A Logic of Partially Satisfied Constraints

Nic Wilson

Cork Constraint Computation Centre
Department of Computer Science
University College Cork, Ireland
nicwilson@eircom.net

Abstract. Soft constraints are recognised as being important for many constraints applications. These include (a) over-constrained problems, where we cannot satisfy all the constraints, (b) situations where a constraint can be partially satisfied, so that there are degrees of satisfaction, and (c) where the identity of a constraint is uncertain, so that it can be uncertain whether a constraint is satisfied or not by a tuple.
This paper takes a logic-based approach to partially satisfied constraints, for situations where a tuple can be assigned partially ordered degrees of satisfaction for a constraint. A semantics is defined, which determines which constraints can be inferred from a set of constraints, and a sound and complete proof theory is given. This formalism is closely related to idempotent semiring-based CSPs and lattice-valued possibilistic logic.

1 Introduction

Soft constraints are recognised as being important for many constraints applications. Situations which soft constraints are intended for include the following:

(a) Over-constrained problems, where we cannot satisfy all the constraints. In this case a degree might be associated to a tuple, which depends on the set of constraints that it falsifies, representing perhaps a cost of failing to satisfy those constraints. Approaches to this kind of problem include those described in (Freuder and Wallace, 92) [9].
(b) Situations where a constraint can be *partially* satisfied, so that there are degrees of satisfaction; fuzzy constraint satisfaction [5] is intended for this type of situation.
(c) Situations where the identity of a constraint is uncertain, so that it can be uncertain whether a constraint is satisfied or not by a tuple, see e.g., [8].

This paper takes a logic-based approach to partially satisfied constraints for situations where a tuple can be assigned partially ordered degrees of satisfaction for a constraint. Allowing partially ordered degrees of satisfaction increases the representational flexibility; it allows one not to have to always decide if the degree to which a tuple satisfies a constraint is more or less than the degree to which the tuple satisfies another constraint. A semantics is defined, which determines which constraints can be inferred from a set of constraints, and a sound and complete proof theory is given. The consequent formalism is closely related to idempotent semiring-based CSPs and in particular, set-based CSPs, as well as lattice-valued possibilistic logic.

In section 2, finite CSPs are expressed as a logic with a simple sound and complete proof theory. This is generalised to deduction for partially satisfied constraints in section 3. In the case where the partially ordered set of degrees is a distributive lattice, the proof theory from section 2 generalises in a very simple way, as shown in section 3.1. The close links with idempotent semiring-based CSPs and lattice-valued possibilistic logic are described in section 3.2 and 3.3, respectively. Section 3.4 considers the general case of partially ordered degrees, by constructing embeddings of partially ordered sets into a particular distributive lattice; this enables one to use the computational approach of 3.1 for the general case. This paper focuses on deduction of constraints; however this is often not what one is primarily interested in, but in finding solutions (though deduction can be used to guide the search); 3.5 looks briefly at the problem of finding optimal complete tuples. Section 4 describes an alternative semantic consequence relation for partially satisfied constraints.

2 Finite CSPs as a Logic

Let V be a finite set of variables, where each variable $V_i \in V$ has finite domain $\underline{V_i}$. For $U \subseteq V$, define $\underline{U}$ to be the set of possible assignments to variables U, that is, $\prod_{V_i \in U} \underline{V_i}$. By convention, the empty set of variables is defined to have a single value $\diamond$. In this framework, we define a constraint c to be a subset of $\underline{V_c}$, for some set of variables $V_c \subseteq V$. A complete tuple x is an element of $\underline{V}$. For $U \subseteq V$ let $x^{\downarrow U}$ be the projection of x to variables U. Associated with constraint c is the constraint $c^{\uparrow V} = \{x \in \underline{V} : x^{\downarrow V_c} \in c\}$ on variables V. c may be considered as a more compact representation of $c^{\uparrow V}$. Similarly, for U such that $V_c \subseteq U \subseteq V$, define $c^{\uparrow U}$ to be $\{y \in \underline{U} : y^{\downarrow V_c} \in c\}$, where $y^{\downarrow V_c}$ is the projection of partial tuple y to variables V_c.

For example, let $V = \{V_1, V_2, V_3, V_4, V_5\}$, for $i = 1, \ldots, 5$, let $\underline{V_i} = \{a_i, b_i\}$. Let $U = \{V_1, V_2\}$. Then $\underline{U} = \{(a_1, a_2), (a_1, b_2), (b_1, a_2), (b_1, b_2)\}$. Define constraint c by: $V_c = \{V_1, V_2\}$, $c = \{(a_1, a_2), (b_1, b_2)\}$. $x = (a_1, a_2, b_3, a_4, b_5)$ is an example of a complete tuple. $x \in c^{\uparrow V}$ since $x^{\downarrow V_c} = (a_1, a_2)$ is in c.

Semantics

A model is intended to represent a possible state of the world. We will define what a model is, and which constraints it satisfies. Then constraint d will be said to be a consequence of set of constraints C (written $C \models d$) if every model of C is a model of d. There are different natural notions of model.

Semantics(I): A model is a complete tuple, i.e., an element of $\underline{V}$. $x \models c$ if and only if $x \in c^{\uparrow V}$ if and only if $x^{\downarrow V_c} \in c$.

This semantics is appropriate if the variables are uncertainty variables, rather than choice/decision variables: it assumes a unique true (but unknown) complete tuple x.

Semantics(II): If we have control over the variables then models in the above sense do not represent mutually exclusive and exhaustive possible states of the world: no complete tuple x may satisfy the constraints, or many might. So it is then natural to define a model M to be a subset of $\underline{V}$. $M \models c$ if and only if $M \subseteq c^{\uparrow V}$. In this semantics each constraint is viewed as a restriction on possible complete tuples. M represents the correct set of all adequate tuples.

One way of viewing this is that we have a particular purpose in mind, and certain choices, represented by complete tuples, will be adequate, and some not. So a possible state of the world is the set of complete tuples that is adequate for that purpose. Each constraint is then interpreted as restricting this set of choices, telling us that some of these choices are not adequate.

In either case we say $C \models d$ if and only if every model of C is a model of d. The two semantics are equivalent in the sense that the same constraints d follow from C.

Note that for any constraint c, c is satisfied by the same models as $c^{\uparrow V}$, so c and $c^{\uparrow V}$ are equivalent. In particular the empty constraint $\emptyset_U$ is equivalent for each U, so that $\emptyset_\emptyset$ is equivalent to $\emptyset_V$: the only model satisfying it is $M = \emptyset$ (in the first semantics it has no models).

C is called *satisfiable* if and only if there exists some complete tuple $x \in \underline{V}$ with x satisfying each element of C (i.e., $x \models C$ in Semantics(I)), which is if and only if there exists some non-empty M with $M \models C$ in Semantics (II). Otherwise C is said to be *unsatisfiable.* C is unsatisfiable if and only if $C \models \emptyset_\emptyset$. The consequence relation thus determines satisfiability for sets of constraints.

The construction ensures certain basic properties of the consequence relation $\models$: it is reflexive (if $d \in C$ then $C \models d$), transitive (if $C \models d$ for all $d \in D$ and $D \models e$ then $C \models e$), and monotonic (if $C \models d$ then $C \cup D \models d$).

Combination of constraints. For constraints c and d, define $c \wedge d$ to be the constraint $c^{\uparrow U} \cap d^{\uparrow U}$ on variables $U = V_c \cup V_d$. So $y \in \underline{U}$ is in $c \wedge d$ if and only if $y^{\downarrow V_c} \in c$ and $y^{\downarrow V_d} \in d$. $c \wedge d$ is essentially intersection, as $c^{\uparrow U}$ is essentially the same constraint as c. This operation is commutative and associative, so for finite set of constraints C we write $\bigwedge C$ for the result of applying $\wedge$ sequentially the elements of C.

Projection of constraints For constraint c, and $U \subseteq V_c$, define $c^{\downarrow U}$, the projection of c to U, to be $\{y^{\downarrow U} : y \in c\}$.

Identity (trivial) constraints For $U \subseteq V$ let 1_U be the constraint $\underline{U}$. (Every U-tuple is allowed.) This gives us no information, and is satisfied by all models.

Proof theory Define the proof theory by the following axiom and inference rules:
Axiom: 1_V.

Inference Rules:

From c and d deduce $c \wedge d$.
For each constraint c and $U \subseteq V_c$ the following inference rule:
From c deduce $c^{\downarrow U}$.
When $V_c = V_d$ and $c \subseteq d$:
From c deduce d.

In the usual fashion we say that d can be proved from C if there exists a sequence of elements in the language (constraints) $d_1, \ldots, d_n$ with $d_n = d$ and each d_i being either an element of C, an axiom, or the result of applying one of the inference rules to earlier elements in the sequence.

Theorem 1. *The above proof theory is sound and complete: $C \models d$ if and only if d can be proved from C.*

Deletion Inference rule A generalised arc consistency can be expressed in terms of derived inference rules: the combination of projections of constraints to a single variable. Another important derived inference rule is deletion of a variable. Let $V_i \in V$. Combine all constraints involving that variable and project to $U - \{V_i\}$: $(\bigwedge \{c \in C : V_c \ni V_i\})^{\downarrow U - \{V_i\}}$, where $U = \bigcup \{V_c : c \in C, V_c \ni c\}$ is the set of variables involved in the combination.

It can be shown that $(C \wedge 1_{V_d})^{\downarrow V_d}$ can be computed by repeated deletion of the variables not in V_d; this follows because combination and projection obey the Shenoy-Shafer axioms (Shenoy and Shafer, 90) [14, 13]; this is an instance of the fusion algorithm [15] (c.f. also Dechter's Bucket Elimination [11]), which for this case corresponds to the adaptive consistency algorithm [12, 11]. Therefore the deletion inference rule gives a sound and complete proof procedure. This approach is efficient if one can find a hypertree cover which doesn't involve too large sets for the constraints' variable sets hypergraph $\{V_c : c \in C\}$ since then none of the combinations need involve too many variables. One can also use the Shenoy-Shafer approach to efficiently find, for example, all the possible values in each domain.

Relationship with propositional logic embedding. This logical representation might be viewed as a partial embedding of Finite CSPs in propositional logic. To represent as a propositional theory (c.f., section 14 of (Mackworth, 92) [10]), one can create a propositional variable for each element in each domain, with mutually exclusivity and exhaustivity axioms between elements in the same domain. So for each variable $V_i \in V$ we have $|\underline{V_i}|$ associated propositional variables, $\{p_i^k : k \in \underline{V_i}\}$. Mutual exclusivity of elements in domain $\underline{V_i}$ are expressed as axioms $\neg(p_i^k \wedge p_i^l)$ for all $k, l \in \underline{V_i}$ with $k \neq l$. Exhaustivity of $\underline{V_i}$ is expressed by the axiom $\bigvee_{k \in \underline{V_i}} p_i^k$. A partial tuple is represented as a conjunction of these propositional variables, and a constraint as a disjunction of partial tuples. So a constraint $c \subseteq V_c$ is represented by the formula $\bigvee_{y \in c} p(y)$ where $p(y)$ is the formula $\bigwedge_{V_i \in V_c} p_i^{y(V_i)}$. Combination of constraints then is essentially conjunction, but where we 'multiply out' to maintain the disjunction of conjunctions form (simplified by the fact that one only has to consider compatible pairs of partial tuples). Projection corresponds to omitting some conjuncts of partial tuples in a constraint.

Alternative Notation A constraint c as defined above can also be viewed as a function from $\underline{V_c}$ to $\{0,1\}$: assigning 1's to partial tuples in the constraint. Similarly, models can be viewed as functions from $\underline{V}$ to $\{0,1\}$. To emphasise the link with the partially satisfied constraints framework of section 3, we give the previous definitions in terms of this alternative notation.

$M \models c$ if and only if $M \leq c^{\uparrow V}$, i.e., for all complete tuples $x \in \underline{V}$, $M(x) \leq c^{\uparrow V}(x)$, which is if and only if for all $x \in \underline{V}$, $M(x) \leq c(x^{\downarrow V_c})$.

$c \wedge d$ is the constraint on $V_c \cup V_d$ given by: for $y \in \underline{V_c \cup V_d}$, $(c \wedge d)(y) = c(y^{\downarrow V_c}) \wedge d(y^{\downarrow V_d})$, where the last $\wedge$ is logical AND (i.e., min). For $U \subseteq V_c$, $c^{\downarrow U}$, the projection of c to U, is given by: for $u \in \underline{U}$, $c^{\downarrow U}(u) = \bigvee \{c(y) : y \in \underline{V_c}, y^{\downarrow U} = u\}$, where $\bigvee$ in the last equation is logical OR (i.e., max). 1_U is the constraint on variables U which is everywhere equal to 1: for all $u \in \underline{U}$, $1_U(u) = 1$. The only difference in the notation in the proof theory is that the subset inference rule is replaced by:

When $V_c = V_d$ and $c \leq d$: *From c deduce d,*

since: $c \leq d$, i.e., for all $y \in \underline{V_c}$, $c(y) \leq d(y)$, if and only if every tuple in c is a tuple in d.

3 Partially Satisfied Constraints

Suppose we now want to allow degrees of satisfaction of constraints. We choose a finite partially ordered set $\mathcal{A} = (A, \preceq, 0, 1)$ to represent these degrees, where A contains a unique a maximal element 1 and a unique minimal element 0. Define an $\mathcal{A}$-constraint c to be a function from $\underline{V_c}$ to A, for some set of variables $V_c \subseteq V$. A value of 1 is intended to mean that the tuple completely satisfies the constraint, and a value of 0 means that the tuple doesn't satisfy the constraint at all.

Semantics We would like to say what $\mathcal{A}$-constraints d can be deduced from a set of $\mathcal{A}$-constraints C. The intuition that this is based on is as follows: we imagine that the constraints are for a particular purpose, and that there exists some unknown function $M : \underline{V} \to A$ which says the true degree that each complete tuple $x \in \underline{V}$ is adequate for our purposes. Each constraint c in our set C is then taken to restrict the possible such models M. A constraint c with $V_c = V$ is interpreted as telling us that for all $x \in \underline{V}$, the true degree of adequacy $M(x)$ is bounded above by $c(x)$, i.e., $M(x) \preceq c(x)$. Other constraints, with $V_c \neq V$

are considered as more compact representations of the constraint $c^{\uparrow V}$ defined by $c^{\uparrow V}(x) = c(x^{\downarrow V_c})$. Hence c is interpreted as telling us that for all $x \in \underline{V}$, the true degree of adequacy $M(x)$ is bounded above by $c^{\uparrow V}(x) = c(x^{\downarrow V_c})$.

So we say that $M \models c$ *(M satisfies c)* if and only if $M \preceq c^{\uparrow V}$, that is, for all $x \in \underline{V}$, $M(x) \preceq c^{\uparrow V}(x)$, in other words: $M(x) \preceq c(x^{\downarrow V_c})$. For set of $\mathcal{A}$-constraints C and $\mathcal{A}$-constraint d we define $C \models d$ if and only if every model M of (every constraint in) C is also model of d. If C is a singleton set $\{c\}$, we may write $c \models d$ instead of $\{c\} \models d$.

There is a natural ordering between constraints c and d with $V_c = V_d$, extending $\preceq$ pointwise: we say $c \preceq d$ if for all $y \in \underline{V_c}$, $c(y) \preceq d(y)$.

An alternative notion of semantic consequence is explored in section 4, based on a similar intuition, but allowing a model M to take values in a partially ordered set that extends $\mathcal{A}$.

The semantic definition is not very helpful for computing the consequences d of C; we need some more computationally useful characterisation of consequence. To do this end we consider a special case first, in 3.1, when $\mathcal{A}$ is a distributive lattice. The formalism is then very closely related to idempotent semiring-based CSPs [2] with the natural notion of consequence being the same, as discussed in section 3.2. Also, for this special case, the formalism is very closely related to lattice-valued possibilistic logic [7], as shown in section 3.3. In 3.4 the general case for A is considered, by embedding the partially ordered set in a particular distributive lattice, and using the formalism of 3.1 to determine consequence. 3.5 considers briefly the problem of finding tuples with degree greater than a particular α, and finding optimal complete tuples.

3.1 The case when $\mathcal{A}$ is a distributive lattice

We first look at this special case of a partial order. The definition of semantic consequence given above is a direct generalisation of that for FCSPs: with the alternative notation, just replacing $\leq$ by $\preceq$. It turns out that the proof theory, soundness and completeness, and associated algorithms generalise in exactly the same way, when $\mathcal{A} = (A, 0, 1, \preceq)$ is a distributive lattice.

For finite A, $(A, 0, 1, \preceq)$ is a distributive lattice if and only if any $\alpha, \beta \in A$ have a greatest lower bound $\alpha \wedge \beta$ in A (so that $\gamma \preceq \alpha, \beta$ implies $\gamma \preceq \alpha \wedge \beta$), and a least upper bound $\alpha \vee \beta$ (so that $\alpha, \beta \preceq \gamma$ implies $\alpha \vee \beta \preceq \gamma$), which satisfy the distributivity property: for all $\alpha, \beta, \gamma \in A$, $\gamma \wedge (\alpha \vee \beta) = (\gamma \wedge \alpha) \vee (\gamma \wedge \beta)$.

An important example of a distributive lattice is a subset lattice: let A be a set of subsets of a set Θ, which is closed under intersection and union: i.e., if $\alpha, \beta \in A$ then $\alpha, \beta \subseteq \Theta$ and $\alpha \cap \beta, \alpha \cup \beta \in A$. In fact, any finite distributive lattice is isomorphic to such a subset lattice (using e.g., the construction in section 3.4).

The lattice properties enable us to define combination and projection of $\mathcal{A}$-constraints, with the same form as before.

Let $c : V_c \to A$ and $d : V_d \to A$ be two $\mathcal{A}$-constraints. Their combination $c \wedge d$ is the $\mathcal{A}$-constraint on variables $V_c \cup V_d$ given by, for $y \in \underline{V_c \cup V_d}$, $(c \wedge d)(y) = c(y^{\downarrow V_c}) \wedge d(y^{\downarrow V_d})$.

For $U \subseteq V_c$, $c^{\downarrow U}$, the projection of c to U, is given by: for $u \in \underline{U}$, $c^{\downarrow U}(u) = \bigvee \{c(y) : y \in \underline{V_c}, y^{\downarrow U} = u\}$, where $\bigvee$ in the last equation is the $\vee$ in the lattice $\mathcal{A}$. The degree of the projection is the meet of $c(y)$ over all y which project to u.

1_U is the constraint on variables U which is everywhere equal to 1: for all $u \in \underline{U}$, $1_U(u) = 1$.

Proof theory

Define the proof theory by the following axiom and inference rules:
Axiom: 1_V.
Inference Rules:
From c and d deduce $c \wedge d$.
For each constraint c and $U \subseteq V_c$ the following inference rule:
From c deduce $c^{\downarrow U}$.
When $V_c = V_d$ and $c \preceq d$:
From c deduce d.

The following lemma gives some basic properties.

Lemma 1. *Let M be a model for $\mathcal{A}$-constraints, let c and d be $\mathcal{A}$-constraints, and let C be a set of $\mathcal{A}$-constraints.*

(i) For all models M, and $U \subseteq V$, $M \models 1_U$.
(ii) $c \wedge 1_V = c^{\uparrow V}$; $M \models c \iff M \models c^{\uparrow V}$.
(iii) $M \models C$ if and only if $M \models \bigwedge C$.
(iv) $c \models d$ if and only if $c^{\uparrow V} \preceq d^{\uparrow V}$ if and only if for all $x \in \underline{V}$, $c(x^{\downarrow V_c}) \preceq d(x^{\downarrow V_d})$.
(v) If $V_c = V_d$ then $c \models d$ if and only if for all $y \in \underline{V_c}$, $c(y) \preceq d(y)$.
(vi) If $U \subseteq V_c$ then $c \models c^{\downarrow U}$.
(vii) If $V_c \supseteq V_d$ then $c \models d \iff c^{\downarrow V_d} \models d$.

The following proposition follows from the application of parts (iii), (i), (vii) and (v) of the lemma.

Proposition 1. *$C \models d$ if and only if $\bigwedge C \models d$ if and only if $(\bigwedge C \wedge 1_{V_d})^{\downarrow V_d} \preceq d$. If $\bigcup_{c \in C} V_c \supseteq V_d$ then $C \models d$ if and only if $(\bigwedge C)^{\downarrow V_d} \preceq d$.*

Theorem 2 (Soundness and Completeness). *$C \models d$ if and only if d can be proved from C using the axiom and inference rules.*

Soundness follows from parts (i), (iii), (v) and (vi) of the above lemma. Completeness follows from the proposition since we can prove 1_{V_d} from 1_V using the projection inference rule; then from 1_{V_d} and C we can prove $\bigwedge C \wedge 1_{V_d}$ using the combination inference rule and then $(\bigwedge C \wedge 1_{V_d})^{\downarrow V_d}$ using the projection inference rule. So, if $C \models d$ we can deduce d from C using finally the $\preceq$ inference rule, showing completeness.

Again we have a form of arc consistency as sound inference rules. Also the Shenoy-Shafer axioms are satisfied by combination and projection for $\mathcal{A}$-constraints (see in particular theorems 18 and 19 of (Bistarelli et al, 97) [2]). This implies that the deletion inference rule leads again to a sound and complete inference procedure which is efficient if there's a nice hypertree cover of the hypergraph $\{V_c : c \in C\}$.

3.2 Relationship with idempotent semiring-based CSPs

It is clear that there is a close relationship between the formalism in 3.1 and idempotent semiring-based CSPs (Bistarelli et al, 97) [2], as the combination and projection operations defined above are the same as those used in semiring-based CSPs. In this section we show that the notion of consequence defined above is the same as the natural notion of consequence in idempotent semiring-based CSPs.

In an idempotent semiring-based CSP the primary objects are $\mathcal{A}$-constraints, with the degrees A forming a distributive lattice (see [2] theorem 10). Without loss of generality, we can consider A to be finite (if it is not we can consider instead the (finite) sublattice generated by the degrees in A that appear in a finite set of constraints.)

A constraint problem, for an idempotent semiring-based CSP, is defined to be a pair $\langle C, U\rangle$ where C is a set of $\mathcal{A}$-constraints and $U \subseteq V$. The solution of this pair is defined to be the $\mathcal{A}$-constraint $\mathit{Sol}(\langle C, U\rangle) = (\bigwedge C)^{\downarrow U}$. For constraint problems $\langle C_1, U\rangle$ and $\langle C_2, U\rangle$, the relation $\langle C_1, U\rangle \sqsubseteq \langle C_2, U\rangle$ is said to hold if $\mathit{Sol}(\langle C_1, U\rangle) \preceq \mathit{Sol}(\langle C_2, U\rangle)$. Equivalence between constraint problems is defined as follows: $\langle C_1, U\rangle \equiv \langle C_2, U\rangle$ if $\langle C_1, U\rangle \sqsubseteq \langle C_2, U\rangle$ and $\langle C_2, U\rangle \sqsubseteq \langle C_1, U\rangle$, i.e., if $\mathit{Sol}(\langle C_1, U\rangle) = \mathit{Sol}(\langle C_2, U\rangle)$.

There is a natural definition for deduction of constraints in this system.

Proposition 2. *Let $C \cup \{d\}$ be a set of $\mathcal{A}$-constraints such that $V_C \supseteq V_d$, where $V_C = \bigcup_{c \in C} V_c$. Then the following are equivalent: (i) $(\bigwedge C)^{\downarrow V_d} \preceq d$; (ii) $(\bigwedge C) \preceq d \wedge 1_{V_C}$; (iii) $\mathit{Sol}(\langle C, V_d\rangle) \preceq d$; (iv) $\langle C, V_d\rangle \sqsubseteq \langle \{d\}, V_d\rangle$; (v) $\langle C, V_C\rangle \sqsubseteq \langle \{d, 1_{V_C}\}, V_C\rangle$; (vi) $\langle C \cup \{d\}, V_C\rangle \equiv \langle C, V_C\rangle$; (vii) for any $U \subseteq V_C$, $\langle C \cup \{d\}, U\rangle \equiv \langle C, U\rangle$; (viii) $(\bigwedge C) \wedge d = \bigwedge C$.*

For set C of $\mathcal{A}$-constraints and $\mathcal{A}$-constraint d with $\bigcup_{c \in C} V_c \supseteq V_d$, we can say that d is a consequence of C if $\mathit{Sol}(\langle C, V_d\rangle) \preceq d$ (or any other of the equivalent forms in the proposition). Using part (viii), this can be seen to tally with the entailment relation defined in (Bistarelli et al, 02) [4] (see definition 6).

By part (i) of this proposition and the previous proposition, this notion of consequence is the same as the consequence defined earlier: $C \models d$. Therefore the framework described above, for the distributive lattice case, gives a semantics for consequence in idempotent semiring-based CSPs.

This semantics differs in a fundamental way from the model-theoretic semantics for semiring-based Constraint Logic Programming given in (Bistarelli et al, 2001) [3]. In the semantics described above (both in the hard constraints and soft constraints cases), a constraint is taken to restrict allowable tuples, and so conveys only negative information: a tuple not in the constraint is inadequate, tuples in the constraint may or may not be adequate. Consequently, there exists a unique maximal model of a set of constraints C, i.e., $(\bigwedge C)^{\uparrow V}$. In contrast, in the logic programming-based semantics of [3], constraints are expressed using a set of facts expressing known instances of the constraints, which is not assumed to be an exhaustive set of instances; these therefore express positive information, and there exists a unique minimal interpretation of such a set of facts, and more generally a program; we then query to see if there exists a known common instance (i.e., a solution) of the set of constraints. Which of the two semantics is more appropriate will depend on the application.

3.3 Relationship with lattice-valued possibilistic logic

This logic of soft constraints is also closely related to a form of possibilistic logic [7] which takes values in a distributive lattice, see (Dubois et al, 91) [6], and (Dubois et al, 94) [7] section 4.3.

Let $\mathcal{A} = (A, \preceq, 0, 1)$ be a complete distributive lattice. Define an $\mathcal{A}$-possibility distribution over set Ω to be a function π from Ω to A. Define the associated $\mathcal{A}$-possibility measure to be the function $\text{Poss}_\pi : 2^\Omega \to A$ given by $\text{Poss}_\pi(X) = \bigvee \{\pi(\omega) : \omega \in X\}$.

A lattice-valued possibilistic logic over FCSPs. We might embed FCSPs in propositional logic and then use possibilistic logic over lattices. Here we take a more direct approach, defining such a possibilistic logic over the language of FCSPs. Possibilistic logic usually involves lower bounds on the necessity values of propositions; these are equivalent to upper bounds on the possibility values of the negated propositions, and for convenience, these are used here.

Let $\mathcal{L}$ be the set of pairs (y, α) for $\alpha \in A$ and partial tuple $y \in \underline{U}$ for some $U \subseteq V$. The intended meaning of this pair (y, α) is that the possibility measure of y is at most α. For $\mathcal{A}$-possibility distribution π over $\underline{V}$, $U \subseteq V$, partial tuple $y \in \underline{U}$, and $\alpha \in A$, define $\pi \models (y, \alpha)$ if and only if $\text{Poss}_\pi(\{x \in \underline{V} : x^{\downarrow U} = y\}) \preceq \alpha$. Therefore $\pi \models (y, \alpha)$ if and only if for all x such that $x^{\downarrow U} = y$, $\pi(x) \preceq \alpha$. For Γ, $\Delta \subseteq \mathcal{L}$ we say $\Gamma \models \Delta$ if and only if $\pi \models \Delta$ for all π such that $\pi \models \Gamma$. ($\pi \models \Gamma$ if and only if $\pi \models (y, \alpha)$ for all $(y, \alpha) \in \Gamma$.)

Relationship with logic of partially satisfied constraints An $\mathcal{A}$-constraint $c : V_c \to A$ corresponds to a set of pairs $c^* \subseteq \mathcal{L}$, where $c^* = \{(y, c(y)) : y \in \underline{V_c}\}$. For set of such soft constraints C, write $C^* = \bigcup_{c \in C} c^*$. $\mathcal{A}$-possibility distributions π are just models in the partially satisfied constraints logic defined above, and, for $\mathcal{A}$-constraint c, $\pi \models c$ in the soft constraint logic if and only if $\pi \models c^*$. Hence we have $C \models d$ if and only if $C^* \models d^*$. Thus deduction for lattice-valued constraints can be considered as lattice-valued possibilistic deduction.

This result suggests the application of the more general partially ordered case (sections 3.4 and 4) to produce a poset-valued-possibilistic logic, of a different form to that produced by Benferhat et al., [1].

3.4 The General Partially Ordered Set Case

Here we consider the general case: where $\mathcal{A} = (A, 0, 1, \preceq)$ is an arbitrary partially ordered set that has a unique minimal element 0 and a unique maximal element 1. The idea is to embed the partially ordered set into a lattice of subsets in such a way that the ordering information is maintained, but without adding any extra ordering of elements. Then we can use the proof procedures for the subset lattice case to make deductions for this general case.

Let $\preceq$ be a partial order on set A. For $B \subseteq A$ and $\gamma \in A$ we write $B \Rightarrow \gamma$ if every lower bound of (every element of) B is a lower bound of γ, i.e., if $\alpha \in A$ is such that $\alpha \preceq B$, then $\alpha \preceq \gamma$ (where $\alpha \preceq B$ iff for all $\beta \in B$, $\alpha \preceq \beta$). Let C be a set of $\mathcal{A}$-constraints: i.e., each $c \in C$ is a function from $\underline{V_c}$ to A, for some $V_c \subseteq V$. For $x \in \underline{V}$ define $C(x)$ to be the set $\{c(x^{\downarrow V_c}) : c \in C\}$.

Proposition 3. *With the above notation, $C \models d$ if and only if for all $x \in \underline{V}$, $C(x) \Rightarrow d(x^{\downarrow V_d})$.*

This proposition expresses deduction in terms of the relation $\Rightarrow$. To prove the 'if' part: if $C(x) \Rightarrow d(x^{\downarrow V_d})$ and $M \models C$ then for all $x \in \underline{V}$, $M(x) \preceq C(x)$ so by the definition of $\Rightarrow$, $M(x) \preceq d(x^{\downarrow V_d})$; this shows $M \models d$, proving $C \models d$. For the converse, suppose that it is not the case that for all $x \in \underline{V}$, $C(x) \Rightarrow d(x^{\downarrow V_d})$. Then there exists $x_0 \in \underline{V}$ and $\alpha \in A$ such that $\alpha \preceq C(x_0)$ but $\alpha \not\preceq d(x_0^{\downarrow V_d})$. Define model M by $M(x_0) = \alpha$ and for all $x \neq x_0$, $M(x) = 0$. It can be easily seen that $M \models C$ but $M \not\models d$, proving that $C \not\models d$.

This leads to the following result, essentially because the condition that $\bigcap_{\beta \in B} X_\beta \subseteq X_\gamma$ is equivalent to the condition $\{X_\beta : \beta \in B\} \Rightarrow X_\gamma$ (in the subset lattice), and the proposition tells us that deduction only depends on $\Rightarrow$.

Theorem 3. *Suppose we have a function χ from A to 2^Θ for some finite set Θ, which we write as $\alpha \mapsto X_\alpha$. For each $c : \underline{V_c} \to A$ define $c^\chi : \underline{V_c} \to 2^\Theta$ to be c followed by χ, so that $c^\chi(y) = X_{c(y)}$. Let C be a set of $\mathcal{A}$-constraints. Define C^χ to be $\{c^\chi : c \in C\}$. Suppose further that function χ satisfies the condition: for all $B \subseteq A$ and $\gamma \in A$,*

$$B \Rightarrow \gamma \iff \bigcap_{\beta \in B} X_\beta \subseteq X_\gamma.$$

Then $C \models d$ if and only if $C^\chi \models d^\chi$.

(Note that the the latter $\models$ is based on models $M : \underline{V} \to 2^\Theta$.)

What this theorem shows is that if we have the appropriate property on the embedding χ then we can determine deduction of the $\mathcal{A}$-constraints by using considering deduction of the 2^Θ-constraints; for the latter we have the results of the previous section: a simple sound and complete proof theory which gives rise to an efficient computational procedure given appropriate structure on the constraints' variables.

It is easy to find an embedding with the appropriate properties. In particular we can define Θ to be A and X_α to be $\{\beta \in A : \beta \preceq \alpha\}$ (see below). However, if A is large, working with subsets of A can be computationally expensive. In the following subsection we give a way of constructing such an embedding which can lead to much smaller Θ than A.

An algorithm for embedding a partial-ordered set into a lattice of subsets.

First we give natural sufficient conditions for the above required property on the embedding χ.

Lemma 2. *Suppose $\alpha \mapsto X_\alpha$ satisfies the following properties:*

(i) $\alpha \preceq \beta \iff X_\alpha \subseteq X_\beta$;
(ii) For any $B \subseteq A$ and $\theta \in \bigcap_{\beta \in B} X_\beta$, there exists $\alpha \in A$ with $\theta \in X_\alpha \subseteq \bigcap_{\beta \in B} X_\beta$.

Then for any $B \subseteq A$ and $\gamma \in A$, $B \Rightarrow \gamma \iff \bigcap_{\beta \in B} X_\beta \subseteq X_\gamma$.

To prove this: because of (i), $B \Rightarrow \gamma$ can be written as: for all α, $[X_\alpha \subseteq \bigcap_{\beta \in B} X_\beta$ implies $X_\alpha \subseteq X_\gamma]$. This clearly is implied by $\bigcap_{\beta \in B} X_\beta \subseteq X_\gamma$. For the converse, consider any $\theta \in \bigcap_{\beta \in B} X_\beta$; by (ii) there exists α with $\theta \in X_\alpha \subseteq \bigcap_{\beta \in B} X_\beta$ so $\theta \in X_\alpha \subseteq X_\gamma$, proving $\bigcap_{\beta \in B} X_\beta \subseteq X_\gamma$.

We will consider embeddings of a particular form. Let A' be a subset of A. For $\alpha \in A$ let $X_\alpha^{A'}$ be the set $\{\alpha' \in A' : \alpha' \preceq \alpha\}$. It can easily be seen that for any A', the mapping $\alpha \mapsto X_\alpha^{A'}$ satisfies (ii) above (using $\alpha = \theta$), and half of (i): if $\alpha \preceq \beta$ then $X_\alpha^{A'} \subseteq X_\beta^{A'}$, (by transitivity of $\preceq$). Setting $A' = A - \{0\}$ (or indeed $A' = A$) we get the other half of (i): if $X_\alpha^{A'} \subseteq X_\beta^{A'}$ then $\alpha \preceq \beta$. However, we can very often find a much smaller A' that still satisfies both conditions (i) and (ii) of the lemma, as shown below.

If set of elements $B \subseteq A$ have a unique greatest lower bound α then $X_\alpha^{A'} = \bigcap_{\beta \in B} X_\beta^{A'}$. A monotonicity property also holds: for $A'' \subseteq A'$, $X_\alpha^{A'} \subseteq X_\beta^{A'}$ implies $X_\alpha^{A''} \subseteq X_\beta^{A''}$ (because $X_\alpha^{A''} = X_\alpha^{A'} \cap A''$). This makes the construction of a particular A' much easier: we can add elements to A', knowing that this can't lead to any extra orderings $X_\alpha^{A'} \subseteq X_\beta^{A'}$.

Construction of a particular A' that satisfies the desired conditions
We list the elements of A in an order compatible with $\preceq$, starting with 0, so that $\alpha_0 = 0$, and if $\alpha_i \preceq \alpha_j$ then $i \leq j$. We build up A', element by element, with the final set A' being $A_{|A|-1}$. We set $A_0 = \emptyset$. For each $i = 0, \ldots, |A| - 1$ we will have $A_i \subseteq \{\alpha_0, \ldots, \alpha_i\}$, with $A_0 \subseteq \cdots \subseteq A_{i-1} \subseteq A_i \cdots \subseteq A_{|A|-1}$. We will also arrange that for $j, k \leq i$, $X_{\alpha_j}^{A_i} \subseteq X_{\alpha_k}^{A_i}$ implies $\alpha_j \preceq \alpha_k$. The monotonicity property ensures that when we increase A_i, this property still holds. In particular, for each i, we just need to check that this property holds for $j, k \leq i$ with either $j = i$ or $k = i$: because for $j, k \leq i - 1$ we have by induction $X_{\alpha_j}^{A_{i-1}} \subseteq X_{\alpha_k}^{A_{i-1}}$ implies $\alpha_j \preceq \alpha_k$, so by monotonicity, $X_{\alpha_j}^{A_i} \subseteq X_{\alpha_k}^{A_i}$ implies $\alpha_j \preceq \alpha_k$.

If we have for all $j, k \leq i$ with either $j = i$ or $k = i$,

$$X_{\alpha_j}^{A_{i-1}} \subseteq X_{\alpha_k}^{A_{i-1}} \text{ implies } \alpha_j \preceq \alpha_k,$$

then we can set $A_i = A_{i-1}$. Otherwise we add extra elements that restore the property. The violations of this property are of two types, according to whether

$j = i$ or $k = i$. In the former case we can restore the property by including α_i in A_i; in the latter case by including α_j in A_i.

If $X_{\alpha_i}^{A_{i-1}} \subseteq X_{\alpha_k}^{A_{i-1}}$ for some $k < i$ (and by the definition of the ordering we don't have $\alpha_i \preceq \alpha_k$) then we can correct this by adding α_i to A_i, ensuring that $X_{\alpha_i}^{A_i} \not\subseteq X_{\alpha_k}^{A_i}$, since $X_{\alpha_i}^{A_i}$ contains α_i but $X_{\alpha_k}^{A_i}$ doesn't.

Suppose $X_{\alpha_j}^{A_{i-1}} \subseteq X_{\alpha_i}^{A_{i-1}}$ for some $j < i$ such that $\alpha_j \not\preceq \alpha_i$. Then we correct this by adding α_j to A_i, ensuring that $X_{\alpha_j}^{A_i} \not\subseteq X_{\alpha_i}^{A_i}$.

So, in summary, we define, for each $i = 1, \ldots, A_{|A|-1}$, $A_i = A_{i-1} \cup Y_i \cup Z_i$ where we set $Y_i = \{\alpha_i\}$ if there exists $k < i$ with $X_{\alpha_i}^{A_{i-1}} \subseteq X_{\alpha_k}^{A_{i-1}}$; otherwise we set $Y_i = \emptyset$. We set Z_i to be the set of all α_j such that (a) $j < i$, (b) $\alpha_j \not\preceq \alpha_i$, and (c) $X_{\alpha_j}^{A_{i-1}} \subseteq X_{\alpha_i}^{A_{i-1}}$.

The monotonicity property then ensures that, with $A' = A_{|A|-1}$, we have $X_\alpha^{A'} \subseteq X_\beta^{A'}$ implies $\alpha \preceq \beta$, as required. As remarked earlier we have the converse, so $X_\alpha^{A'} \subseteq X_\beta^{A'}$ if and only if $\alpha \preceq \beta$. In particular $\alpha \mapsto X_\alpha^{A'}$ is one-to-one.

To summarise, we have shown that this embedding $\alpha \mapsto X_\alpha$, when $X_\alpha = \{\alpha' \in A' : \alpha' \preceq \alpha\}$ satisfies the properties (i) and (ii) of the above lemma, and hence the conditions of the theorem. So by the theorem, we can use this embedding to compute consequences of sets of $\mathcal{A}$-constraints, by using the proof theory of section 3.1 applied to the lattice of subsets $\{X_\alpha : \alpha \in A\}$.

This particular construction also has the following nice property (α is said to *cover* β if i.e., $\alpha \succ \beta$ and there does not exist γ with $\alpha \succ \gamma \succ \beta$):

Proposition 4. *If $(A, \preceq, 0, 1)$ is a finite distributive lattice then A' as defined in the above algorithm is the set of all $\alpha \in A$ such that there exists a unique element β which α covers. We then have $X_0 = \emptyset$, $X_1 = A'$, and for all $\alpha, \beta \in A$, $X_{\alpha \wedge \beta} = X_\alpha \cap X_\beta$ and $X_{\alpha \vee \beta} = X_\alpha \cup X_\beta$, so that $\alpha \mapsto X_\alpha$ is an isomorphism between distributive lattices.*

The purpose of the construction is to produce an embedding into not too large a set. In the worst case, when $\preceq$ is a total order, the set A' is just $A - \{0\}$, so $|A'| = |A| - 1$. The other extreme is when A is the lattice of all subsets of a set Ω. Then A' is the set of singleton subsets of Ω, so $|A'| = |\Omega| = \log_2 |A|$.

3.5 Finding complete tuples with degree greater than α, and finding optimal tuples

For any $\alpha \in A$, we can use standard FCSP algorithms to find tuples $x \in \underline{V}$ satisfying each constraint at least to degree α. These are the tuples satisfying set of (conventional) constraints $C_\alpha = \{c_\alpha : c \in C\}$, where $c_\alpha = \{y \in V_c : c(y) \succeq \alpha\}$. Complete tuple $x \in \underline{V}$ might then be considered optimal if it satisfies C_α for some α maximal such that C_α is satisfiable; (one might also further constrain the solutions with additional maximality criteria). Such maximal α can be found using an iterative algorithm.

4 A Different Semantics for Partially Satisfied Constraints

Consider partially ordered set $(A, 0, 1, \preceq)$ with $A = \{0, 1, \alpha, \beta\}$, and $0 \preceq \alpha, \beta \preceq 1$, but no order between α and β. Consider constraints c_1 and c_2 on a single boolean variable given by: $c_1(a) = c_2(a) = 1$, $c_1(b) = \alpha$, $c_2(b) = \beta$. With the semantics given in the previous section, $\{c_1, c_2\}$ entails the constraint d given by $d(a) = 1$, $d(b) = 0$, which states that value b is impossible. This is because, for model M, $M \models c_1$ if and only if $M(b) \preceq \alpha$, and $M \models c_2$ if and only if $M(b) \preceq \beta$. So $M \models \{c_1, c_2\}$ if and only if $M(b) \preceq \alpha, \beta$, hence $M(b) = 0$, since 0 is the only element of A which is a lower bound for both α and β.

Now this is perfectly correct, and also reasonable if the constraints were considered as restricting possible A-valued assignments. However, often the intention will not be precisely that: our input information may consist in the constraints which assign elements to partial tuples, and where we are given some ordering information between some of these elements. We can then generate the partially ordered set $(A, 0, 1, \preceq)$, where A is the set of elements mentioned in the constraints. However, these partially ordered elements may have come from an unknown larger partially ordered set $(A', 0, 1, \preceq')$ which extends $(A, 0, 1, \preceq)$, i.e., $A' \supseteq A$, and for $\alpha, \beta \in A$, $\alpha \preceq \beta$ implies $\alpha \preceq' \beta$.

This leads to the following semantics for $\mathcal{A}$-constraints. A model M is a function from $\underline{V}$ to A' where $(A', 0, 1, \preceq')$ extends $(A, 0, 1, \preceq)$. For $\mathcal{A}$-constraint c, we then define $M \models c$ (M is a model of c, or M satisfies c) if for all $x \in \underline{V}$, $M(x) \preceq' c(x^{\downarrow V_c})$. For set of constraints C and constraint d we then say $C \models^* d$ if $M \models d$ for any M satisfying every constraint in C.

We can proceed in a similar fashion as in 3.4. Suppose $B \cup \{\gamma\} \subseteq A$ and $(A', 0, 1, \preceq')$ extends $(A, 0, 1, \preceq)$. A *lower bound in A' of γ* is an element $\alpha' \in A$ with $\alpha' \preceq' \gamma$. Similarly a lower bound in A' of B is a lower bound of every element of B. We define $B \Rightarrow^* \gamma$ if for every $(A', 0, 1, \preceq')$ extending $(A, 0, 1, \preceq)$, every lower bound in A' of B is a lower bound of γ. We give two intermediate results.

Proposition 5. *Let $C \cup \{d\}$ be a set of $\mathcal{A}$-constraints. With the above notation, $C \models^* d$ if and only if for all $x \in \underline{V}$, $\{c(x^{\downarrow V_c}) : c \in C\} \Rightarrow^* d(x^{\downarrow V_d})$.*

Lemma 3. *Suppose $B \cup \{\gamma\} \subseteq A$. Then $B \Rightarrow^* \gamma$ if and only if there exists $\beta \in B$ with $\beta \preceq \gamma$.*

Bringing the proposition and lemma together we have

Theorem 4. *Let $C \cup \{d\}$ be a set of $\mathcal{A}$-constraints. Then $C \models^* d$ if and only if for all $x \in \underline{V}$, there exists $c \in C$ with $c(x^{\downarrow V_c}) \preceq d(x^{\downarrow V_d})$.*

This result indicates that the consequence relation $\models^*$ will often be much weaker than the consequence $\models$ defined in section 3, since $\Rightarrow^*$ will often be a much weaker relation than $\Rightarrow$ (the antecedents B for $\Rightarrow^*$ don't interact).

Proposition 6. *$C \models^* d$ if and only if for all $z \in \underline{V_d}$, $\{c_{\not\preceq d(z)} : c \in C\} \cup \{z\}$ is unsatisfiable.*

The result can be written in a different way, as this proposition shows. This gives a computational procedure to determine the entailment $C \models^* d$: for each $z \in \underline{V_d}$ we check the satisfiability of a set of standard constraints. This can be done using any of the standard methods, such as the approach described in section 2.

The naturalness of this consequence relation depends on what we know and don't know about the partially ordered set. If $(A, 0, 1, \preceq)$ is a lattice, then this consequence relation will often not be appropriate as it can be unnecessarily weak: this is because we are only requiring the extension to extend the ordering relation $\preceq$, whereas it is often natural to only consider extensions that also extend $\vee$ and $\wedge$; for example, one might expect that a statement that (for a constraint c) $c(y_1)$ is the greatest lower bound of $c(y_2)$ and $c(y_3)$ should not just hold in A, but in any extension we consider.

This will lead to $\Rightarrow^*$ just being $\Rightarrow$ and consequence being the same as that defined in section 3. However, this suggests that sometimes a richer representation of a partially ordered set may be appropriate, for example, as $\mathcal{A} = (A, 0, 1, \preceq, \wedge, \vee)$ where $\wedge$ and $\vee$ are partial operations on A: they are only defined for some pairs of elements. When $\alpha \wedge \beta$ *is* defined, the result should be the greatest lowest bound not only in $\mathcal{A}$ but in any extension of $\mathcal{A}$; extensions $\mathcal{A}'$ should extend these partial functions as well as the ordering relation. We can then apply the same approach as used above, but based on only these extensions.

This extra structure can be used, for example, to define mutual exclusivity between elements in A: $\alpha \wedge \beta = 0$, so that a constraint c with $c(y) = \alpha$ and a constraint d with $c(y) = \beta$ will jointly imply that y is impossible: $M(y) = 0$.

5 Summary

It has been shown that a logical formulation of finite CSPs, involving a semantics, proof theory and computational procedures, generalises in a very simple way to a logic of partially satisfied constraints. This formalism allows one to reason with soft constraints that assign arbitrary partially ordered degrees to tuples, representing the degree that the tuple satisfies the constraint.

In the case where the partially ordered set is a distributive lattice, the logic can be viewed as a lattice-valued possibilistic logic of constraints, and has the same consequence relation as that generated by idempotent semiring-based CSPs. The logic can therefore be considered as giving a new semantics for idempotent semiring-based CSPs.

Acknowledgements. This work was done mainly while I was at the Department of Computer Science, Keele University, UK. It was supported by the REVIGIS project, IST-1999-14189, and by Science Foundation Ireland under Grant 00/PI.1/C075. I am grateful also to the referee for their helpful comments.

References

1. Benferhat, S., Lagrue, S., Papini, O., Reasoning with partially ordered information in a possibilistic framework, *Proc. of the Ninth International Conference, Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU'2002)*; long version to appear in *Fuzzy Sets and Systems*.
2. Bistarelli, S., Montanari, U., Rossi, F., Semiring-based Constraint Solving and Optimization, *Journal of the ACM*, vol. 44, no. 2, 201–236 (1997).
3. Bistarelli, S., Montanari, U., Rossi, F., Semiring-Based Constraint Logic Programming: Syntax and Semantics, *ACM Transactions on Programming Languages and Systems*, ACM Press New York, USA, Pages: 1–29 Vol. 23, issue 1, (2001).
4. Bistarelli, S., Montanari, U., Rossi, F., Soft Concurrent Constraint Programming, *Proc. ESOP*, April 6–14, 2002, Grenoble, France Proceedings Springer LNCS vol. 2305, (2002).
5. Dubois, D., Fargier, H., Prade, H., The Calculus of Fuzzy Restrictions as a Basis for Flexible Constraint Satisfaction, *Proc. IEEE Int. Conf. on Fuzzy Systems*, IEEE (1993).
6. Dubois, D., Lang, J., Prade, H., Timed Possibilistic Logic, *Fundamenta Informaticae*, XV: 211–234 (1991).
7. Dubois, D., Lang, J., Prade, H., Possibilistic Logic, *Handbook of Logic in Artificial Intelligence and Logic Programming*, D. Gabbay, C. Hogger, J. Robinson (eds.), Vol. 3, 439–513, Oxford University Press (1994).
8. Fargier, H., Lang, J., Uncertainty in Constraint Satisfaction Problems: a Probabilistic Approach, *2nd European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU-93)*, Kruse, R., Clarke, M. and Moral, S., (eds.), Springer Verlag, 97–104, (1993).
9. Freuder, E., and Wallace, R., Partial Constraint Satisfaction, Artificial Intelligence, 58, 1–3, 21–70 (1992).
10. Mackworth, A. K., The Logic of Constraint Satisfaction, *Artificial Intelligence* 58, 3–20, (1992).
11. Dechter, R., Bucket Elimination: A Unifying Framework for Reasoning, *Artificial Intelligence*, 113, 41–85 (1999).
12. Dechter, R., and Pearl,J., Network-based Heuristics for Constraint Satisfaction Problems, *Artificial Intelligence*, 34, 1–38, (1987).
13. Kohlas, J., and Shenoy, P., *Volume 5: Algorithms for Uncertainty and Defeasible Reasoning* J. Kohlas, S. Moral (eds.), *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, (Series eds.: D. Gabbay, P. Smets), Kluwer Academic Publishers (2000).
14. Shenoy, P. P., and Shafer, G., Axioms for Probability and Belief Function Propagation, in *Uncertainty in Artificial Intelligence 4*, R. D. Shachter, T. S. Levitt, L. N. Kanal, J. F. Lemmer (eds.), North Holland, also in *Readings in Uncertain Reasoning*, Shafer, G., and Pearl, J., (eds.) Morgan Kaufmann, San Mateo, California, 575–610 (1990).
15. Shenoy, P. P., Valuation-Based Systems: A Framework for Managing Uncertainty in Expert Systems, pages 83–104, in *Fuzzy Logic for the Management of Uncertainty*, Zadeh, L.A., & Kacprzyk, J., (eds.), John Wiley & Sons (1992).