf. of Uncertainty in AI*, pages 269–275, 1992.
21. T. Schiex, H. Fargier, and G. Verfaille. Valued Constraint Satisfaction Problems: Hard and Easy Problems. In *Proc. IJCAI95*, pages 631–637. Morgan Kaufmann, 1995.
22. 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. http://www.cs.utexas.edu/users/schrag/SARA.ps.
23. M. Wallace. Practical applications of constraint programming. *Constraints: An International Journal*, 1, 1996.