



**Systemhaus für intelligente  
EDV-Anwendungen**

Richard-Wagner-Straße 91  
67655 Kaiserslautern  
Telefon +49 631 65566  
Telefax +49 631 65464  
E-Mail: [sieda@sieda.com](mailto:sieda@sieda.com)  
WWW: <http://www.sieda.com>

## **Global Constraints with Fuzzy Extensions**

**Harald Meyer auf'm Hofe**

SIEDA W-2002/1

Technical Report



# Global Constraints with Fuzzy Extensions

Harald Meyer auf'm Hofe

## Abstract

Undoubtedly, constraint-based reasoning owes a large portion of its success to the effort that has been spent on the development of global constraints. This paper enhances a state of the art formalization of soft constraints by a notion of global constraints. As a result, formalizations common to the constraint satisfaction community are able to concern specifications of algorithms which are specialized on the propagation of certain global constraints. It turns out, that — on constraint-based optimization — parts of the constraint satisfaction community and people that develop and apply constraint programming tools to real world applications use quite different notions of constraint propagation.

This paper introduces a formalization (and examples) for propagating fuzzy global constraints in a reduced but practicable fashion. This shallow kind of propagation is compared to constraint propagation which is common in formalizations like valued CSP or semi-ring-based constraint optimization referring to backtrack-search and the fixed point of propagation.

**Keywords:** constraint-based optimization, fuzzy constraints, global constraints, constraint propagation.

## 1 Introduction

Most of the success of constraint-based reasoning in real world application comes from the development of global constraints in *constraint logic programming (CLP)*. For this reason, Mark Wallace's challenging talk on ECAI'00 recommended researchers on constraint satisfaction to drop commonly accepted limitations of CSPs — namely concerning the constraints. Driven by real world experiences, this paper tries to follow this advice since the criticized research community can answer with important results: During the past decade, extensions to the well known *constraint satisfaction problem (CSP)* concerning also soft constraints led to expressive and declarative formalisms for the representation of optimization problems. Especially valued CSPs (VCSP) [Schiex *et al.*, 1995] and semi-ring-based constraint problems (SCSP) [Bistarelli *et al.*, 1995] have to be mentioned in this context [Bistarelli *et al.*, 1996, Bistarelli *et al.*, 1997].

This paper tries to bridge this gap between CSP and CLP in the following manner. The next section presents a state-of-the-art formalization of soft constraints that is based on ideas from VCSP and SCSP. As a tribute to the practice in CLP, this formalization concerns also constraint types as a representation for the applicability of specialized algorithms for constraint propagation. Founded on this terminology, the

following section describes two methods for constraint propagation: *Perfect propagation* describes the idea on propagating soft constraints in extensions of the CSP. *Shallow propagation* is inspired by common practice in CLP that eases the development of algorithms for the propagation of global constraints. This claim is illustrated by some simple examples. The next section compares the fixed point of both kinds of constraint propagation. The fixed point of constraint propagation is closely related to arc-consistency, the traditional heart of research on constraint satisfaction. Finally, concluding remarks sum up the results.

## 2 A Formalism for Global Constraints with Fuzzy Extensions

This section starts with a formalization of constraint optimization which follows the state of the art in extending constraint satisfaction. This formalization is then used to describe constraint types which are appropriate to represent applicability of specialized algorithms for constraint propagation.

Some notes on the notation in advance: Throughout the paper,  $X$  denotes a set of variables,  $D$  a set of values used as a domain and  $C$  a set of constraints.  $D^X$  denotes the set of variable/value-pairs  $\{x \leftarrow d \mid x \in X, d \in D\}$  representing the set of all assignments of values from  $D$  to the variables in  $X$ . Let  $A \in D^X$ , then  $A \downarrow X'$  is the projection of  $A$  on the variables  $X \cap X'$  and  $A \downarrow x$  with  $x \in X$  denotes the value assigned by  $A$  to variable  $x$ .

### 2.1 Representation of Optimization by Constraints

Optimization in general may be characterized as finding an assignment to variables that causes minimal costs regarding a set of objective functions. Thus, beside variables, values, and objective functions, general optimization problems consist of a valuation structure that (1) defines the valuations which form the range of the objective functions, (2) provides an ordering of valuations in order to distinguish better from worse valuations, i.e. smaller from larger valuations, and (3) defines a binary operation to join valuations from different objectives into a unique objective function.

**Def. 1** A valuation structure  $S = \langle E, \succ, \oplus \rangle$  consists of a set of valuations  $E$ , a (not necessarily total) preference ordering  $\succ$  of valuations, and a binary operation  $\oplus$  on  $E$  called conjunction (or additive operation), where a  $0_E \in E$  exists with the following properties: (1) All other valuations are larger than  $0_E$  due to  $\succ$ . (2) Conjunction  $\oplus$  is associative, commutative, and closed on  $E$ . (3)  $0_E$  is neutral concerning  $\oplus$ , i.e.  $\forall e \in E : e \oplus 0_E = e$ . (4) For all  $e, e_1, e_2 \in E$ , monotony condition  $e_1 \succeq e_2 \implies (e_1 \oplus e) \succeq (e_2 \oplus e)$  holds true.  $\square$

**Cor. 1.1** From monotony and  $e_1 \succeq 0_E$  follows  $e_1 \oplus e_2 \succeq e_2$  for all  $e_1, e_2 \in E$ .  $\square$

This definition describes quite intuitive demands. The neutral element is the valuation of a perfect solution fulfilling all demands at no costs. Monotony guarantees that

putting the same penalty on two solutions does not reverse preference. The following corollaries present some valuation structures which are quite similar to some examples given by Schiex [Schiex *et al.*, 1995].

**Cor. 1.2** *The structure  $S^\wedge = \langle \{T, F\}, \wedge, \not\rightarrow \rangle$  — where  $T$  and  $F$  are the Boolean truth values,  $\wedge$  denotes Boolean conjunction and  $\not\rightarrow$  the negated implication — is a valuation structure with  $0_{\{T, F\}} = T$ .  $F \not\rightarrow T$  is the only way to satisfy the negated implication. Thus,  $\not\rightarrow$  is a total ordering of the Boolean truth values with  $T$  as minimal element.  $T$  and  $F$  correspond to classical two-valued logics. This structure is also called dichotomic valuation.  $\square$*

**Cor. 1.3** *The structure  $S^+ = \langle \mathbb{R}_0^+, +, > \rangle$  — the set of positive real numbers including the 0 combined with addition and ordered by the natural ordering — is a valuation structure with  $0_{\mathbb{R}_0^+} = 0$ . This valuation structure is obviously especially appropriate to the task of adding costs [Freuder and Wallace, 1992]. Additionally,  $S^{\max} = \langle [0; 1], \max, > \rangle$  is a second valuation structure on real numbers that relates closely to fuzzy sets [Dubois *et al.*, 1993].  $\square$*

**Cor. 1.4** *If the  $S_0 = \langle E_0, \oplus_0, \succ_0 \rangle, \dots, S_n = \langle E_n, \oplus_n, \succ_n \rangle$  are valuation structures then so is the lexicographic combination  $\text{lex}(S_0, \dots, S_n) = \langle E_0 \times \dots \times E_n, \oplus, \succ \rangle$  of these structures where*

*$\langle e_0, \dots, e_n \rangle \oplus \langle e'_0, \dots, e'_n \rangle = \langle e_0 \oplus_0 e'_0, \dots, e_n \oplus_n e'_n \rangle$  and  $\langle e_0, \dots, e_n \rangle \succ \langle e'_0, \dots, e'_n \rangle$  is true iff an  $i$  exists with  $0 \leq i \leq n$  and  $e_i \succ_i e'_i$  and  $e_j = e'_j$  for all  $j$  with  $0 \leq j < i$ . These structures are closely related to constraint hierarchies including inter-hierarchy comparison [Wilson and Borning, 1989].  $\square$*

Defining optimization with such valuations is straight forward.

**Def. 2** *A tuple  $\mathcal{P} = \langle X, D, C, S \rangle$  defines a constraint optimization problem (COP) iff  $X$  is a set of variables,  $D$  is a set of values called initial domain,  $S = \langle E, \succ, \oplus \rangle$  is a valuation structure, and  $C$  is a set of locally defined objectives of the form  $c = \langle X_c, \varphi_c \rangle \in C$  which are called soft constraints.  $X_c \subseteq X$  is called the set of local variables.  $\varphi_c \in (D^{X_c} \rightarrow E)$  is the characteristic function of constraint  $c$ 's extension.*

*An assignment  $A$  to the local variables of constraint  $c$  is called to satisfy the constraint iff  $\varphi_c(A) = 0_E$ . Otherwise,  $A$  is called to violate the constraint.*

*An assignment  $A$  with values from  $D$  to all variables in  $X$  is a solution to  $\mathcal{P}$  iff  $\mathcal{V}_{\mathcal{P}}(A) = \bigoplus_{\langle X_c, \varphi_c \rangle \in C} [\varphi_c(A \downarrow X_c)]$  is minimal due to  $\succ$ .*

*Two problems  $\mathcal{P}_1$  and  $\mathcal{P}_2$  are equivalent iff for all assignments  $A_1, A_2 \in D^X$ :  $\mathcal{V}_{\mathcal{P}_1}(A_1) \succ_1 \mathcal{V}_{\mathcal{P}_1}(A_2)$  iff  $\mathcal{V}_{\mathcal{P}_2}(A_1) \succ_2 \mathcal{V}_{\mathcal{P}_2}(A_2)$ .  $\square$*

This is a very general notion of optimization. Def. 2 covers all problems that concern minimization of partially ordered valuations by a series of objective functions. Nevertheless, the soft constraints in COPs according to Def. 2 are quite similar to hard constraints in standard CSP regarding their structure. They are defined by a set of local variables and an extension. The only difference is that their extension is a fuzzy set rather than a crisp set of assignments to the local variables where the membership of

each tuple is represented by a valuation. Thus, an assignment  $A \in D^{X_c}$  may be considered as a  $\varphi_c(A)$ -member of the fuzzy extension of constraint  $c = \langle X_c, \varphi_c \rangle$ . However, in contrast to the standard notion of fuzzy sets [Zadeh, 1978], this membership is not necessarily required to be a number. Several other notions of — also only partially ordered — memberships are conceivable.

A constraint optimization problem behaves like a standard CSP if the characteristic functions have a two-valued range — e.g. the dichotomic valuation  $S^\wedge$  of Corollary 1.2. On such problems, characteristic function only distinguish exactly one degree of constraint violation.

**Cor. 2.1** *A COP of the form  $\mathcal{P} = \langle X, D, C, S^\wedge \rangle$  has exactly that assignments of values from  $D$  to the variables in  $X$  as solutions that satisfy all constraints in  $C$  as long as such assignments exist.*  $\square$

Thus, hard constraints may be viewed as soft constraints using the valuations  $T$  (satisfaction) and  $F$  (violation). Vice versa, it is often useful to weaken a hard constraint  $c = \langle X_c, \varphi_c \rangle$  whose extension is represented by a characteristic function  $\varphi_c$  of range  $S^\wedge$ . A function  $f_{E,e} \in (\{T, F\} \rightarrow E)$  mapping  $T$  to  $0_E$  and  $F$  to another valuation  $e$  is appropriate to adapt constraint  $c$  to another valuation structure  $S = \langle E, \otimes, \succ \rangle$  using a constraint  $\langle X_c, f_{E,e} \circ \varphi_c \rangle$  instead. Parameter  $e$  can be considered as a measure for the relative importance of the constraint describing the impact of violating the constraint on the quality of a solution.

## 2.2 Using Constraint Types

Constraint-based reasoning owes a large portion of its success to systems like CHIP [Dincbas *et al.*, 1988] and ILOG Solver [Puget, 1994] that integrate powerful inferences on global constraints into backtracking algorithms [Nadel, 1989]. Because of the well-founded semantics of hard constraints [Mackworth, 1992], PROLOG-like languages denote constraints by built-in predicates. Constraint propagation is used to improve unification [Diaz and Codognot, 1993] of these predicates. When using soft constraints, the situation is a bit more complex. Fages presented a characterization of optimization within constraint logic programming [Fages *et al.*, 1996]. However, this characterization makes a distinction between objective functions and predicates/constraints. Predicates and constraints are dichotomic. As usual in standard predicate logics, predicates have a crisp set of models. Thus, constraints in CLP are restricted to have a crisp extension. This means: All constraints use the dichotomic valuation.

In contrast to standard CLP, the previous section allows constraints to produce non-dichotomic valuations. Constraint optimization problems represent also objective functions directly by constraints. Def. 2 of constraint optimization problems suggests to use propagation algorithms on soft constraints analogously to the use of constraint libraries in standard CLP. The class of constraints that can be implemented by the same propagation algorithms is often referred to as *constraint type*. As mentioned above, in CLP this is usually a built-in predicate. The following section gives a more general algebraic definition of a constraint type as a function  $\Phi$  that maps variables and a parameter to the characteristic function of an extension.

**Def. 3** A constraint type using domain  $D$  and valuations from  $E$  is a function  $\Phi$  that maps a parameter  $\sigma$  and a sequence of constraint variables  $\langle x_1, \dots, x_n \rangle$  to a characteristic function on assignments to the variables  $x_1$  to  $x_n$ :  $\Phi(\sigma, \langle x_1, \dots, x_n \rangle) \in (D^{\{x_1, \dots, x_n\}} \rightarrow E)$ .

A constraint language  $\mathcal{L}$  is a set of constraint types. □

A constraint language according to 3 is used to ease representation of constraints in optimization problems. The following definition declares constraint definitions to define soft constraints according to a constraint type. Constraint definitions of this kind have the advantages that (1) occasionally very complex constraints can be defined easily by use of templates from a constraint language and that (2) like in CLP the main building blocks of the problem representation — the constraints — relate directly to constraint types which typically define applicability of procedures for constraint propagation.

**Def. 4** A tuple  $c = \langle \Phi, \sigma, x_1, \dots, x_n \rangle$  is a constraint definition using constraint type  $\Phi$ , iff  $\Phi$  is a constraint type that maps parameter  $\sigma$  and the variables  $\langle x_1, \dots, x_n \rangle$  to a characteristic function  $\varphi = \Phi(\sigma, \langle x_1, \dots, x_n \rangle)$ .

The constraint  $\mathcal{I}(c) = \langle \{x_1, \dots, x_n\}, \varphi \rangle$  is called the interpretation of constraint definition  $c$ . □

Constraint definitions are building blocks of another form to state constraint optimization problems.

**Def. 5** A tuple  $\mathcal{P} = \langle X, D, C, S, \mathcal{L} \rangle$  is a constraint optimization problem using constraint language  $\mathcal{L}$  ( $\mathcal{L}$ -COP) iff  $X$  is a set of variables,  $D$  is a set of values,  $S = \langle E, \oplus, \succ \rangle$  is a valuation structure, and  $C$  is a set of constraint definitions  $c$  with  $\mathcal{I}(c) = \langle X_c, \varphi_c \rangle$  and the following holds true for the interpretation of  $c$ :  $X_c \subseteq X$  and  $\varphi_c \in (D^{X_c} \rightarrow E)$ .

The interpretation of  $\mathcal{P}$  is the COP  $\mathcal{I}(\mathcal{P}) = \langle X, D, \{\mathcal{I}(c) \mid c \in C\}, S \rangle$ .

The solutions of  $\mathcal{P}$  are exactly the solutions of its interpretation  $\mathcal{I}(\mathcal{P})$ . □

To figure out the benefits of constraint types, the remainder of this section presents some examples for constraint types which form the backbone of a real world application described elsewhere [Meyer auf'm Hofe, 1997, Meyer auf'm Hofe, 2001, Meyer auf'm Hofe, 2000c]: Nurse rostering. The following constraint types are essential to these problems.

Several papers on constraint satisfaction focus on crisp and binary constraints since all of these constraints can be propagated testing at most  $|D|^2$  assignments to the local variables<sup>1</sup>. A single constraint type  $\Phi_2$  may be used to capture all binary constraints using the parameters  $ext \in D^{\{x_1, x_2\}}$  to provide the extension of the constraint and  $e \in E$  to represent the valuation resulting from violating the constraint. The resulting characteristic function  $\Phi_2(\langle E, e, ext \rangle, \langle x_1, x_2 \rangle)$  may be subdivided into  $f_{E,e} \circ \varphi_2$ . Let  $A$  be an assignment to the local variables  $x_1$  and  $x_2$ . Then,  $\varphi_2(A) = T$  iff  $A \in ext$  and

<sup>1</sup>Propagation of functional dependencies as in the AC-4 [Van Hentenryck *et al.*, 1992] is a subclass of  $\Phi_2$  that can be propagated enumerating  $O(|D|)$  assignments to the local variables.

$\varphi_2(A) = F$  otherwise. Function  $f_{E,e}$  is used as in section 2.1 to weaken the binary constraint described by  $\varphi_2$  according to the parameters  $E$  and the relative importance of the constraint  $e \in E$ .  $f_{E,e}(T) = 0_E$  and  $f_{E,e}(F) = e$ .

The second constraint type  $\Phi_{atleast}$  is a simple example for a crisp global constraint that is appropriate to constrain the cardinality of some values in assignments to the local variables. Parameter  $\mu \in (D \rightarrow \mathbb{R}^+)$  maps each value from the domain to a penalty to be counted in assignments. Parameter  $b$  is a lower bound on the sum of these penalties. Again, parameters  $E$  and a relative importance  $e \in E$  are used to weaken the constraint. For an arbitrary number of variables,  $\Phi_{atleast}(\langle E, e, \mu, b \rangle, \langle x_1, \dots, x_n \rangle)$  defines a characteristic function  $f_{e,E} \circ \varphi_{atleast}$  with  $\varphi_{atleast}(A) = T$  iff  $b \leq \sum_{i=1}^n \mu(A \downarrow x_i)$ . Otherwise, the result is  $F$ .

Analogously, a constraint type  $\Phi_{atmost}$  is defined for the same parameters where  $\Phi_{atmost}(\langle E, e, \mu, b \rangle, \langle x_1, \dots, x_n \rangle) = f_{e,E} \circ \varphi_{atmost}$  with  $\varphi_{atmost}(A) = T$  iff  $b \geq \sum_{i=1}^n \mu(A \downarrow x_i)$ . Both constraints are one-dimensional simplifications of the global cardinality constraint [Régin, 1996].

The concluding constraint type  $\Phi_{approx}$  is an example for a fuzzy, non-dichotomic extension that uses valuations except  $T$  and  $F$ . Parameters are again a function  $\mu \in (D \rightarrow \mathbb{R}^+)$  returning a number for each value of domain  $D$ . In contrast to  $\Phi_{atleast}$  and  $\Phi_{atmost}$ , this constraint requires the sum of these numbers to approximate a certain goal sum  $g \in \mathbb{R}^+$ . Thus,  $\Phi_{approx}(\langle f_S, \mu, g \rangle, \langle x_1, \dots, x_n \rangle) = f_S \circ \varphi_{approx}$  where  $\varphi_{approx}$  returns a real number as valuation according to:  $\varphi_{approx}(A) = |g - \sum_{i=1}^n \mu(A \downarrow x_i)|$ . Function  $f_S$  is used to adapt this extension to valuations  $S = \langle E, \otimes, \succ \rangle$  beside  $\mathbb{R}$ . This function is required to be monotonic, i.e. for all  $x, y \in \mathbb{R}$  with  $x > y$  follows  $f_S(x) \succ f_S(y)$ . This property is later on needed by the procedure for efficient propagation.

### 3 Propagation of Soft Constraint

The purpose of propagating hard constraints is to infer *a priori* unknown constraints which are implied by the constraint problem. In most applications, the inferred constraints affect only one variable directly. Thus, constraint propagation is used to achieve a kind of *invers consistency* [Freuder and Elfe, 1996, Verfaillie *et al.*, 1999] between the constraints from a set  $C$  and an assignment of domains of the form  $\mathcal{A} = \{x_1 \leftarrow D_1, \dots, x_n \leftarrow D_n\}$  where the  $D_i$  are subsets of the initial domain  $D$ . For all of the various forms of consistency, the implied constraints  $x_i \in D_i$  are implied by the conjunction of the constraints in  $C$ , or, in other words,  $C \cup \bigcup_{i=1}^n [x_i \in D_i]$  is logically equivalent to  $C$ .

Propagation of soft constraints from a COP shall be a generalization of propagating hard constraints. In contrast to the consistent domains derived from hard constraints, consistent domains in COP are equivalent to soft constraints.

**Def. 6** A set  $\mathcal{A} = \{x_1 \leftarrow \mu_1, \dots, x_n \leftarrow \mu_n\}$  is called consistent domain assignment to a COP  $\mathcal{P} = \langle X, D, C, S \rangle$  with  $S = \langle E, \otimes, \succ \rangle$  of valuation  $\mathcal{V}_{\mathcal{A}}(A) = \bigoplus_{i=1}^n [\mu_i(A \downarrow x_i)]$  iff  $x_1, \dots, x_n \in X$ ,  $\mu_1, \dots, \mu_n \in (D \rightarrow E)$ , and  $\mathcal{V}_{\mathcal{P}} \oplus \mathcal{V}_{\mathcal{A}} = \mathcal{V}_{\mathcal{P}}$  holds true for all labelings  $A \in D^X$ .

The domain assignment  $\mathcal{A}^0$ , that assigns domain  $\mu$  with  $\mu(d) = 0_E$  for all values  $d \in D$  to all variables in  $X$ , is called initial domain assignment.  $\square$

Please note, that the representation of domains by assignments allows reuse of notations on projection:  $\{\dots, x \leftarrow \mu, \dots\} \downarrow x = \mu$ .

Definition 6 provides a notion of implication by soft constraints that extends implication of hard constraints.

**Cor. 6.1** Assume that  $\mathcal{P}_{\mathcal{A}} = \langle X, D, C_{\mathcal{A}}, S \rangle$  is the COP with  $C_{\mathcal{A}} = C \cup \bigcup_{i=1}^m [\{\{x_i\}, \mu_i\}]$  that results from adding the constraints from a domain assignment  $\mathcal{A}$  to  $\mathcal{P}$ . Then, obviously,  $\mathcal{V}_{\mathcal{P}_{\mathcal{A}}}(A) = \mathcal{V}_{\mathcal{P}}(A) \oplus \mathcal{V}_{\mathcal{A}}(A)$ . Iff  $\mathcal{A}$  is a consistent domain assignment then  $\mathcal{V}_{\mathcal{P}_{\mathcal{A}}}(A) = \mathcal{V}_{\mathcal{A}}(A)$ .  $\square$

Consistent domain assignments provide additionally an optimistic estimate of the best valuation that can be achieved assigning a certain value to a certain variable (cf. the  $A^*$ -heuristic [Nilsson, 1982]).

**Cor. 6.2** If  $\mathcal{A} = \{\dots, x \leftarrow \mu, \dots\}$  is a consistent domain assignment, then  $\mu(A \downarrow x) \oplus \mathcal{V}_{\mathcal{P}}(A) = \mathcal{V}_{\mathcal{P}}(A) \succeq \mu(A \downarrow x)$  holds true for all assignments  $A$  of values to the variables (according to Corollary 1.1).  $\square$

This section concentrates on procedures for propagating single constraints that guarantee to produce consistent domain assignments<sup>2</sup>. The following sections present two examples for such procedures where the first one is derived from literature on extensions to constraint satisfaction whereas the second procedure is inspired by practice in constraint logic programming.

### 3.1 Constraint Propagation á la Constraint Satisfaction: Perfect Propagation

In the tradition of constraint satisfaction, constraint propagation is often viewed as a two-step procedure, where a step “combination” collects all assignments to a subset of variables that comply with the extension of the currently propagated constraint as well as with the currently known domain assignment. The second step “projection” then generates a new domain assignment that is used for either propagating the next constraint or as final result. The well-known min-max-procedure [Rosenfeld *et al.*, 1976, Snow and Freuder, 1990] generalizes the combination to the valuation structure  $S^{\max}$  from Corollary 1.3: For each assignment to the local variables of the constraints, the maximum valuation by the constraint and the domain assignments is recorded. Finally, the resulting domain assignment maps a valuation to each value that is the minimum of all maximums that have been computed to labelings assigning the value in question. The maximum corresponds to conjunction: A value has to comply with the propagated explicit constraint *and* with the currently known implicit constraints which

<sup>2</sup>Levels of consistency which are stronger than the fixed point of propagation — for instance path invers consistency or neighborhood invers consistency — can be achieved analogously to the procedure with hard constraints [Verfaillie *et al.*, 1999] using the propagation procedures as described in this section.

are represented by the domain assignment. Analogously, the minimum corresponds to a disjunction. Unfortunately, valuation structures according to Def. 1 allow also partial preference orderings. So, the minimum is not guaranteed to be unique. Semiring-based constraints provide a solution to this problem: The preference ordering is thought to be defined by a multiplication operation.

**Def. 7**  $\hat{S} = \langle E, \oplus, \otimes, \succ \rangle$  is an extended valuation structure extending valuation structure  $\langle E, \oplus, \succ \rangle$  iff (1)  $\otimes$  is associative, commutative, and closed on  $E$ , and (2) for all  $e_1, e_2, e \in E$  the following monotony conditions hold true:  $e_1 \succ e_2$  implies  $(e_1 \otimes e) \succeq (e_2 \otimes e)$  and always  $e_1 \succeq (e_1 \otimes e_2)$ .  $\otimes$  is then called disjunction (or multiplication).  $\square$

**Cor. 7.1** For all  $e, e_1, e_2 \in E$ :  $e_1 \succeq e_2$  implies  $(e_1 \otimes e_2) = e_2$ .  $0_E$  is, thus, absorbing element of  $\otimes$ . Idempotency of  $\otimes$ , i.e.  $(e \otimes e) = e$ , follows from  $e \succeq e$ . Furthermore,  $(e \oplus e_1) \otimes (e \oplus e_2) \succeq (e \oplus (e_1 \otimes e_2)) \otimes (e \oplus (e_1 \otimes e_2)) = e \oplus (e_1 \otimes e_2)$ .  $\square$

**Cor. 7.2** If  $\langle E, \oplus, \succ \rangle$  is a valuation structure and  $\succ$  is total ordering on  $E$  then  $\langle W, \oplus, \min_{\succ}, \succ \rangle$  is an extended valuation structure with the minimum  $\min_{\succ}$  according to ordering  $\succ$  as disjunction. The valuation structures of the corollaries 1.2 to 1.3 use total preference orderings and lexicographic valuations of Corollary 1.4 are totally ordered if all combined valuation structures are so.  $\square$

The availability of extended valuation structures enables the use of disjunction in constraint propagation. So, the min-max-procedure can be generalized as follows.

**Def. 8** Let  $\mathcal{A}$  be a domain assignment to all variables of a COP  $\mathcal{P} = \langle X, D, C, S \rangle$  with  $\hat{S} = \langle E, \oplus, \otimes, \succ \rangle$  extending  $S$ . Then, function  $pprop$  mapping a constraint  $c = \langle X_c, \varphi_c \rangle$  and one domain assignment to another domain assignment is a perfect propagation iff (1) for all  $x \notin X_c$ :  $(pprop(c, \mathcal{A}) \downarrow x) = \mathcal{A} \downarrow x$ , and (2) for all  $x \in X_c, d \in D$ :  $(pprop(c, \mathcal{A}) \downarrow x)(d) = \bigotimes_{A \in D^{X_c}, A \downarrow x=d} [\varphi_c(A) \oplus \mathcal{V}_{A \downarrow X_c}(A)]$ .

Notations:

$$\begin{aligned} pprop(c) &\equiv pprop(c, \mathcal{A}^0). \\ pprop(c_1, c_2) &\equiv pprop(c_1, pprop(c_2)) \text{ and so on.} \end{aligned}$$

$\square$

Propagation does only affect the domains which are assigned to the local variables of the constraint. Expression  $(pprop(c, \mathcal{A}) \downarrow x)(d)$  denotes the valuation that the domain of variable  $x$  assigns after propagation to value  $d$ . This valuation reflects the valuation of the best assignment  $A$  to the local variables that is valued by  $\varphi_c(A)$  according to the currently propagated constraint and by  $\mathcal{V}_{A \downarrow X_c}(A)$  due to the currently assigned domains. If the dichotomic valuation structure is used, perfect propagation rules out all values without a supporting assignment to the local variables casting a valuation  $F$ . This is exactly like propagation in standard CSP. Perfect propagation has additionally the following characteristics which are well known from VCSP and SCSP.

**Thm. 9** Assume COP  $\mathcal{P} = \langle X, D, C, S \rangle$  where  $\hat{S} = \langle E, \oplus, \otimes, \succ \rangle$  extends  $S$ . If  $\oplus$  is idempotent. Then,  $pprop(c_1, \dots, c_m)$  is a consistent domain assignment for all  $c_1, \dots, c_m \in C$  and all disjunctions  $\otimes$ .

*Proof: Assumption 1: If domain assignment  $A$  is consistent and  $c$  is a constraint, then  $\text{pprop}(c, A)$  is consistent for any assignment  $A$ . Proof: For any assignment  $A$  concludes  $\mathcal{V}_A(A) \oplus \mathcal{V}_{\text{pprop}(c, A)}(A) \preceq^{(1)} \mathcal{V}_{\mathcal{P}}(A) \oplus \varphi_c(A \downarrow X_c) \oplus \mathcal{V}_{A \downarrow X_c}(A) \preceq^{(2)} (\mathcal{V}_{\mathcal{P}}(A) \oplus \varphi_c(A \downarrow X_c)) \oplus (\mathcal{V}_{A \downarrow X_c}(A) \oplus \mathcal{V}_A) \preceq^{(3)} \mathcal{V}_{\mathcal{P}}(A) \oplus \mathcal{V}_A =^{(4)} \mathcal{V}_{\mathcal{P}}(A)$ . The first inference concludes from the definition of propagation. The second exploits monotony and associativity of conjunction. The third uses idempotency, and the final conclusion is justified by consistency of  $A$ . Since  $\mathcal{V}_{\mathcal{P}}(A) \oplus \text{pprop}(c, A) \succ \mathcal{V}_{\mathcal{P}}(A)$  concludes from monotony,  $\mathcal{V}_{\mathcal{P}}(A) \oplus \text{pprop}(c, A) = \mathcal{V}_{\mathcal{P}}(A)$ .*

*Assumption 2: The theorem follows from assumption 1. Proof by recursion over the propagated constraints.  $\square$*

On a first glance, Thm. 9 does not seem to be too valuable since it requires the used conjunction to be idempotent. This precondition avoids situations in which the same conflict is counted twice. Fortunately, non-idempotent valuation structures can be translated efficiently into idempotent ones without changing the nature of the constraint problem to be solved. The following definition describes a method for generating idempotent valuation structures by the addition of bookmarking information that separates valuations referring to different constraints.

**Thm. 10** *Let  $\mathcal{P} = \langle X, D, C, S \rangle$  be a COP with valuation structure  $S = \langle E, \oplus, \succ \rangle$  where  $\succ$  is a total ordering and  $E^C$  denotes assignments of valuations to constraints. Furthermore, let  $\cup_{\min}, \cup_{\max}$  represent two binary operations on  $E^C$  with  $(e_1 \cup_{\min} e_2) \downarrow c = \min_{\succ} \{e_1 \downarrow c, e_2 \downarrow c\}$  and  $(e_1 \cup_{\max} e_2) \downarrow c = \max_{\succ} \{e_1 \downarrow c, e_2 \downarrow c\}$  where  $e_1, e_2 \in E^C$  and  $c \in C$ . Additionally,  $\succ'$  is the partial ordering with  $e_1 \succ' e_2$  iff  $\bigoplus_{c \in C} [e_1 \downarrow c] \succ \bigoplus_{c \in C} [e_2 \downarrow c]$ . Then*

*(1)  $S' = \langle E^C, \cup_{\max}, \cup_{\min}, \succ' \rangle$  is an extended valuation structure with idempotent conjunction  $\cup_{\max}$ .*

*(2)  $\mathcal{P}' = \langle X, D, C', S' \rangle$  with  $C' = \{\langle X_c, f_c \circ \varphi_c \rangle \mid c = \langle X_c, \varphi_c \rangle \in C\}$  — where  $f_c$  converts a valuation from  $E$  to an assignment from  $E^C$  with  $f_c(e) \downarrow c = e$  and  $f_c(e) \downarrow c' = 0_E$  for all  $c' \in C$  with  $c' \neq c$  — is equivalent to  $\mathcal{P}$ .*

*Proof: Assumption (1) follows straight from the definitions. Assumption (2): Both problems are equivalent since for each equivalent  $\langle X_c, f_c \circ \varphi_c \rangle$  in  $\mathcal{P}'$  of a constraint  $\langle X_c, \varphi_c \rangle$  in  $\mathcal{P}$ , only constraint  $c$  gets a valuation larger than  $0_E$  assigned. This is exactly the valuation according to  $\varphi_c$ .  $\square$*

This transformation turns a problem with a non-idempotent conjunction and a totally ordered preference into a problem with an idempotent conjunction and a partially ordered preference. More complex transformations can deal with partially orderings as preference<sup>3</sup>. Thus, the applicability of Thm. 9 can be extended at the cost of introducing a much larger set of valuations that is only partially ordered by the preference.

Consequently, this section introduced a formalization of a generally applicable method for constraint propagation that roughly follows the idea of soft constraints in

<sup>3</sup>The idea is to store for each constraint a set of valuations. The conjunction collects all maximal valuations. The disjunction collects all minimal valuations. Preference is according to the conjunction of the disjunction of valuations to each constraint. More complex transformation that record conflicts for combinations of conflicting constraints may even be used to implement assumption based *truth maintenance systems* (TMS) as COPs [Meyer auf'm Hofe, 2000b].

VCSP and SCSP. The major characteristic of this kind of propagation is that not only the explicit constraints from the problem description but also all inferred constraints — which are represented by consistent domain assignments — are soft constraints. Thus, this notion of constraint propagation is very accurate since all information is used that is available from the domain assignments. This property is the justification for calling this method of propagation perfect. However, perfect propagation always implies the problem of combining valuations from the propagated constraints and from derived information which are both soft. A later section will turn out, that this problem disables many algorithms which are used in state of the art CLP applications to implement global constraints. For the moment, this brief remark shall suffice to motivate another variant of constraint propagation that follows the practice in CLP.

### 3.2 Adopting Ideas From Constraint Logic Programming: Shallow Propagation

Constraint logic programming is typically based on backtracking algorithms like *branch-and-bound* (BB). The algorithm completes a partial assignment assigning values to yet unlabeled variables. A domain assignment records consistency with this partial assignment. Valuations of the domain assignment can be used to assign the most promising values first and to neglect values which do not promise improvements of an upper bound on the valuations of acceptable solutions.

---

**Algorithm 1**  $BB(\mathcal{P} = \langle X, D, C, S \rangle, \beta, A, \mathcal{A})$

---

```

1: if  $A$  labels all vars then return  $A$ , end if
2:  $A' \leftarrow \emptyset$ , choose a not yet labeled  $x \in X$ 
3: for all  $d \in D$  with  $\mathcal{A} \downarrow x(d) \not\prec \beta$  do
4:    $A' \leftarrow \mathcal{A} \downarrow (X \setminus \{x\}) \cup \{x \leftarrow \mu_{\beta,d}\}$ .
5:   Increase all valuations casted by  $A'$  by  $\mathcal{V}_{\mathcal{P}}(A \cup \{x \leftarrow d\})$ .
6:    $A'' \leftarrow BB_{EPROP}(\mathcal{P}, \beta, A \cup \{x \leftarrow d\}, prop(\beta))$ .
7:   if  $A'' \neq \emptyset$  then  $A' \leftarrow A'', \beta \leftarrow \mathcal{V}_{\mathcal{P}}(A'')$  end if.
8: end for.
9: return  $A'$ 

```

---

*Assume: Conjunction is idempotent.  $\beta$  is an upper bound on the valuation of acceptable solutions.  $A$  and  $\mathcal{A}$  are initially empty.  $\mu_{\beta,d}(d) = 0_E$  and  $\mu_{\beta,d}(d') = \beta$  for all  $d' \neq d$ . This function describes a domain after assigning value  $d$ .  $prop$  is the function that propagates at least the constraints with  $x$  as local variables.*

---

Algorithm 1 presents a simple version of the BB that uses constraint propagation in order to obtain a domain assignment  $A'$  reflecting consequences of the current partial assignment  $A \cup \{x \leftarrow d\}$ . This domain assignment is used to keep the search tree small since only that values are concerned for branching in line 3 that promise solutions better than bound  $\beta$ . The perfect propagation of the previous section is appropriate to implement propagation  $prop$  in the algorithm. However, this section presents another kind of *shallow propagation* that enables use of propagation algorithms which are specialized on certain types of constraints. As the examples of the next section will show, many of these algorithms cannot deal with soft domains like this is demanded by the perfect propagation. Several propagation procedures can only distinguish between values that can be a part of a solution or which have been pruned by previous inferences.

Algorithm BB uses bound  $\beta$  to make this distinction. Analogously, shallow propagation receives a bound  $\beta$  as an argument to distinguish admissible from non-admissible values.

**Def. 11** Let  $\mathcal{A}$  be a domain assignment to all variables of a COP  $\mathcal{P} = \langle X, D, C, S \rangle$  with  $\hat{S} = \langle E, \oplus, \otimes, \succ \rangle$  extending  $S$ . Then, function *sprop* mapping a bound  $\beta \in E$ , a constraint  $c = \langle X_c, \varphi_c \rangle$ , and a domain assignment to another domain assignment is a shallow propagation iff

- (1) for all  $x \notin X_c$ :  $(sprop(\beta, c, \mathcal{A}) \downarrow x) = \mathcal{A} \downarrow x$ , and  
(2) for all  $x \in X_c, d \in D$ :  $(sprop(\beta, c, \mathcal{A}) \downarrow x)(d) =$

$$(\mathcal{A} \downarrow x)(d) \oplus \left( \beta \otimes \bigotimes_{A \in D^{X_c}, \mathcal{A} \downarrow x = d, \mathcal{V}_{\mathcal{A}}(A) \not\prec \beta} [\varphi_c(A)] \right).$$

*Additional notations:*

$$\begin{aligned} sprop(\beta, c) &\equiv sprop(\beta, c, \mathcal{A}^0). \\ sprop(\beta, c_1, c_2) &\equiv sprop(\beta, c_1, sprop(\beta, c_2)). \\ sprop(\beta_1, c_1, \beta_2, c_2) &\equiv sprop(\beta_1, c_1, sprop(\beta_2, c_2)) \text{ and so on.} \end{aligned}$$

□

Thus, shallow propagation increases the valuation of domain assignments by the best valuation that can be achieved referring to admissible values. Admissible values have a valuation below bound  $\beta$ . Referring to global constraints, the most remarkable property of this definition is that, in contrast to the perfect propagation, the disjunction covers only valuations of the characteristic function. The implementation of this disjunction causes the complexity of constraint propagation.

The rest of this section proves some interesting characteristics of shallow propagation.

**Thm. 12** For all constraints  $c = \langle X_c, \varphi_c \rangle$ , all domain assignments  $\mathcal{A}$ , all upper bounds  $\beta$  of valuations casted by  $\mathcal{A}$  and the characteristic function of  $c$ , and all values  $d \in D$ :  $(pprop(c, \mathcal{A}) \downarrow x)(d) \succeq (sprop(\beta, c, \mathcal{A}) \downarrow x)(d)$ . This implies that shallow propagation of consistent domain assignments is also consistent.

*Proof:*  $\bigotimes_{A \in D^{X_c}, \mathcal{A} \downarrow x = d} [\varphi_c(A) \oplus \mathcal{V}_{\mathcal{A} \downarrow X_c}(A)] =^{(1)} \bigotimes_{A \in D^{X_c}, \mathcal{A} \downarrow x = d} [\varphi_c(A) \oplus (\mathcal{A} \downarrow x)(d) \oplus \mathcal{V}_{\mathcal{A} \downarrow X_c \setminus \{x\}}(A)] \succeq^{(2)} \bigotimes_{A \in D^{X_c}, \mathcal{A} \downarrow x = d, \mathcal{V}_{\mathcal{A}}(A) \not\prec \beta} [\varphi_c(A) \oplus (\mathcal{A} \downarrow x)(d)] \succeq^{(3)} (\mathcal{A} \downarrow x)(d) \oplus \bigotimes_{A \in D^{X_c}, \mathcal{A} \downarrow x = d, \mathcal{V}_{\mathcal{A}}(A) \not\prec \beta} [\varphi_c(A)]$ . *Inference (1) simply separates parts of the valuation. Inference (2) drops some parts of the disjunction which would result into a valuation larger than  $\beta$ . Additionally, this inference makes all disjunctions smaller. Finally, inference (3) uses Corollary 7.1.* □

Thm. 12 ensures also a not yet proven property of the perfect propagation: Both kinds of propagation infer indeed stronger constraints than they get as arguments.

**Cor. 12.1** Obviously, all valuations casted by the result of shallow propagation are larger or equal than the valuations casted by the consistent domain assignment of the argument. Thm. 12 ensures that this is also true for the perfect propagation. □

Additionally, the following theorem shows that both, perfect and shallow propagation, produce the same result if used in conjunction with a dichotomic valuation structure, the emulation of a standard CSP. Thus, the theorem justifies the idea that both propagation procedures are generalizations of constraint propagation in CSPs.

**Thm. 13** *Assume COPs of the form  $\mathcal{P} = \langle X, D, C, S^\wedge \rangle$ . Then,  $sprop(c, \mathcal{A}) = sprop(F, c, \mathcal{A})$  for all constraints  $c \in C$  and domain assignments  $\mathcal{A}$ .*

*Proof:* Assume that  $c = \langle X_c, \varphi_c \rangle$ . The following is to be shown for all variables  $x \in X_c$  and values  $d \in D$ :  $\bigvee_{A \in D^{X_c}, A \downarrow x = d} [\varphi_c(A) \wedge \mathcal{V}_{\mathcal{A} \downarrow X_c}(A)] = (\mathcal{A} \downarrow x)(A \downarrow x) \wedge \bigvee_{A \in D^{X_c}, A \downarrow x = d, \mathcal{V}_{\mathcal{A}}(A) = T} [\varphi_c(A)]$ . Assume first, that an admissible assignment  $A$  with  $\mathcal{V}_{\mathcal{A} \downarrow X_c}(A) = T$  does exist. Then, both propagations result into a  $T$ . Otherwise, both result into  $F$ .  $\square$

Thus, this section introduced a second kind of propagation that produces consistent domain assignments. Back to Algorithm 1, this shallow kind of propagation is also appropriate to implement propagation *sprop*. In contrast to the perfect propagation, shallow propagation requires the current bound  $\beta$  as an additional argument. While backtracking search, shallow propagation adds the best valuations to the domain assignments, that is justified by a still admissible assignment to the local variables of a constraint. The next section turns out, that efficient constraint types are much easier to implement for shallow propagation than for perfect propagation.

### 3.3 Variants of Constraint Propagation and Constraint Types

Propagation procedures for special constraint types usually follow the basic idea of constraint propagation in AC-7 [Freuder, 1995]: If an assignment of a value to a variable is not yet proven to be consistent then look for a support of this assignment extending it to an assignment to all local variables. Specialized algorithms can exploit known properties of the constraint types. Thus, search for a support can be guided by knowledge on the extension of the propagated constraints. Unfortunately, most of these strategies work only with shallow propagation, since perfect propagation takes also all valuations casted by the provided domain assignment into account. This section points out the differences between shallow and perfect propagation providing very simple algorithms for shallow propagation of the constraint types  $\Phi_{atleast}$  and  $\Phi_{approx}$  (cf. Section 2.2).

A constraint definition  $c = \langle \Phi_{atleast}, \langle E, e, \mu, b \rangle, x_1, \dots, x_n \rangle$  requires an assignment  $A$  to the local variables to observe  $b \leq \sum_{i=1}^n \mu(A \downarrow x_i)$  to be satisfied. Shallow propagation  $sprop(\beta, \mathcal{I}(c), \mathcal{A})$  can use any assignment to the local variables in order to generate support as long as the valuation of this assignment is not larger than the bound  $\beta$ . The characteristic functions produced by this constraint type suggest to use the admissible assignments with maximal  $\mu$  as support for an assignment  $x_i \leftarrow d$ . If even the help of the assignments with maximal  $\mu$  does not suffice to get a sum above  $b$  then  $x_i \leftarrow d$  is guaranteed to be without any support and the domain assignment resulting from propagation has to cast a valuation  $(\mathcal{A} \downarrow x_i)(d) \oplus e$ .

---

**Algorithm 2**  $sprop_{atleast}(\beta, \langle E, e, \mu, b \rangle, x_1, \dots, x_n)$ 

---

```
1: for all  $i$  from 1 to  $n$  do
2:    $b_i^{\max} \leftarrow \max\{\mu(d) \mid \mathcal{A} \downarrow x_i(d) \neq \beta\}$ .
3: end for
4:  $b_{\Sigma}^{\max} \leftarrow \sum_{i=1}^n b_i$ .
5: for all  $i$  from 1 to  $n$  and  $d \in D$  do
6:   if  $\mu(d) + b_{\Sigma}^{\max} - b_i^{\max} < b$  then  $\mathcal{A} \downarrow x_i(d) \leftarrow \mathcal{A} \downarrow x_i(d) \oplus e$  end if.
7: end for
8: return  $\mathcal{A}$ 
```

---

Shallow propagation of these constraints (cf. Alg. 2) needs  $O(n \cdot |D|)$  queries to domain assignment  $\mathcal{A}$ . This number can be considered as a good measure for the complexity of the algorithm. Line 2 determines the values which can serve as support. In contrast to shallow propagation, the perfect propagation requires a support that is optimized with respect to the valuations casted by the domain assignment as well as with reference to the characteristic function of the currently propagated constraint. Implementations of the search for such a support are not that simple and probably not that efficient.

Constraint definition  $\langle \Phi_{approx}, \langle f, \mu, g \rangle, x_1, \dots, x_n \rangle$  is an example of a constraint of a fuzzy extension: The sum of results from  $\mu$  is preferred to come as near as possible to  $g$ . The algorithm for shallow propagation of these constraints (cf. Alg. 3) is an extension of the *atleast*-propagation: Instead of only acquiring support of a maximal sum of  $\mu$ s, this algorithm records also support for a minimal sum of  $\mu$ s. The procedure increases the valuation of a value if either the maximal support is smaller than goal  $g$  or the minimal support is larger than  $g$ .

---

**Algorithm 3**  $sprop_{approx}(\beta, \langle f, \mu, g \rangle, x_1, \dots, x_n)$ 

---

```
1: for all  $i$  from 1 to  $n$  do
2:    $g_i^{\max} \leftarrow \max\{\mu(d) \mid \mathcal{A} \downarrow x_i(d) \neq \beta\}$ .  $g_i^{\min} \leftarrow \min\{\mu(d) \mid \mathcal{A} \downarrow x_i(d) \neq \beta\}$ .
3: end for
4: for all  $i$  from 1 to  $n$  and  $d \in D$  do
5:    $g_{\Sigma}^{\max} \leftarrow \mu(d) - g_i^{\max} + \sum_{j=1}^n g_j^{\max}$ .  $g_{\Sigma}^{\min} \leftarrow \mu(d) - g_i^{\min} + \sum_{j=1}^n g_j^{\min}$ .
6:   if  $g_{\Sigma}^{\max} < g$  then  $\mathcal{A} \downarrow x_i(d) \leftarrow \mathcal{A} \downarrow x_i(d) \oplus f(|g - g_{\Sigma}^{\max}|)$  end if.
7:   if  $g_{\Sigma}^{\min} > g$  then  $\mathcal{A} \downarrow x_i(d) \leftarrow \mathcal{A} \downarrow x_i(d) \oplus f(|g_{\Sigma}^{\min} - g|)$  end if.
8: end for
9: return  $\mathcal{A}$ 
```

---

The complexity of this algorithm is equal to the *atleast*-propagation:  $O(n \cdot |D|)$ . This example shows, that strategies for optimized support by fuzzy extensions can be quite simple as long as propagation is not required to take also valuations into account that are casted by the current domain assignment.

## 4 The Fixed Point of Propagation

The previous sections addressed only one of the standard algorithms which are usually used in conjunction with constraint processing: Backtracking search. This section adds some notes concerning the fixed point of propagation. This paper concentrates

on the fixed point of constraint propagation and refrains from defining a kind of arc-consistency for COPs although both issues are closely related.

**Thm. 14** *Both, perfect and shallow propagation have a unique fixed point, i.e. propagation of any fair sequence of constraints of infinite length results into the same domain assignment (which is probably different for perfect and for shallow propagation).*

*Proof:* This is rather an overview over the proof: Assume two sequences  $c_1, \dots, c_n$  and  $c'_1, \dots, c'_m$  of constraints, whose propagation ends up with a fixed point. Any further propagation of a constraint fails to change the domain assignment. Furthermore, both kinds of propagations return a domain assignment casting larger valuations if they receive a domain assignment that casts larger valuations. So,  $pprop(c_n, \dots, c_1) = pprop(c'_m, \dots, c'_1, c_n, \dots, c_1) \succeq pprop(c'_m, \dots, c'_1)$  and  $sprop(\beta, c_n, \dots, c_1) = sprop(\beta, c'_m, \dots, c'_1, c_n, \dots, c_1) \succeq sprop(\beta, c'_m, \dots, c'_1)$ . The other direction can be proved analogously. Hence, both fixed points are equal.  $\square$

However, since perfect propagation leads to stronger results (cf. Thm. 12), the fixed point of perfect propagation is likely to represent stronger inferences than the fixed point of the shallow propagation. Thm. 13 presented the exception of dichotomic valuations, where both kinds of propagation are equivalent. Thus, in this situation also their fixed points are equal.

Thm. 14 justifies the introduction of new notations to denote the unique fixed point of propagating constraints from set  $C$  with  $pprop(C)$  or  $sprop(\beta, C)$ , respectively. Since the shallow propagation has a bound as second argument, an additional expression  $sprop(\beta_1, C_1, \beta_2, C_2)$  is useful to denote computation of a fixed point of propagating constraints from  $C_1$  with bound  $\beta_1$  on the fixed point of propagating constraints from  $C_2$  with bound  $\beta_2$ .

For the most prominent kind of soft constraints in the context of arc-consistency — prioritized or flexible constraints [Rosenfeld *et al.*, 1976, Snow and Freuder, 1990, Dubois *et al.*, 1993] — it is possible to compute the fixed point of perfect propagation by an algorithm using shallow propagation. In the terminology of this paper, these problems can be seen as COPs using valuation structure  $S^{\max}$  from Cor. 1.3 and soft but crisp constraints. As pointed out in Section 2.1, such a constraint assigns the same valuation to all assignments that violate the constraint. Assume, that  $e_1 < \dots < e_n$  is the sequence of valuations that can result from violating a single constraint. The set  $C$  of all constraints can be grouped into the constraint sets  $C_1, \dots, C_n$  in such a way that  $c \in C$  is also in  $C_i$  iff  $c$  returns a valuation larger or equal than  $e_i$  on constraint violations. Consequently,  $C_1$  is equal to  $C$ . Then, the following holds true:  $pprop(C) = sprop(e_1, C_1, \dots, e_n, C_n)$ . The following text presents rather an explanation of this claim than a proof since this has been proven elsewhere [Meyer auf'm Hofe, 2000b].

$sprop(e_1, C_1, \dots, e_n, C_n)$  describes a procedure that computes subsequently the fixed point of shallow propagation on a growing set of constraints with a decreasing bound.  $sprop(e_n, C_n)$  assigns valuation  $e_n$  to all assignments  $x \leftarrow d$  which are inconsistent with the constraints in  $C_n$ . All these assignments will be neglected by further propagations with smaller bound. Thus,  $sprop(e_{n-1}, C_{n-1}, e_n, C_n)$  assigns valuation

$e_{n-1}$  to all assignments  $x \leftarrow d$  that are consistent with constraint set  $C_n$  but inconsistent with superset  $C_{n-1}$ , and so on. Finally,  $sprop(e_1, C_1, \dots, e_n, C_n)$  leaves valuation 0 to all the assignments that comply with all constraints  $C_1 = C$  whereas  $e_1$  gets assigned to all  $x \leftarrow d$  that comply with  $C_2$  but violate  $C_1$ . This is exactly the effect that results from  $pprop(C)$ . However, the procedure using shallow propagation may propagate global constraints efficiently.

## 5 Conclusion

The following contributions of this paper try to bridge the gaps between optimization in *constraint logic programming (CLP)* and *constraint satisfaction (CSP)*. *Constraint optimization problems (COP)* provide a formalization of soft constraints that is comparable to the state of the art in constraint satisfaction: Valued CSP [Schiex *et al.*, 1995] and semi-ring-based CSP [Bistarelli *et al.*, 1995]. COPs know additionally about constraint types that serve as representations for the applicability of specialized algorithms for constraint propagation. Perfect and shallow propagation exemplify two distinguished views on how to propagate soft constraints. Shallow propagation eases the development of algorithms for the propagation of global constraints. This has been illustrated by simple examples from a commercially available system on nurse rostering [Meyer auf'm Hofe, 2000c]. Finally, it has been shown that both kinds of propagation have a unique fixed point. On special problems, shallow propagation can be used to emulate the results of perfect propagation.

The relation of perfect and shallow propagation has also been investigated in related work with respect to constraint problems whose constraint graph is a hyper-tree [Meyer auf'm Hofe, 2000a].

In conclusion, this paper may be seen as a first step to integrate results from the field of CLP into research on constraint satisfaction. However, research on CLP has also some things to learn from extensions of the CSP — for instance the foundations of programming in multi-valued logics.

## References

- [Bistarelli *et al.*, 1995] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. Constraint solving over semirings. In Chris Mellish, editor, *IJCAI-95. Proceedings, 14th International Joint Conference on Artificial Intelligence*, pages 624–630. Morgan Kaufmann, 1995.
- [Bistarelli *et al.*, 1996] Stefano Bistarelli, Hélène Fargier, Ugo Montanari, Francesca Rossi, Thomas Schiex, and Gérard Verfaillie. Semiring-based CSPs and valued CSPs: Basic properties and comparisons. In M. Jampel, editor, *Over-Constrained Systems*. Springer Verlag, 1996.
- [Bistarelli *et al.*, 1997] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. Semiring-based constraint satisfaction and optimization. *Journal of the ACM*, 44(2):210–236, 1997.
- [Diaz and Codognet, 1993] D. Diaz and P. Codognet. A minimal extension of the wam for  $\text{c1p}(\text{fd})$ . In *Proceedings of the International Conference on Logic Programming*, pages 774–790, Budapest, Hungary, 1993.
- [Dincbas *et al.*, 1988] M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf, and F. Berthier. The constraint logic programming language CHIP. In *Proceedings of the international conference on fifth generation computer systems*, 1988.

- [Dubois *et al.*, 1993] Didier Dubois, Hélène Fargier, and Henri Prade. The calculus of fuzzy restrictions as a basis for flexible constraint satisfaction. In *Proc. of the 2nd IEEE International Conference on Fuzzy Systems*, pages 1131–1136, San Francisco, CA, 1993.
- [Fages *et al.*, 1996] François Fages, Julian Fowler, and Thierry Sola. Handling preferences in constraint logic programming with relational optimization. In M. Jampel, editor, *Over-Constrained Systems*. Springer Verlag, 1996.
- [Freuder and Elfe, 1996] Eugene C. Freuder and Charles D. Elfe. Neighbourhood inverse consistency preprocessing. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 202–208, Portland, Oregon, 1996. AAAI Press/ The MIT Press.
- [Freuder and Wallace, 1992] Eugene C. Freuder and Richard J. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 58:21–70, 1992.
- [Freuder, 1995] Eugene C. Freuder. Using metalevel constraint knowledge to reduce constraint checking. In Manfred Meyer, editor, *Constraint Processing — Selected Papers*, volume 923 of *LNCS*, chapter 10, pages 171–184. Springer, 1995.
- [Mackworth, 1992] Alan K. Mackworth. The logic of constraint satisfaction. *Artificial Intelligence*, 58:3–20, 1992.
- [Meyer auf'm Hofe, 1997] Harald Meyer auf'm Hofe. ConPlan/ SIEDAplan: Personnel assignment as a problem of hierarchical constraint satisfaction. In *PACT-97: Proceedings of the Third International Conference on the Practical Application of Constraint Technology*, pages 257–272, Practical Application Expo, London, April 1997.
- [Meyer auf'm Hofe, 2000a] Harald Meyer auf'm Hofe. Benefits and problems of using cycle-cutset within iterative improvement algorithms. In *ECAI-2000 Workshop: PuK-2000 — Planen und Konfigurieren*, Berlin, August 2000.
- [Meyer auf'm Hofe, 2000b] Harald Meyer auf'm Hofe. *Kombinatorische Optimierung mit Constraintverfahren — Problemlösung ohne anwendungsspezifische Suchstrategien*, volume 242 of *DISKI – Dissertationen zur Künstlichen Intelligenz*. Infix, akademische Verlagsgesellschaft, 2000. ISBN 3-89838-242-7.
- [Meyer auf'm Hofe, 2000c] Harald Meyer auf'm Hofe. Solving rostering tasks as constraint optimization. In *PATAT-2000: Practical Applications and Theory of Automated Time-Tabling*, pages 280–298, Konstanz, August 2000.
- [Meyer auf'm Hofe, 2001] Harald Meyer auf'm Hofe. Nurse rostering as constraint satisfaction with fuzzy constraints and inferred control strategies. In Eugene C. Freuder and Rick J. Wallace, editors, *Constraint Programming and Large Scale Optimisation Problems*, DIMACS Volume Series, pages 67–99. American Mathematical Society, 2001. DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, Vol. 57, ISSN: 1052-1798.
- [Nadel, 1989] B. A. Nadel. Constraint satisfaction algorithms. *Computational Intelligence*, 5:188–224, 1989.
- [Nilsson, 1982] Nils Nilsson. *Principles of Artificial Intelligence*, chapter 3.2. Symbolic Computation. Springer, Berlin, 1982.
- [Puget, 1994] Jean-Francois Puget. A C++ implementation of CLP. In *Proceedings of the Second Singapore International Conference on Intelligent Systems*, Singapore, 1994.
- [Régis, 1996] Jean-Charles Régis. Generalized arc consistency for global cardinality constraints. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 209–215, Portland, Oregon, 1996. AAAI Press/ The MIT Press.
- [Rosenfeld *et al.*, 1976] A. Rosenfeld, R. A. Hummel, and S. W. Zucker. Scene labeling by relaxation operations. *IEEE Trans. on Systems, Man, and Cybernetics*, 6(6):173–184, 1976.
- [Schiex *et al.*, 1995] Thomas Schiex, Hélène Fargier, and Gérard Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. In Chris Mellish, editor, *IJCAI-95. Proceedings, 14th International Joint Conference on Artificial Intelligence*, pages 631–637. Morgan Kaufmann, 1995.
- [Snow and Freuder, 1990] Paul Snow and Eugene C. Freuder. Improved relaxation and search methods for approximate constraint satisfaction with a maximin criterion. In *Proc. of the 8<sup>th</sup> biennial conf. of the canadian society for comput. studies of intelligence*, pages 227–230, May 1990.

- [Van Hentenryck *et al.*, 1992] Pascal Van Hentenryck, Yves Deville, and Choh-Man Teng. A generic arc-consistency algorithm and its specializations. *Artificial Intelligence*, 57:291–321, 1992.
- [Verfaillie *et al.*, 1999] Gérard Verfaillie, David Martinez, and Christian Bessière. A generic customizable framework for inverse local consistency. In *AAAI-99: Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 169–174, Orlando, FL, 1999.
- [Wilson and Borning, 1989] Molly Wilson and Alan Borning. Extending hierarchical constraint logic programming: Nonmonotonicity and inter-hierarchical comparison. In *Proc. of the North American Conference on Logic Programming*, pages 3–19, Cleveland, Ohio, 1989.
- [Zadeh, 1978] Lofti A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1(2):3–28, 1978.