A Logic of Soft Constraints based on Partially Ordered Preferences

Nic Wilson

Cork Constraint Computation Centre,
Department of Computer Science,
University College Cork, Cork, Ireland,
n.wilson@4c.ucc.ie

Abstract. Representing and reasoning with an agent's preferences is important in many applications of constraints formalisms. Such preferences are often only partially ordered. One class of soft constraints formalisms, semiring-based CSPs, allows a partially ordered set of preference degrees, but this set must form a distributive lattice; whilst this is convenient computationally, it considerably restricts the representational power. This paper constructs a logic of soft constraints where it is only assumed that the set of preference degrees is a partially ordered set, with a maximum element 1 and a minimum element 0. When the partially ordered set is a distributive lattice, this reduces to the idempotent semiring-based CSP approach, and the lattice operations can be used to define a sound and complete proof theory. A generalised possibilistic logic, based on partially ordered values of possibility, is also constructed, and shown to be formally very strongly related to the logic of soft constraints. It is shown how the machinery that exists for the distributive lattice case can be used to perform sound and complete deduction, using a compact embedding of the partially ordered set in a distributive lattice.

1 Introduction

Representing and reasoning with an agent's preferences is important in many applications of constraints formalisms, so that more appropriate solutions can be generated. Preferences are often most naturally only partially ordered, reflecting an agent being unable or unwilling to order certain choices, or wishing to delay making such an ordering decision. This is reflected in many mathematical models of preferences; for example, in CP-nets [10, 9], where the associated ordering on complete assignments is almost never a total order (see, in particular, Proposition 4 of [23]). Most soft constraints formalisms assume a total order on the degrees of preference. On the other hand, the semiring-based CSP framework [5] does allow a partially ordered set of preference degrees, but this partially ordered set must form a distributive lattice (when combination is idempotent); whilst this is convenient computationally, it considerably restricts the representational power (see Section 4), since a distributive lattice is a very special type of partial order.

In Section 2 a logic of soft constraints is constructed where it is only assumed that the set of preference degrees is a partially ordered set, with a maximum element 1 and a minimum element 0. A soft constraint assigns a preference degree to a tuple, which is interpreted as an upper bound for the overall preference degree of the tuple. Semantic consequence is defined in Section 2.2, where models assign degrees of preferences to complete assignments. In this interpretation, the variables are assumed to be controllable, so that the agent can choose their values. A different interpretation of such models is given if the variables are not controllable, but describe the possible states of an unknown situation. They can then be considered as generalised possibility distributions, where possibility values are only partially ordered, so the models describe partially ordered degrees of 'possibility' of states of a system. In Section 2.3, a generalised possibilistic logic is constructed, which can be embedded in the logic of soft constraints.

When the partially ordered set is a distributive lattice, consequence in this logic of soft constraints is the same as consequence in an idempotent semiring-based CSP approach, and the lattice operations can be used to define a sound and complete proof theory, as shown in Section 3. Therefore computational techniques for idempotent semiring-based CSPs can be used for consequence in the logic of soft constraints for the distributive lattice case.

In Section 4 the general partially ordered case is considered. The idea behind computation of consequence in the general case is to embed the partially ordered set in a distributive lattice, and then use the embedded soft constraints to perform the deduction. Section 4.1 shows what properties allow this to work, and Section 4.2 and Section 4.3 show how to construct such an embedding, which is minimal in a certain sense for the case of a lattice. Section 4.4 discusses the problems of answering queries and generating simple derived soft constraints.

The semantics for the soft constraints given in Section 2.2 is natural, but is not the only choice. Section 5 describes an alternative semantics, and its semantic consequence relation for partially satisfied constraints. The issue of finding good and optimal complete assignments is briefly discussed in Section 6.

This paper includes and extends work in [22, 24].

2 A Logic Of Soft Constraints

In this section a logical formalism for reasoning with soft constraints is developed. Section 2.1 defines a soft constraint to be a function which maps an assignment (of a set of variables) to an element of a partially ordered set. A very simple semantics is defined in Section 2.2, which leads to the consequence operator $\models$ which expresses which soft constraints are implied by a set of soft constraints. The logical approach to soft constraints developed here contrasts with the algebraic approach taken, for example, in the semiring-based CSP framework [5, 8], and in the valued CSP framework [19, 8]. In the AI literature, reasoning formalisms are very often defined as logics, and this has a number of advantages. A benefit of our logical approach is that the notion of consequence follows from the simple semantics, defined in Section 2.2; this demonstrates that the defini-

tion of consequence is not arbitrary. Taking the logical approach also makes the comparison easier with other AI reasoning formalisms. In particular, we show in Section 2.3 how the formalism can also be considered in terms of a partially ordered possibilistic logic.

2.1 Partially Ordered Degrees of Preference

Let V be a finite set of variables, where each variable $V_i \in V$ has finite domain $D(V_i)$. Unless otherwise stated, the variables in V should all be thought of as decision variables, that is, the value of a variable can be chosen; this is in contrast to a situation (as discussed in Section 2.3) where the value of the variable reflects the state of a system which we are observing but have no control over. For $U \subseteq V$, define $D(U)$ to be the set $\prod_{V_i \in U} D(V_i)$ of possible assignments to variables U (i.e., the set of functions on U which, for each $V_i \in U$, assign a value in $D(V_i)$ to variable V_i). A complete tuple x is an element of $D(V)$. For $W \subseteq U \subseteq V$ and $u \in D(U)$ let $u^{\downarrow W}$ be the projection of u to variables W, given by: for $V_i \in W$, $u^{\downarrow W}(V_i) = u(V_i)$.

The intention is to produce a formalism that allows degrees of preference (or, alternatively, satisfaction, or adequacy) for tuples. So we choose a finite partially ordered set $\mathcal{A} = (A, \preceq, 0, 1)$ to represent these degrees, where A contains[1] a maximum element 1 and a minimum element 0 (with $0 \neq 1$). $\preceq$ is a partial order, so is reflexive, transitive and such that if $\alpha \preceq \beta$ and $\beta \preceq \alpha$ then $\alpha = \beta$. Associated relation $\prec$ is defined to be the strict part of $\preceq$, i.e., $\alpha \prec \beta$ if and only if $\alpha \preceq \beta$ and $\alpha \neq \beta$. Symbols $\succeq$ and $\succ$ are also used sometimes in the obvious way, e.g., $\alpha \succeq \beta$ holds if and only if $\beta \preceq \alpha$.

For finite A, the partially ordered set $\mathcal{A} = (A, 0, 1, \preceq)$ is a *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 $\gamma \preceq \alpha \wedge \beta \preceq \alpha, \beta$), and a least upper bound $\alpha \vee \beta$ in A (so that $\alpha, \beta \preceq \gamma$ implies $\alpha, \beta \preceq \alpha \vee \beta \preceq \gamma$); it is said to be a *distributive lattice* if the following distributivity property is satisfied: 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 given in Section 4).

Define an $\mathcal{A}$-constraint c to be a function from $D(V_c)$ to A, for some set of variables $V_c \subseteq V$. For complete assignment $x \in D(V)$, $\mathcal{A}$-constraint c can be applied to x by first projecting x to V_c; we use $c(x)$ as an abbreviation for $c(x^{\downarrow V_c})$. A value of 1 is intended to mean that the tuple is maximally preferred (or completely satisfies the constraint), and a value of 0 means that the tuple is least preferred (doesn't satisfy the constraint at all). Ordinary (hard) constraints can be represented by a special kind of $\mathcal{A}$-constraint: those c which assign only

[1] Assuming that there is a maximum element and a minimum element in the partially ordered set is not really restrictive, as we can add such elements to any partially ordered set (without necessarily involving them in any $\mathcal{A}$-constraint).

values in $\{0, 1\}$. 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 $y \in D(V_c)$, $c(y) \preceq d(y)$.

Example 1. We are trying to organise a meeting between Emily, Gerard and Pat, who each have their preferences about location, type of hotel, date, and so on. We want to ensure that the decision made satisfies at least two of the three people. Define A to be the set $\{0, \overline{e}, \overline{g}, \overline{p}, 1\}$, with 1 meaning all three are happy with the choice, 0 meaning that at least two are not happy, and so the choice is inadequate, and e.g., $\overline{e}$ meaning that Emily is not happy but the other two are. The order $\prec$ is defined by $0 \prec \overline{e}, \overline{g}, \overline{p} \prec 1$, so that $\overline{e}$, $\overline{g}$ and $\overline{p}$ are incomparable.

Gerard wants the meeting not to be in March. This is represented by an $\mathcal{A}$-constraint c on the date variable(s) with $c(D) = \overline{g}$ if D is any date in March. From this we can deduce that, if x is a complete assignment where the date is in March, then the preference degree for x is at best $\overline{g}$; it may be worse (i.e, 0) since we may have another $\mathcal{A}$-constraint implying a worse value. Therefore $\mathcal{A}$-constraints can be seen to be giving upper bounds on the preference degrees of complete assignments to the variables. (This is the basis of the semantics defined in the next section.)

As illustrated by this example, it is natural sometimes to allow incomparable degrees of preference, and hence allow sets of degrees of preferences which are not totally ordered; a user may not be comfortable in having to say that either (i) x is preferred to y, or (ii) y is preferred to x, or (iii) x and y are equally preferred.

In the formalism developed in this paper, arbitrary partially ordered sets of preferences are allowed; on the other hand, in the semiring-based CSPs framework [5], the partially ordered set of preferences is assumed to form a distributive lattice. This assumption is apparently very restrictive since many natural partially ordered sets are not distributive lattices, and a distributive lattice is a very special kind of partial order. In particular, one simple kind of partial order $\preceq$ on A which is, in some sense, close to being a total order, is when $\prec$ is a strict weak order, corresponding to a total order over a partition of A. This is where A can be partitioned into non-empty sets $E_i : i = 0, \ldots, k$, with $E_0 = \{0\}$, $E_k = \{1\}$, and if $\alpha \in E_i$ and $\beta \in E_j$ then $\alpha \prec \beta$ if and only if $i < j$. Such an order is a [distributive] lattice only under special conditions. It forms a lattice only if for $i = 1, \ldots, k-1$, $|E_i| > 1$ implies $|E_{i-1}| = |E_{i+1}| = 1$. It only forms a distributive lattice if in addition, for all i, $|E_i| \leq 2$. To illustrate further, consider all partially ordered sets $\mathcal{A} = (A, \preceq, 0, 1)$ where A has just seven elements (including 0 and 1). There are more than sixty such (essentially different) partially ordered sets, with only eight of them being distributive lattices, and with ten of them not being lattices.

Sets of preferences forming a distributive lattice are natural in many situations. For example, in a multi-criteria decision making context, we may use the 'pareto' or 'product' order given by: x is preferred to y if it is at least as good on each criterion.

In Example 1, $\mathcal{A} = (A, \preceq, 0, 1)$, is a lattice, but not a distributive lattice, since e.g., $(\overline{e} \vee \overline{g}) \wedge \overline{p} = 1 \wedge \overline{p} = \overline{p}$, whereas $(\overline{e} \wedge \overline{p}) \vee (\overline{g} \wedge \overline{p}) = 0 \vee 0 = 0$. One

might add elements e, g and p to A to give $A' = \{0, e, g, p, \overline{e}, \overline{g}, \overline{p}, 1\}$, with e.g., e meaning that only Emily is happy with the choice. The order is extended by $e \prec \overline{g}, \overline{p}$; and $g \prec \overline{e}, \overline{p}$; and $p \prec \overline{e}, \overline{g}$. We now have a distributive lattice (but at the cost of increasing the size of the set) with the order being essentially the subset order (with e.g., $\overline{e}$ being associated with set $\{g, p\}$), which is a special case of a 'product' order.

The product order is somewhat weak (especially when there are several criteria), and it is often natural to add to the order, in particular, to represent tradeoffs. For example, we may decide that 'only Pat being unhappy with the choice' is worse than 'only Gerard being unhappy', and hence add the condition $\overline{p} \prec \overline{g}$ (and form the transitive closure). A' is no longer a lattice under this new order since $\overline{g}$ and $\overline{e}$ now have no greatest lower bound.

More generally, partially ordered sets of preferences which are not distributive lattices arise naturally from either adding extra orderings to a distributive lattice, or from merging elements of the set (as in Example 1).

2.2 Semantics

We would like to say what $\mathcal{A}$-constraints d can be deduced from a set of $\mathcal{A}$-constraints C. We imagine that there exists some (unknown) function $M : D(V) \to A$ which gives the degree of preference of each complete tuple $x \in D(V)$. An $\mathcal{A}$-constraint c is interpreted as telling us that for all $x \in D(V)$, the preference degree $M(x)$ of x is bounded above by $c(x)$, i.e.,

$$M(x) \preceq c(x). \tag{1}$$

(As usual in constraints formalisms, a constraint c can be considered as a compact representation of a constraint on the whole set V of variables.) An alternative intuition which leads to the same formal system is where we imagine that the constraints are for a particular purpose, and $M(x)$ gives the true degree of adequacy of tuple x for that purpose. $\mathcal{A}$-constraints then give upper bounds on the degrees of adequacy. Each constraint c in our set C is taken to restrict the possible models M, via equation (1). For function $M : D(V) \to A$ we say $M \models c$ (M *satisfies* c, or M *is a model of* c) if for all $x \in D(V)$, $M(x)$ is bounded above by $c(x)$.

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. Soft constraints C express information about the preferences (of e.g., an agent), and so such a d expresses derived preference information, which may be thought of as a property of the agent's preferences. An alternative notion of semantic consequence is explored in Section 5, based on a similar intuition, but allowing a model M to take values in a partially ordered set that extends $\mathcal{A}$. The consequence relation $\models$ on $\mathcal{A}$-constraints is, by the form of its construction, reflexive, monotonic and transitive. If C is a singleton set $\{c\}$, we may write $c \models d$ instead of $\{c\} \models d$.

The semantic definition is not very helpful for computing the consequences d of C; we need a more computationally useful characterisation. To do this we

consider a special case first, in Section 3, when $\mathcal{A}$ is a lattice, and use that special case in computing the general case, in Section 4.

First, we express consequence in a slightly different way. A *lower bound* (with respect to $\preceq$) of an element $\gamma \in A$ is an element $\alpha \in A$ with $\alpha \preceq \gamma$. A *lower bound of a subset* B of A is an element $\alpha \in A$ which is a lower bound for each element of B; we then write $\alpha \preceq B$. (Every element is a lower bound of the empty set.) For $B \subseteq A$ and $\gamma \in A$ we write $B \trianglelefteq \gamma$ if every lower bound of (every element of) B is a lower bound of γ, i.e., for all $\alpha \in A$, $[\alpha \preceq B$ implies $\alpha \preceq \gamma]$.

For $x \in D(V)$ define $C(x)$ to be the set $\{c(x) : c \in C\}$. The following proposition expresses deduction in terms of the relation $\trianglelefteq$. (Recall $d(x)$ is defined to be $d(x^{\downarrow V_d})$.)

Proposition 1. *With the above notation, $C \models d$ if and only if for all $x \in D(V)$, $C(x) \trianglelefteq d(x)$.*

Proof. To prove the 'if' part: if $C(x) \trianglelefteq d(x)$ and $M \models C$ then for all $x \in D(V)$, $M(x) \preceq C(x)$ so by the definition of $\trianglelefteq$, $M(x) \preceq d(x)$; this shows $M \models d$, proving $C \models d$. For the converse, suppose that it is not the case that for all $x \in D(V)$, $C(x) \trianglelefteq d(x)$. Then there exists $x_0 \in D(V)$ and $\alpha \in A$ such that $\alpha \preceq C(x_0)$ but $\alpha \not\preceq d(x_0)$. Define model M by $M(x_0) = \alpha$ and for all $x \neq x_0$, $M(x) = 0$. Then $M \models C$ but $M \not\models d$, proving that $C \not\models d$. □

Complexity of Deduction. Let $\mathcal{A} = (A, \preceq, 0, 1)$ be a partially ordered set. The Deduction Problem for $\mathcal{A}$-constraints is the following problem: *Given set of $\mathcal{A}$-constraints $C \cup \{d\}$ over some set of variables V, determine if $C \models d$.*

It is shown below that the complexity of deduction is in the same complexity class as deduction in propositional logic, and deduction for finite CSPs; adding the preference values does not put the problem into a harder complexity class. We assume that for any instance, for any $x \in D(V)$, and any $\alpha \in A$, the condition $\alpha \preceq c(x)$ can be tested in polynomial time. (The size of an instance depends on the number of variables and the size of the representation of the input $\mathcal{A}$-constraints, but does not depend on the size of A.) The result is quite general: the constraints can be represented extensionally or intensionally, as can the elements of A.

Proposition 2. *Let $\mathcal{A} = (A, \preceq, 0, 1)$ be a partially ordered set. The Deduction Problem for $\mathcal{A}$-constraints is* coNP*-complete.*

Proof. To prove this result we show that the complement of the problem (determining if $C \not\models d$) is NP-complete. First we show that the complement of the problem is in NP. Let $C \cup \{d\}$ be a set of $\mathcal{A}$-constraints. Let $x \in D(V)$ be a complete assignment, and let α be an element of A. Say that pair (x, α) *succeeds* if $d(x) \not\succeq \alpha$ and for all $c \in C$, $c(x) \succeq \alpha$. Testing if (x, α) succeeds or not can be done in polynomial time. By Proposition 1, $C \not\models d$ if and only if there exists some pair (x, α) that succeeds. So a nondeterministic algorithm [16] determining

if $C \not\models d$ is given by guessing $x \in D(V)$ and $\alpha \in A$, and verifying whether (x, α) succeeds. This shows that the complement of the problem is in NP.

To show that the complement of the problem is NP-hard, we use a reduction from 3SAT. Consider an instance of 3SAT over set of propositional variables V. Each clause, involving variables U, can be mapped to an $\mathcal{A}$-constraint c with scope U, and with $c(y) = 0$ if y does not satisfy the clause, and $c(y) = 1$ otherwise. Let C be this set of $\mathcal{A}$-constraints. Define d to be the nullary $\mathcal{A}$-constraint equal to constant 0. Define model M' by $M'(x) = 1$ for any x satisfying the set of clauses, and $M'(x) = 0$ otherwise. It can be shown that a model $M : D(V) \to A$ satisfies C if and only if for all $x \in D(V)$, $M(x) \leq M'(x)$. Therefore the set of clauses is satisfiable if and only if there exists a model of C which is not uniformly 0, which is if and only if $C \not\models d$ (since the only model of d is uniformly 0). The fact that 3SAT is NP-hard then implies that the complement of the deduction problem for $\mathcal{A}$-constraints is NP-hard. □

2.3 Possibilistic Logic based on a Partially Ordered Set

This section defines a possibilistic logic [14] based on partially ordered sets, and shows how this can be embedded in the logic defined in Section 2.2. (A weaker possibilistic logic can also be produced in a similar fashion, but instead based on the semantics described in Section 5; this also reduces to Standard Possibilistic Logic in the total order case.)

Let $\mathcal{A} = (A, \preceq, 0, 1)$ be a partially ordered set. Define a bijection $\alpha \mapsto \hat{\alpha}$ from A to some set $\hat{A}$ which is disjoint from A. We order elements $\hat{A}$ in the opposite way compared to A: $\hat{\alpha} \preceq \hat{\beta}$ if and only if $\alpha \succeq \beta$. Elements of $\hat{A}$ are used as degrees of certainty, or 'Necessity'. Let V be a set of variables. In contrast to the situation described in Section 2.2, these are not decision variables that we have control over, but variables representing the unknown state of a system which we are observing: (this generalised) possibilistic logic is therefore used for deducing further information in an uncertain environment.

The language $\mathcal{L}$ of this logic consists of statements of the form $\mathrm{Nec}(\varphi, \hat{\alpha})$, for some φ and $\alpha \in A$. φ is a formula involving variables V_φ which corresponds to a set Ω_φ of partial assignments to variables V_φ: the assignments that satisfy φ. (In the case of propositional variables, φ is a propositional formula.) $\mathrm{Nec}(\varphi, \hat{\alpha})$ is to be considered as saying that the 'Necessity' (a 'degree of certainty') of φ is at least $\hat{\alpha}$. Possibility distributions, which are the models of this logic, are functions π from $D(V)$ to A. Possibility distribution π satisfies $\mathrm{Nec}(\varphi, \hat{\alpha})$ if and only if $\pi(x) \preceq \alpha$ for all $x \in D(V)$ such that x doesn't satisfy φ, i.e., such that $x^{\downarrow V_\varphi} \notin \Omega_\varphi$. Let $\Delta \subseteq \mathcal{L}$ be a set of such statements. Then $\Delta \models \mathrm{Nec}(\varphi, \hat{\alpha})$ if and only if π satisfies $\mathrm{Nec}(\varphi, \hat{\alpha})$ for all π satisfying every element of Δ. When $\preceq$ is a total order on A (and the variables are all propositional variables) this is precisely Standard Possibilistic Logic [14]. Thus the above logic generalises Standard Possibilistic Logic to being over a partially ordered set. (A distributive-lattice-valued possibilistic logic is described in [13, 14]. In contrast, [1] generalises

possibility-based inference relations (on ordinary propositions) to allow partially ordered possibility.)

This logic can be embedded in the logic defined in Section 2.2 (and in fact can be considered as equivalent to it), using the fact that possibility distributions, as just defined, are exactly the same as models of $\mathcal{A}$-constraints defined in Section 2.2. For φ and α define $\mathcal{A}$-constraint $c_{(\varphi,\alpha)}$ as follows: $V_{c_{(\varphi,\alpha)}} = V_\varphi$; for $y \in D(V_\varphi)$, $c_{(\varphi,\alpha)}(y) = \alpha$, if $y \notin \Omega_\varphi$, and $c_{(\varphi,\alpha)}(y) = 1$ if $y \in \Omega_\varphi$. For $\Delta \subseteq \mathcal{L}$, let C_Δ be the set of $\mathcal{A}$-constraints $c_{(\varphi,\alpha)}$ for $\mathrm{Nec}(\varphi, \hat{\alpha})$ in Δ.

Proposition 3. *With the above definitions, $\Delta \models Nec(\varphi, \hat{\alpha})$ if and only if $C_\Delta \models c_{(\varphi,\alpha)}$.*

Proof. We have: $\pi \models c_{(\varphi,\alpha)}$ if and only if for all $x \in D(V)$, $\pi(x) \preceq c_{(\varphi,\alpha)}(x)$, which is if and only if $\pi(x) \preceq \alpha$ for all $x \in D(V)$ with $x^{\downarrow V_\varphi} \notin \Omega_\varphi$, which is if and only if π satisfies $\mathrm{Nec}(\varphi, \hat{\alpha})$. Therefore π satisfies every element in Δ if and only if $\pi \models C_\Delta$. So $\Delta \models \mathrm{Nec}(\varphi, \hat{\alpha})$ if and only if π satisfies $\mathrm{Nec}(\varphi, \hat{\alpha})$ for all π satisfying Δ which is if and only if $\pi \models c_{(\varphi,\alpha)}$ for all $\pi \models C_\Delta$ which is if and only if $C_\Delta \models c_{(\varphi,\alpha)}$. □

Therefore the computational techniques for deduction of $\mathcal{A}$-constraints developed in this paper can be used for deduction in this possibilistic logic over a partially ordered set.

3 Consequence when $\mathcal{A}$ is a Lattice

We first look at determining deductions of the form $C \models d$ when $\mathcal{A}$ is a lattice. The lattice properties enable us to define combination and projection of $\mathcal{A}$-constraints. 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 $y \in D(V_c \cup V_d)$, $(c \wedge d)(y) = c(y^{\downarrow V_c}) \wedge d(y^{\downarrow V_d})$. Combination of $\mathcal{A}$-constraints is commutative and associative, so we can define $\bigwedge C$ to be the combination of all $\mathcal{A}$-constraints in finite set C. Then $V_{\bigwedge C} = \bigcup_{c \in C} V_c$, and $(\bigwedge C)(y) = \bigwedge_{c \in C} c(y^{\downarrow V_c})$ for $y \in D(V_{\bigwedge C})$. For $U \subseteq V_c$, $c^{\downarrow U}$, the projection of c to U, is given by: for $u \in D(U)$, $c^{\downarrow U}(u) = \bigvee \{c(y) : y \in D(V_c), y^{\downarrow U} = u\}$. Define the $\mathcal{A}$-constraint 1_U on variables U to be everywhere equal to 1: for all $u \in D(U)$, $1_U(u) = 1$.

3.1 Properties of Semantic Consequence

The following lemma gives some basic properties. Part (i) follows from the definition. Part (ii) follows from the definition of greatest lower bound. (iii) follows e.g., using Proposition 1. (iv) follows from (iii), as does (v). One half of (vi) follows from (v) and transitivity of $\models$. For the other half, suppose $c \models d$ and let $z \in D(V_d)$. Then, using (iii), for all $y \in D(V_c)$ such that $y^{\downarrow V_d} = z$, $c(y) \preceq d(z)$, so $c^{\downarrow V_d}(z) = \bigvee \{c(y) : y^{\downarrow V_d} = z\} \preceq d(z)$, proving $c^{\downarrow V_d} \models d$ using (iv).

Lemma 1. *Let $\mathcal{A} = (A, \preceq, 0, 1)$ be a lattice, let M be a model for $\mathcal{A}$-constraints, let c and d be $\mathcal{A}$-constraints, let C be a set of $\mathcal{A}$-constraints, and let $U \subseteq V$ be a set of variables. Then: (i) $M \models 1_U$. (ii) $M \models C$ if and only if $M \models \bigwedge C$. (iii) $c \models d$ if and only if for all $x \in D(V)$, $c(x) \preceq d(x)$. (iv) Given $V_c = V_d$, then $c \models d \iff c \preceq d$. (v) If $U \subseteq V_c$ then $c \models c^{\downarrow U}$. (vi) Given $V_c \supseteq V_d$ then $c \models d \iff c^{\downarrow V_d} \models d$.*

The following result gives a useful characterisation of semantic consequence.

Theorem 1. *Let $\mathcal{A} = (A, \preceq, 0, 1)$ be a lattice, and let $C \cup \{d\}$ be a set of $\mathcal{A}$-constraints. Then $C \models d$ if and only if $\bigwedge C \models d$ if and only if $(\bigwedge C \wedge 1_{V_d})^{\downarrow V_d} \preceq d$. Now suppose $\bigcup_{c \in C} V_c \supseteq V_d$. Then $C \models d$ if and only if $(\bigwedge C)^{\downarrow V_d} \preceq d$.*

Proof. Let M be a model. By (ii) and (i) of Lemma 1, $M \models C$ if and only if $M \models \bigwedge C$ if and only if $M \models \bigwedge C \wedge 1_{V_d}$. This implies that $C \models d$ if and only if $\bigwedge C \models d$ if and only if $\bigwedge C \wedge 1_{V_d} \models d$. Part (vi) and (iv) then imply that this is 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 (vi) and (iv) implies that $\bigwedge C \models d$ holds if and only if $(\bigwedge C)^{\downarrow V_d} \preceq d$ holds. □

3.2 Proof Theory for the Lattice Case

A simple sound and complete proof theory can be defined using the combination and projection operations. Define the proof theory by the following axioms and inference rules:

Axioms: For all $U \subseteq V$, 1_U.

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.

A proof of d from C is a finite sequence of $\mathcal{A}$-constraints $d_1, \ldots, d_k = d$ such that each d_i is either (i) an element of C; (ii) an axiom; (iii) the result of applying one of the inference rules to earlier elements in the sequence. The following lemma gives the properties needed for Soundness. It follows easily from parts (i), (ii), (v) and (iv) of Lemma 1.

Lemma 2. *Let $\mathcal{A} = (A, \preceq, 0, 1)$ be a lattice and let $C \cup \{d\}$ be a set of $\mathcal{A}$-constraints. Then: (i) If $c \in C$ then $C \models c$; (ii) For all $U \subseteq V$, $C \models 1_U$; (iii) If $C \models c$ and $C \models d$ then $C \models c \wedge d$; (iv) If $C \models c$ and $U \subseteq V_c$ then $C \models c^{\downarrow U}$; (v) If $V_c = V_d$ and $c \preceq d$ and $C \models c$ then $C \models d$.*

This proof theory is sound and complete:

Theorem 2 (Soundness and Completeness). *Let $\mathcal{A} = (A, \preceq, 0, 1)$ be a lattice and let $C \cup \{d\}$ be a set of $\mathcal{A}$-constraints on some set of variables V. Then $C \models d$ if and only if d can be proved from C using the axiom and inference rules.*

Proof. Soundness follows from Lemma 2 in the usual way, by induction on the length of the proof of d from C. To prove Completeness: suppose $C \models d$. 1_{V_d} can be proved from C since it's an axiom. By repeated use of the combination inference rule we can prove $\bigwedge C$ from C, and so also $\bigwedge C \wedge 1_{V_d}$. The projection inference rule can be used to then prove $(\bigwedge C \wedge 1_{V_d})^{\downarrow V_d}$. By Theorem 1 we have $(\bigwedge C \wedge 1_{V_d})^{\downarrow V_d} \preceq d$, so the last inference rule allows us to deduce d as required. □

3.3 Relationship with Idempotent Semiring-based CSPs

It is clear that there is a strong relationship with idempotent semiring-based CSPs (Bistarelli et al, 97) [5], as the combination and projection operations defined above are the same as those used in 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 [5] 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 $Sol(\langle C, U\rangle) = (\bigwedge C)^{\downarrow U}$. There is a natural definition for deduction of constraints in this system (cf. [7]). Let $C \cup \{d\}$ be a set of $\mathcal{A}$-constraints such that $\bigcup_{c\in C} V_c \supseteq V_d$. We can say that d is a consequence of C if $Sol(\langle C, V_d\rangle) \preceq d$. Theorem 1 implies that this is if and only if $C \models d$. Therefore consequence for idempotent semiring-based CSPs can be considered as the special case of consequence for $\mathcal{A}$-constraints when $\mathcal{A}$ is a distributive lattice.

The semantics defined in Section 2 differs in a fundamental way from the model-theoretic semantics for semiring-based Constraint Logic Programming given in (Bistarelli et al, 2001) [6]. In the semantics described above, a constraint gives an upper bound on the model (and can be viewed as conveying negative information). Consequently, there exists a unique *maximal* model of a set of constraints C, i.e., $1_V \wedge \bigwedge C$. In contrast, in the logic programming-based semantics of [6], constraints are expressed using a set of facts representing 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 of the set of constraints. Which of the two semantics is more appropriate will depend on the application.

Non-idempotent semiring-based CSPs. A natural question is whether our logical approach can be extended to also cover the case of semiring-based CSPs where combination is not idempotent. It is clear that major changes to the set up would be needed. To begin with we would need to consider *multisets* C (rather than sets) of $\mathcal{A}$-constraints. Also equation (1) can no longer be the basis of the

semantics, since whether a model satisfies multiset C cannot just depend on it satisfying each element of C.

3.4 Computational Techniques for the Distributive Lattice Case

The connection shown above means that, for the case when $\mathcal{A}$ is a distributive lattice, we can use computational approaches for idempotent semiring-based CSPs to determine consequences of sets of $\mathcal{A}$-constraints. For example, approaches such as in [5, 6, 4, 3], or a decision diagram approach [25]. Also, a general method for this kind of problem is based on variable elimination.

Variable Elimination An important derived inference rule is elimination of a variable. Variable $V_i \in V$ is eliminated as follows: we combine the set C_i of $\mathcal{A}$-constraints involving that variable and project away from that variable. Formally, C_i is $\{c \in C : V_c \ni V_i\}$, and let $U = \bigcup_{c \in C_i} V_c$, be the set of variables involved in the combination of C_i. Lemma 2 implies that the following inference rule is sound: *From C infer the $\mathcal{A}$-constraint* $C^i = (\bigwedge C_i)^{\downarrow U - \{V_i\}}$. This can also be used to build a complete deduction mechanism (for certain important types of queries). When $\mathcal{A}$ is a distributive lattice, the combination and projection operations on $\mathcal{A}$-valuations obey the Shenoy-Shafer axioms [21, 18]: see Theorems 18 and 19 of (Bistarelli et al, 97) [5]. This implies that variable elimination algorithms can be used to compute projections of combinations. To illustrate, define $C^{\downarrow -V_i}$, the result of eliminating variable V_i, to be C when C_i is replaced by C^i, i.e., $C^{\downarrow -V_i} = C - C_i \cup \{C^i\}$. It can be shown, using the Shenoy-Shafer axioms, that if $T \subseteq V$ is such that $T \not\ni V_i$ then $(\bigwedge C)^{\downarrow T} = \left(\bigwedge C^{\downarrow -V_i}\right)^{\downarrow T}$. So, if $V_d \not\ni V_i$ then by Theorem 1, $C \models d$ if and only if $C^{\downarrow -V_i} \models d$. We can sequentially eliminate all the variables in $V - V_d$ to obtain a set of $\mathcal{A}$-constraints C' involving only variables V_d and such that $C \models d$ if and only if $C' \models d$; hence we can determine if $C \models d$ by testing if $\bigwedge C' \preceq d$. This variable elimination algorithm is an instance of the fusion algorithm [20, 21, 17] (cf. also Dechter's Bucket Elimination [11], non-serial dynamic programming [2], and Section 5 of [5]).

This approach is efficient if one can find a hypertree cover [21] which doesn't involve too large sets, since then none of the combinations need involve too many variables. (These are the same conditions that ensure efficient computation in Bayesian networks.) If such a hypertree cover cannot be found, then one might instead use a mini-buckets approximation approach to the computations (Dechter and Rish, 2003) [12].

4 Deduction in 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 minimum element 0 and a maximum element 1). This is important because we cannot in general expect a partially ordered set of preference degrees to necessarily form a distributive lattice.

Our approach is to embed the partially ordered set in a lattice of subsets in such a way that the ordering information is maintained, but without adding any extra ordering information. Then we can use the computational techniques for the subset lattice case to make deductions for this general case.

Let $\mathcal{A} = (A, \preceq, 0, 1)$ be a finite partially ordered set, let Θ be another finite set, let Q be a function from A to 2^Θ, and let $\mathcal{B}$ be the partially ordered set $(2^\Theta, \subseteq, \emptyset, \Theta)$. The function Q can be used to map $\mathcal{A}$-constraints to $\mathcal{B}$-constraints. We extend Q to subsets of A by: for $B \subseteq A$, $Q(B) = \{Q(\beta) : \beta \in B\}$. For each soft constraint $c : D(V_c) \to A$ define $c^Q : D(V_c) \to 2^\Theta$ to be c followed by Q, so that $c^Q(y) = Q(c(y))$. For set C of $\mathcal{A}$-constraints, define C^Q to be $\{c^Q : c \in C\}$. So, for $x \in D(V)$, $C^Q(x) = \{c^Q(x^{\downarrow V_c}) : c \in C\}$, which equals $Q(C(x))$ (recall $C(x) = \{c(x) : c \in C\}$). Constraints c^Q are $\mathcal{B}$-constraints (as defined in Section 2).

Say that $Q : A \to 2^\Theta$ is *deduction-adequate* if for any set $C \cup \{d\}$ of $\mathcal{A}$-constraints (over some set of variables) the following equivalence holds:

$$C \models d \text{ if and only if } C^Q \models d^Q.$$

(Note that the latter $\models$ is the semantic relation for $\mathcal{B}$-models.) Therefore if we can find a deduction-adequate Q, then we can determine deductions from sets C of $\mathcal{A}$-constraints by mapping them using Q to sets C^Q of $\mathcal{B}$-constraints, and using the computational techniques available for the distributive lattice case, discussed above in Section 3.4. The main result of Section 4 is a method (in Section 4.3) for constructing deduction-adequate mapping $Q : A \to 2^\Theta$, where Θ is kept relatively compact, and is of minimum size when $\mathcal{A}$ is a lattice.

Section 4.1 gives some necessary and sufficient conditions for Q to be deduction-adequate. Section 4.2 firstly shows how to generate a deduction-adequate mapping with $\Theta = A$; this Θ is often unnecessarily large, however. For the case of a lattice, it is shown how to generate a deduction-adequate Q with Θ of minimum size. Section 4.3 describes a construction of a deduction-adequate mapping for the general partially ordered case, which generalises the construction for the lattice case. Section 4.4 considers applying these results to answering queries and finding simple derived constraints.

4.1 Properties for Deduction-Adequate Mappings

We say that Q is *order-preserving* if for all $\alpha, \beta \in A$, $\alpha \preceq \beta \iff Q(\alpha) \subseteq Q(\beta)$. Being order-preserving is a necessary condition (see Proposition 4(2) below) but not a sufficient one for Q to be deduction-adequate. The following result gives some necessary and sufficient conditions.

Proposition 4. *Let $\mathcal{A} = (A, \preceq, 0, 1)$ be a finite partially ordered set, let Θ be a finite set and let Q be a function from A to 2^Θ*

(1) Q is deduction-adequate if and only if the following holds:
for any $B \subseteq A$ and $\gamma \in A$, $\quad B \trianglelefteq \gamma \iff \bigcap_{\beta \in B} Q(\beta) \subseteq Q(\gamma)$.

(2) If Q is deduction-adequate then (a) Q is order-preserving; and (b) for any $\alpha, \beta \in A$, if α and β have a greatest lower bound γ then $Q(\alpha) \cap Q(\beta) = Q(\gamma)$.

(3) The following pair of conditions are sufficient for Q to be deduction-adequate: (i) Q is order-preserving; and (ii) for any $B \subseteq A$ and $\theta \in \bigcap_{\beta \in B} Q(\beta)$, there exists $\alpha \in A$ with $\theta \in Q(\alpha) \subseteq \bigcap_{\beta \in B} Q(\beta)$.

Proof. (1) First we show that for any $B \subseteq A$ and $\gamma \in A$, $Q(B) \trianglelefteq Q(\gamma)$ if and only if $\bigcap_{\beta \in B} Q(\beta) \subseteq Q(\gamma)$.

$W \subseteq \Theta$ is a lower bound for $Q(B) = \{Q(\beta) : \beta \in B\}$ if and only if for all $\beta \in B$, $W \subseteq Q(\beta)$, which is if and only if $W \subseteq \bigcap_{\beta \in B} Q(\beta)$. So $\{Q(\beta) : \beta \in B\} \trianglelefteq Q(\gamma)$ if and only if $W \subseteq Q(\gamma)$ for all W such that $W \subseteq \bigcap_{\beta \in B} Q(\beta)$, which is if and only if $\bigcap_{\beta \in B} Q(\beta) \subseteq Q(\gamma)$, since we can set $W = \bigcap_{\beta \in B} Q(\beta)$.

By Proposition 1, $C \models d$ if and only if for all $x \in D(V)$, $C(x) \trianglelefteq d(x)$. Similarly, $C^Q \models d^Q$ if and only if if for all $x \in D(V)$, $C^Q(x) \trianglelefteq d^Q(x^{\downarrow V_d})$, i.e., $Q(C(x)) \trianglelefteq Q(d(x))$.

Suppose, for any $B \subseteq A$ and $\gamma \in A$, $B \trianglelefteq \gamma \iff \bigcap_{\beta \in B} Q(\beta) \subseteq Q(\gamma)$, so by the argument above we have $B \trianglelefteq \gamma \iff Q(B) \trianglelefteq Q(\gamma)$. Let $C \cup \{d\}$ be a set of $\mathcal{A}$-constraints. Then, for any $x \in D(V)$, we can set $B = C(x)$ and $\gamma = d(x)$, which leads to [for all $x \in D(V)$, $C(x) \trianglelefteq d(x)$] $\iff$ [for all $x \in D(V)$, $Q(C(x)) \trianglelefteq Q(d(x))$]. So we have $C \models d$ if and only if $C^Q \models d^Q$. This proves that Q is deduction-adequate.

Conversely, suppose that there exists $B \subseteq A$ and $\gamma \in A$, such that it is not the case that $B \trianglelefteq \gamma \iff Q(B) \trianglelefteq Q(\gamma)$. Define all the elements of C and d to have empty scope (i.e., for all $e \in C \cup \{d\}$, $V_e = \emptyset$), so are constants, and define d to be the constant γ and C to be the set of constants B. Then $C \models d$ if and only if $B \trianglelefteq \gamma$ and $C^Q \models d^Q$ if and only if $Q(B) \trianglelefteq Q(\gamma)$, so for this choice of C and d it is not the case that $C \models d \iff C^Q \models d^Q$, showing that Q is not deduction-adequate.

(2)(a) Assume that Q is deduction-adequate. Then, by part (1), $\{\alpha\} \trianglelefteq \beta$ if and only if $Q(\alpha) \subseteq Q(\beta)$, which implies (2)(a) that Q is order-preserving, since $\{\alpha\} \trianglelefteq \beta$ holds if and only if $\alpha \preceq \beta$, by reflexivity and transitivity of $\preceq$.

(2)(b) $\gamma \preceq \alpha$ and $\gamma \preceq \beta$, so by part (a), $Q(\gamma) \subseteq Q(\alpha), Q(\beta)$, so $Q(\gamma) \subseteq Q(\alpha) \cap Q(\beta)$. Since γ is the greatest lower bound of α and β, we have $\{\alpha, \beta\} \trianglelefteq \gamma$. This implies by part (1), $Q(\alpha) \cap Q(\beta) \subseteq Q(\gamma)$, completing the proof.

(3) We assume conditions (i) and (ii) hold. By definition, $B \trianglelefteq \gamma$ holds if and only if $\alpha \preceq \gamma$ holds for all α such that $\alpha \preceq B$. Also, condition $\alpha \preceq B$ holds if and only if $\alpha \preceq \beta$ holds for all $\beta \in B$, which using (i) is if and only if for all $\beta \in B$, $Q(\alpha) \subseteq Q(\beta)$, which is if and only if $Q(\alpha) \subseteq \bigcap_{\beta \in B} Q(\beta)$. Thus $B \trianglelefteq \gamma$ holds if and only if the following condition $(*)$ holds:

$Q(\alpha) \subseteq Q(\gamma)$ for all $\alpha \in A$ such that $Q(\alpha) \subseteq \bigcap_{\beta \in B} Q(\beta)$.

Suppose $(*)$ holds, and consider any $\theta \in \bigcap_{\beta \in B} Q(\beta)$. By (ii), there exists $\alpha \in A$ with $\theta \in Q(\alpha) \subseteq \bigcap_{\beta \in B} Q(\beta)$, so $\theta \in Q(\gamma)$ by $(*)$, showing that $\bigcap_{\beta \in B} Q(\beta) \subseteq Q(\gamma)$. Conversely, suppose that $\bigcap_{\beta \in B} Q(\beta) \subseteq Q(\gamma)$. Then obviously if $Q(\alpha) \subseteq \bigcap_{\beta \in B} Q(\beta)$ then $Q(\alpha) \subseteq Q(\gamma)$, so condition $(*)$ holds.

We have shown that, given (i) and (ii), condition $(*)$ is equivalent to $\bigcap_{\beta \in B} Q(\beta) \subseteq Q(\gamma)$, and hence, $B \trianglelefteq \gamma$ if and only if $\bigcap_{\beta \in B} Q(\beta) \subseteq Q(\gamma)$, which implies, using

part (1), that Q is deduction-adequate. □

4.2 Deduction-Adequate $Q : A \to 2^{\Theta}$ with $\Theta \subseteq A$

In this section and the next it is shown how a deduction-adequate mapping $Q : A \to 2^{\Theta}$ can be constructed. As shown below in Proposition 5(iii), we can define Θ to be A and, for each α, $Q(\alpha)$ to be $\{\beta \in A : \beta \preceq \alpha\}$. However, if A is large, working with subsets of A can be computationally expensive. We go on to give a way of constructing an embedding which can lead to much smaller Θ than A, in this section for the special case of a lattice, and then more generally in Section 4.3

We will consider mappings Q of a particular form. Let A' be a subset of A. For $\alpha \in A$ let $Q_{A'}(\alpha)$ be the set $\{\alpha' \in A' : \alpha' \preceq \alpha\}$. The following result shows that for any A', the mapping $Q_{A'} : A \to 2^{A'}$ satisfies condition (ii) of Proposition 4(3), and half of (i).

Proposition 5. *Let A' be any subset of A. Then $Q_{A'}$ satisfies:*

(i)′ if $\alpha \preceq \beta$ then $Q_{A'}(\alpha) \subseteq Q_{A'}(\beta)$, and
(ii) for any $B \subseteq A$ and $\theta \in \bigcap_{\beta \in B} Q_{A'}(\beta)$, there exists $\alpha \in A$ with $\theta \in Q_{A'}(\alpha) \subseteq \bigcap_{\beta \in B} Q_{A'}(\beta)$.
(iii) Furthermore if $A' = A$ or $A' = A - \{0\}$ then $Q_{A'}$ is order-preserving and so deduction-adequate.

Proof. (i)′: Suppose $\alpha \preceq \beta$. If $\gamma \in Q_{A'}(\alpha)$ then $\gamma \in A'$ and $\gamma \preceq \alpha$, so, by transitivity of $\preceq$, $\gamma \preceq \beta$ and $\gamma \in Q_{A'}(\beta)$.

(ii): We'll use $\alpha = \theta$. Firstly, $Q_{A'}(\theta) \ni \theta$, since $\theta \in A'$ and $\theta \preceq \theta$. Secondly, for any $\beta \in B$, we have $\theta \in Q_{A'}(\beta)$ so $\theta \preceq \beta$. If $\delta \in Q_{A'}(\theta)$ then $\delta \in A'$ and $\delta \preceq \theta$ so $\delta \preceq \beta$ and $\delta \in Q_{A'}(\beta)$. Hence $\delta \in \bigcap_{\beta \in B} Q_{A'}(\beta)$. This shows that $Q_{A'}(\theta) \subseteq \bigcap_{\beta \in B} Q_{A'}(\beta)$.

(iii): Suppose $A' = A$ or $A' = A - \{0\}$, and $Q_{A'}(\alpha) \subseteq Q_{A'}(\beta)$. If $\alpha \neq 0$ then $\alpha \in A'$ so $\alpha \in Q_{A'}(\alpha)$ and $\alpha \in Q_{A'}(\beta)$, implying $\alpha \preceq \beta$. If $\alpha = 0$ then also $\alpha \preceq \beta$. This shows, together with (i)′, that Q is order-preserving, and hence by (ii) and Proposition 4(3), Q is deduction-adequate. □

Singles and the Lattice Case Let $\mathcal{A} = (A, \preceq, 0, 1)$ be a partially ordered set. For $\alpha, \beta \in A$, α is said to *cover* β if $\alpha \succ \beta$ and there does not exist $\gamma \in A$ with $\alpha \succ \gamma \succ \beta$. We say that $\alpha \in A$ is a *single* in $\mathcal{A}$ if there exists a unique $\beta \in A$ such that α covers β. If α is a single, we write α_* for the unique element that α covers. Since A is finite, it follows that, for $\gamma \in A$, if $\gamma \prec \alpha$ then $\gamma \preceq \alpha_*$.

Let S be the set of singles of $\mathcal{A}$. If $\mathcal{A}$ is a lattice, for $\alpha \in A$ define $\underline{\alpha} = \bigvee \{\beta \in S : \beta \preceq \alpha\}$, where $\bigvee \emptyset$ is defined to be 0. The following lemma shows that $\underline{\alpha}$ is always equal to α.

Lemma 3. *Let $\mathcal{A} = (A, \preceq, 0, 1)$ be a finite partially ordered set which is a lattice. Then, for all $\alpha \in A$, $\underline{\alpha} = \alpha$.*

Proof. Let S be the set of singles in $\mathcal{A}$. We prove this result using induction. $0 \notin S$ and $\underline{0} = \bigvee \emptyset = 0$, so the statement holds for $\alpha = 0$.

Suppose now the statement is true for all $\beta \prec \alpha$, i.e., $\underline{\beta} = \beta$ for all $\beta \prec \alpha$. We will show that $\underline{\alpha} = \alpha$. Since A is finite, this proves that $\underline{\alpha} = \alpha$ for all $\alpha \in A$.

Firstly, note that $\underline{\alpha} \preceq \alpha$. This is because α is an upper bound of all lower bounds of α. If $\alpha \in S$, then $\bigvee \{\beta \in S : \beta \preceq \alpha\} \succeq \alpha$, so $\underline{\alpha} = \alpha$. Therefore we can assume $\alpha \notin S$. Let $\alpha' = \bigvee \{\beta : \beta \prec \alpha\}$. Clearly $\alpha' \preceq \alpha$, since α is an upper bound of any lower bound of α. Also if $\beta \prec \alpha$ then $\beta \preceq \alpha'$. So if it were the case that $\alpha' \prec \alpha$ then α' would be the unique element that α covers, which is a contradiction since α is not a single. Therefore $\alpha' = \alpha$. Now, using the inductive hypothesis, we get $\alpha = \alpha' = \bigvee \{\underline{\beta} : \beta \prec \alpha\}$, which (because of commutativity, associativity and idempotence of $\vee$) is equal to $\bigvee \{\gamma \in S : \gamma \prec \alpha\}$, which equals $\underline{\alpha}$ since $\alpha \notin S$, completing the induction and proving the result. □

Proposition 6. *Let $\mathcal{A} = (A, \preceq, 0, 1)$ be a (finite) partially ordered set, and let S be the set of singles in $\mathcal{A}$. Then*

(1) If Q is any deduction-adequate function from A to 2^Θ (for some set Θ), then $|\Theta| \geq |S|$.
(2) If $B \subseteq A$ is such that Q_B is deduction-adequate then $B \supseteq S$. If $\mathcal{A}$ is a lattice then the converse also holds: Q_B is deduction-adequate if and only if $B \supseteq S$.

In particular, if $\mathcal{A}$ is a lattice then $Q_S : A \to 2^S$ is deduction-adequate, and Q_S is minimal in the following strong sense: there is no deduction-adequate mapping $Q : A \to 2^\Theta$ with Θ of smaller cardinality than S.

Proof. (1) Suppose $Q : A \to 2^\Theta$ is deduction-adequate. Let $\alpha \in S$. Since Q is deduction-adequate, it is order-preserving by Proposition 4(2), so $Q(\alpha) \supseteq Q(\alpha_*)$. Also $Q(\alpha) \neq Q(\alpha_*)$ since $Q(\alpha) \subseteq Q(\alpha_*)$ would imply $\alpha \preceq \alpha_*$, which is a contradiction. So $Q(\alpha) - Q(\alpha_*)$ is non-empty, and we choose some arbitrary element θ_α of it. We will show that if $\alpha \neq \beta$ then $\theta_\alpha \neq \theta_\beta$, which proves the result that $|\Theta| \geq |S|$.

Let α and β be two different elements of S. There are two possibilities: (a) if $\alpha \prec \beta$; then $\alpha \preceq \beta_*$, and so $\theta_\beta \notin Q(\beta_*) \supseteq Q(\alpha)$, since Q is order-preserving, so $\theta_\beta \notin Q(\alpha)$. This implies that $\theta_\alpha \neq \theta_\beta$. (b) Suppose that $\alpha \not\preceq \beta$. For $\gamma \in A$, if $\gamma \preceq \alpha, \beta$, then $\gamma \neq \alpha$ (since $\alpha \not\preceq \beta$) so $\gamma \prec \alpha$ and therefore $\gamma \preceq \alpha_*$. This shows that $\{\alpha, \beta\} \trianglelefteq \alpha_*$. Hence, by Proposition 4(1), $Q(\alpha) \cap Q(\beta) \subseteq Q(\alpha_*)$. Now, $\theta_\alpha \in Q(\alpha)$ and $\theta_\beta \in Q(\beta)$ so if θ_α were equal to θ_β then θ_α would be a member of $Q(\alpha) \cap Q(\beta)$ and hence of $Q(\alpha_*)$, which contradicts the definition of θ_α. So $\theta_\alpha \neq \theta_\beta$, as required.

(2) Suppose now that $B \subseteq A$ is such that Q_B is deduction-adequate. Then, in particular (by Proposition 4(2)(a)) it is order-preserving. This implies that if α is a single then $Q_B(\alpha_*) \not\supseteq Q_B(\alpha)$, so there exists $\beta \in B$ with $\beta \preceq \alpha$ and $\beta \not\preceq \alpha_*$.

But then β must be equal to α, showing that $\alpha \in B$. Since α was an arbitrary single, this shows that $B \supseteq S$.

Suppose that $\mathcal{A}$ is a lattice. Let B be a subset of A containing S. We will show first that Q_B is order-preserving. Suppose $Q_B(\alpha) \subseteq Q_B(\beta)$. So, for $\gamma \in B$, if $\gamma \preceq \alpha$ then $\gamma \preceq \beta$. In particular this holds for $\gamma \in S$. Therefore $\bigvee \{\gamma \in S : \gamma \preceq \alpha\} \preceq \bigvee \{\gamma \in S : \gamma \preceq \beta\}$, i.e., $\underline{\alpha} \preceq \underline{\beta}$. Hence, by Lemma 3, $\alpha \preceq \beta$. This combines with Proposition 5(i)$'$ to show that Q_B is order-preserving. Therefore, by Proposition 5(ii) and Proposition 4(3), Q_B is deduction-adequate. □

The above result shows us how to construct a deduction-adequate mapping $Q : A \to 2^{\Theta}$ which is minimal (with respect to $|\Theta|$) for the case of a lattice, by letting Θ be the set of singles.

It can also be shown, using Proposition 6 and Lemma 3, that if we consider only distributive lattices $\mathcal{A}$, then Q is an isomorphism between distributive lattices in which infimum and supremum in $Q(\mathcal{A})$ are intersection and union, respectively.

4.3 Construction of a deduction-adequate mapping $Q = Q_{A^\dagger}$ for the general case.

For $B \subseteq A$ define relations $\preceq_B$ by $\alpha \preceq_B \beta$ if and only if $\gamma \preceq \beta$ for all $\gamma \in B$ such that $\gamma \preceq \alpha$, i.e., if and only if $Q_B(\alpha) \subseteq Q_B(\beta)$. So Q_B is order-preserving if and only if $\preceq_B = \preceq$. Hence, by Proposition 5(iii), $\preceq_A = \preceq$. These relations contain $\preceq$, and are monotonic (decreasing) with respect to B: if $B' \subseteq B$ then $\preceq_{B'} \supseteq \preceq_B \supseteq \preceq_A = \preceq$, so that $\alpha \preceq \beta$ implies $\alpha \preceq_B \beta$, which implies $\alpha \preceq_{B'} \beta$.

We will construct a subset $A^\dagger$ of A with $\preceq_{A^\dagger} = \preceq$ so that $Q_{A^\dagger}$ is order-preserving and hence deduction-adequate (by Proposition 5, and Proposition 4(3)). Let $m = |A| - 1$. 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 $A^\dagger$ incrementally with $A^\dagger$ being the final set A_m.

Define $A_0 = \emptyset$, and for $i = 1, \ldots, m$, define Y_i, Z_i and A_i inductively as follows:

— set $Y_i = \{\alpha_i\}$ if there exists $k < i$ with $\alpha_i \preceq_{A_{i-1}} \alpha_k$; otherwise set $Y_i = \emptyset$;
— 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) $\alpha_j \preceq_{A_{i-1}} \alpha_i$;
— let $A_i = A_{i-1} \cup Y_i \cup Z_i$

Finally, we let $A^\dagger = A_m$, and define $Q = Q_{A^\dagger}$, i.e., for each $\alpha \in A$, $Q(\alpha) = \{\beta \in A^\dagger : \beta \preceq \alpha\}$.

We have to add sufficient elements to ensure that $\preceq = \preceq_{A^\dagger}$. Adding Y_i ensures that $\alpha_i \not\preceq_{A_i} \alpha_k$ and hence $\alpha_i \not\preceq_{A^\dagger} \alpha_k$ for $k < i$ (which we need since for $k < i$, $\alpha_i \not\preceq \alpha_k$); adding Z_i ensures that $\alpha_j \not\preceq_{A_i} \alpha_i$, and hence $\alpha_j \not\preceq_{A^\dagger} \alpha_i$, if $j < i$ and $\alpha_j \not\preceq \alpha_i$.

Example 2. We use the extension of Example 1 described in Section 2.1 with $A = \{0, e, g, p, \overline{e}, \overline{g}, \overline{p}, 1\}$, and the additional ordering $\overline{p} \prec \overline{g}$. We order A as $0, e, g, p, \overline{p}, \overline{g}, \overline{e}, 1$, so that $\alpha_0 = 0$, $\alpha_1 = e$, ..., $\alpha_4 = \overline{p}$, $\alpha_5 = \overline{g}$, etc. Then $\alpha_1, \alpha_2 \prec \alpha_4$, and $\alpha_1, \alpha_2, \alpha_3 \prec \alpha_5$, and $\alpha_2, \alpha_3 \prec \alpha_6$, and $\alpha_4 \prec \alpha_5$. Applying the algorithm gives $Y_1 = \{\alpha_1\}$, $Y_2 = \{\alpha_2\}$, $Y_3 = \{\alpha_3\}$, $Y_4 = Y_5 = Y_7 = \emptyset$ and $Y_6 = \{\alpha_6\}$, with for all i, $Z_i = \emptyset$. Thus $A^\dagger = \{\alpha_1, \alpha_2, \alpha_3, \alpha_6\}$. Elements α_1, α_2 and α_3 needed to be added because they are all singles. Element α_6 was added because if $B = \{\alpha_1, \alpha_2, \alpha_3\}$ then $\alpha_6 \preceq_B \alpha_5$ (since α_5 is an upper bound of α_1 α_2 and α_3) but $\alpha_6 \not\preceq \alpha_5$. It can be confirmed that for all i, j, we have $\alpha_i \preceq \alpha_j$ if and only if $Q_{A^\dagger}(\alpha_i) \subseteq Q_{A^\dagger}(\alpha_j)$. For example, $\alpha_4 \prec \alpha_5$, and $Q_{A^\dagger}(\alpha_4) = \{\alpha_1, \alpha_2\}$, which is a subset of $Q_{A^\dagger}(\alpha_5) = \{\alpha_1, \alpha_2, \alpha_2\}$.

Theorem 3. *With the above definition of $A^\dagger$ and $Q = Q_{A^\dagger}$, the relations $\preceq$ and $\preceq_{A^\dagger}$ are the same, and Q is order-preserving, i.e., for all $\alpha, \beta \in A$, $\alpha \preceq \beta$ if and only if $Q(\alpha) \subseteq Q(\beta)$. Furthermore, Q is deduction-adequate, so for any $\mathcal{A}$-constraint d and set of $\mathcal{A}$-constraints C, we have $C \models d$ if and only if $C^Q \models d^Q$.*

Proof. The monotonicity property implies that if $\alpha \preceq \beta$ then $\alpha \preceq_{A^\dagger} \beta$. We need to show that for all j, k, if $\alpha_j \not\preceq \alpha_k$ then $\alpha_j \not\preceq_{A^\dagger} \alpha_k$. So suppose $\alpha_j \not\preceq \alpha_k$. By reflexivity of $\preceq$, $j \neq k$, so there are two cases: (a) $j < k$ and (b) $k < j$.

(a) If $j < k$ and $\alpha_j \not\preceq_{A_{k-1}} \alpha_k$, then by monotonicity, $\alpha_j \not\preceq_{A^\dagger} \alpha_k$. On the other hand, if $j < k$ and $\alpha_j \preceq_{A_{k-1}} \alpha_k$ then, by definition, Z_k contains α_j, so $A^\dagger \ni \alpha_j$. This implies $\alpha_j \not\preceq_{A^\dagger} \alpha_k$ since $\alpha_j \in A^\dagger$, $\alpha_j \preceq \alpha_j$ but $\alpha_j \not\preceq \alpha_k$.

(b) If $k < j$ and $\alpha_j \not\preceq_{A_{j-1}} \alpha_k$, then by monotonicity, $\alpha_j \not\preceq_{A^\dagger} \alpha_k$. On the other hand, if $k < j$ and $\alpha_j \preceq_{A_{j-1}} \alpha_k$ then $Y_j = \{\alpha_j\}$, so $A^\dagger \ni \alpha_j$, and again $\alpha_j \not\preceq_{A^\dagger} \alpha_k$ since $\alpha_j \in A^\dagger$, $\alpha_j \preceq \alpha_j$ but $\alpha_j \not\preceq \alpha_k$.

We have just shown that $\alpha \preceq \beta$ if and only if $\alpha \preceq_{A^\dagger} \beta$, and this is if and only if $Q(\alpha) \subseteq Q(\beta)$. This implies, along with Proposition 5 (ii), that Q satisfies conditions (i) and (ii) of Proposition 4(3), and so that result implies that Q is deduction-adequate. □

This theorem gives us a sound and complete method for determining if inferences of the form $C \models d$ hold (for general $\mathcal{A}$-constraints): we construct $A^\dagger$ and $Q_{A^\dagger}$ as defined above, convert the $\mathcal{A}$-constraints into $2^{A^\dagger}$-constraints, and use the computational techniques available for the distributive lattice case (see Section 3.4) to determine if $C^Q \models d^Q$.

Minimality for the lattice case The complexity of basic operations ($\cap$ and $\cup$) needed for computation in 2^Θ is linear in $|\Theta|$. One might wonder whether it's possible to embed $\mathcal{A}$ into 2^Θ for some Θ much smaller than $A^\dagger$. The following result shows that, at least if $\mathcal{A}$ is a lattice, $A^\dagger$ is of minimum possible size, by showing that $A^\dagger$ is the set of singles, and using Proposition 6(1).

Proposition 7. *Let $\mathcal{A} = (A, \preceq, 0, 1)$ be a (finite) partially ordered set which is a lattice, and define $A^\dagger$ as in the construction above. Then $A^\dagger$ is the set of singles*

in $\mathcal{A}$. Also, if Q' is any deduction-adequate function from A to 2^{Θ} (for some set Θ), then $|\Theta| \geq |A^\dagger|$.

Proof. Let S be the set of singles in $\mathcal{A}$. We will show that $S = \bigcup_i Y_i$ and each Z_i is empty. Suppose α_i is a single, and let α_k be the unique element that α_i covers. $k < i$ since $\alpha_k \prec \alpha_i$. If $\alpha_j \prec \alpha_i$ then $\alpha_j \preceq \alpha_k$. Since $A_{i-1} \not\ni \alpha_i$, $\alpha_i \preceq_{A_{i-1}} \alpha_k$. This implies that $Y_i = \{\alpha_i\}$ and $\alpha_i \in A_i$ and hence $\alpha_i \in A^\dagger$. Therefore $A^\dagger \supseteq S$.

Suppose $j < i$ and $\alpha_j \preceq_{A_{i-1}} \alpha_i$; we will show that $\alpha_j \preceq \alpha_i$, proving that, for each i, $Z_i = \emptyset$. Consider any $\alpha_l \in S$ such that $\alpha_l \preceq \alpha_j$. Then we must have $l \leq j < i$, and hence $\alpha_l \in A_{i-1}$, since $\alpha_l \in Y_l$ as shown above. Since $\alpha_j \preceq_{A_{i-1}} \alpha_i$ and $\alpha_l \preceq \alpha_j$ we have $\alpha_l \preceq \alpha_i$. This argument implies that $\bigvee \{\alpha_l \in S : \alpha_l \preceq \alpha_j\} \preceq \alpha_i$, so $\underline{\alpha_j} \preceq \alpha_i$. By Lemma 3, $\alpha_j = \underline{\alpha_j}$, implying $\alpha_j \preceq \alpha_i$, as required.

We'll next show that $Y_i = \overline{\emptyset}$ if $\alpha_i \notin S$. We proceed using Proof by Contradiction: suppose $Y_i \neq \emptyset$, so there exists $k < i$ with $\alpha_i \preceq_{A_{i-1}} \alpha_k$. Consider any $\alpha_l \in S$ such that $\alpha_l \preceq \alpha_i$. So $l \leq i$, and in fact $l < i$ since $\alpha_i \notin S$. Hence $\alpha_l \in A_{i-1}$ since $l \leq i - 1$ and $Y_l = \{\alpha_l\}$, as shown above. Then $\alpha_i \preceq_{A_{i-1}} \alpha_k$ implies $\alpha_l \preceq \alpha_k$. Therefore we have $\bigvee \{\alpha_l \in S : \alpha_l \preceq \alpha_i\} \preceq \alpha_k$, so $\underline{\alpha_i} \preceq \alpha_k$. By Lemma 3, $\alpha_i = \underline{\alpha_i}$, giving $\alpha_i \preceq \alpha_k$ which contradicts $k < i$.

Therefore, $\alpha \in A^\dagger$ if and only if for some i, $\alpha \in Y_i \cup Z_i$, which is if and only if $\alpha \in S$, showing that $A^\dagger = S$.

The last part follows from Proposition 6(1). □

The purpose of the construction is to produce an embedding into not too large a set. One extreme is when $\preceq$ is a total order, the set $A^\dagger$ is just $A - \{0\}$, so $|A^\dagger| = |A| - 1$. Propositions 6 and 7 show that when $\mathcal{A}$ has a strong structure, $A^\dagger$ can be very compact in comparison to A. For example, if A is the lattice of all subsets of a set Ω; then $A^\dagger$ is the set of singleton subsets of Ω so $|A^\dagger| = |\Omega| = \log_2 |A|$.

4.4 Answering Queries and Finding Simple Derived Constraints

Deduced $\mathcal{A}$-constraints d with small V_d are often of particular interest as they represent more general statements about the preferences. For example, in the problem of arranging a meeting discussed earlier, if we can derive a constraint, on just the date variable(s), saying that the meeting will either have to be next week or not before next year, it may prompt immediate action, or e.g., it may help in persuading Gerard to rethink his preferences.

Let $\mathcal{B}$ be a distributive lattice, and let C be a set of $\mathcal{B}$-constraints. For $U \subseteq V$, define $\mathcal{B}$-constraint C_U to be $(\bigwedge C \wedge 1_U)^{\downarrow U}$. A hypergraph G on V is a set of sets of variables. A hypertree is a hypergraph that has a particular acyclicity property. Hypertree H is said to be a hypertree cover of hypergraph G if for all $U \in G$, there exists $U' \in H$ with $U' \supseteq U$. We can use a standard method to generate a hypertree cover H of $G = \{V_c : c \in C\}$ [21]. As shown in [21, 18], the variable elimination approach (see Section 3.4) can be extended to generate C_U for each $U \in H$, using a message passing algorithm on a join tree; this can then

be used to generate (if we wish) C_U for any $U \in G$, and C_U for any smaller sets U. (This might be thought of as enforcing global consistency.) Once we have performed this computation, we can quickly answer any query of the form $C \models d$, where d is any $\mathcal{B}$-constraint such that V_d is a subset of some set U in the hypertree. We project C_U to set of variables V_d to give $c' = (C_U)^{\downarrow V_d}$. Then $C \models d$ if and only if for all $y \in D(V_d)$, $c'(y) \preceq d(y)$, which can be checked easily if $D(V_d)$ is small.

Generating derived $\mathcal{B}$-constraints C_U involves a sequence of combinations and projections. For the complexity, the critical part is (usually) using combination to generate $\mathcal{B}$-constraints c with $V_c = U$ for some $U \in H$. This involves (at least with the most obvious implementation) proportional to $|D(U)|$ lattice operations $\wedge$.

Suppose we are using the embedding $Q_{A'}$ to answer queries given a set of $\mathcal{A}$-constraints, using this approach. Then performing the lattice operations $\wedge$ takes time proportional to $|A'|$. So the computation time is typically proportional to $|A'| \max_{U \in H} |D(U)|$. This indicates that generating a set A', such as $A^\dagger$, with $|A'|$ much smaller than $|A|$ will often make a considerable difference. Producing $A^\dagger$ and the sets $Q_{A^\dagger}(\alpha)$ by crudely implementing the above definitions takes time proportional to $|A|^2|A^\dagger|$, which is an issue if $|A|$ is large. But it can be done off-line: it only ever needs to be done once for a given partially ordered set $\mathcal{A}$. (If A is so large that the time to produce $A^\dagger$ would dominate the computation time, then an alternative is to use $A' = A$, in which case producing $Q_{A'}$ is $O(|A|^2)$.)

An alternative approach, based on decision diagrams (which implement a certain form of dynamic programming) can also be used for answering queries, by computing C_U for certain small sets U [25].

5 A Different Semantics

The semantics described earlier assumed that we knew the identity of the partially ordered set of degrees. In this section we consider a situation where the set of preference degrees is unknown except that it contains a known set A.

Example 3. Consider partially ordered set $\mathcal{A} = (A, 0, 1, \preceq)$ with $A = \{0, 1, \alpha, \beta\}$, and $0 \prec \alpha, \beta \prec 1$, but no order between α and β. Define $\mathcal{A}$-constraints c_1 and c_2 on a single variable (with two possible assignments a and b) by: $c_1(a) = c_2(a) = 1$, $c_1(b) = \alpha$, $c_2(b) = \beta$. With the semantics given in Section 2, $\{c_1, c_2\}$ entails the $\mathcal{A}$-constraint d given by $d(a) = 1$, $d(b) = 0$, which states that value b has the lowest preference degree. 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 completely reasonable (and correct) given that the $\mathcal{A}$-constraints were considered as restricting possible A-valued assignments. However, sometimes the intention will not be precisely that: our input information may consist of $\mathcal{A}$-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 $\mathcal{A}$-constraints. However, these partially ordered elements may have come from an unknown larger partially ordered set $(A^*, 0, 1, \preceq^*)$ which extends[2] $(A, 0, 1, \preceq)$, i.e., $A^* \supseteq A$, and for $\alpha, \beta \in A$, $\alpha \preceq \beta$ implies $\alpha \preceq^* \beta$.

This suggests the following semantics for $\mathcal{A}$-constraints. In this semantics, a model is a pair $(M, \mathcal{A}^*)$ where $\mathcal{A}^* = (A^*, 0, 1, \preceq^*)$ is a partially ordered set extending $(A, 0, 1, \preceq)$, and M is a function from $D(V)$ to A^*. For $\mathcal{A}$-constraint c, we then define $(M, \mathcal{A}^*) \models c$ (that is, $(M, \mathcal{A}^*)$ *is a model of* c, or $(M, \mathcal{A}^*)$ *satisfies* c) if for all $x \in D(V)$, $M(x) \preceq^* c(x)$. For set of $\mathcal{A}$-constraints C and $\mathcal{A}$-constraint d we then say $C \models^* d$ if $(M, \mathcal{A}^*) \models d$ for any $(M, \mathcal{A}^*)$ satisfying every $\mathcal{A}$-constraint in C.

We can proceed in a similar fashion as for $\models$ defined in Section 2. Suppose $B \cup \{\gamma\} \subseteq A$ and $\mathcal{A}^* = (A^*, 0, 1, \preceq^*)$ extends $(A, 0, 1, \preceq)$. A lower bound in $\mathcal{A}^*$ of γ is an element $\alpha' \in A^*$ with $\alpha' \preceq^* \gamma$. Similarly a lower bound α' in $\mathcal{A}^*$ of B is a lower bound of every element of B; we then write $\alpha' \preceq^* B$. We define $B \trianglelefteq^* \gamma$ if for every $(A^*, 0, 1, \preceq^*)$ extending $(A, 0, 1, \preceq)$, every lower bound in $\mathcal{A}^*$ of B is a lower bound of γ. Model $(M, \mathcal{A}^*)$ satisfies (every element of) C if and only if for all $x \in D(V)$, $M(x)$ is a lower bound for $C(x) = \{c(x) : c \in C\}$. The reason for introducing relation $\trianglelefteq^*$ is the following characterisation of consequence.

Proposition 8. *Let $C \cup \{d\}$ be a set of $\mathcal{A}$-constraints with C non-empty. Then $C \models^* d$ if and only if for all $x \in D(V)$, $C(x) \trianglelefteq^* d(x)$.*

Proof. Suppose, for some $x_0 \in D(V)$, it is not the case that $C(x_0) \trianglelefteq^* d(x_0)$, so that there exists $\mathcal{A}^* = (A^*, 0, 1, \preceq^*)$ extending $(A, 0, 1, \preceq)$ and $\alpha \in A^*$ such that α is a lower bound of $C(x_0)$ (with respect to $\preceq^*$) but it is not the case that $\alpha \preceq^* d(x_0)$. Define model $(M, \mathcal{A}^*)$ by $M(x_0) = \alpha$ and, for $x \neq x_0$, define $M(x) = 0$. Clearly, for each $x \in D(V)$, $M(x)$ is a lower bound for $C(x)$, so $(M, \mathcal{A}^*)$ satisfies C. But $(M, \mathcal{A}^*)$ does not satisfy d, because $M(x_0) \not\preceq^* d(x_0)$. This shows that $C \not\models^* d$.

Conversely, suppose that for all $x \in D(V)$, $C(x) \trianglelefteq^* d(x)$. Suppose $\mathcal{A}^*$ extends $\mathcal{A}$ and let $(M, \mathcal{A}^*)$ be a model of C. Then for all $x \in D(V)$, $M(x)$ is a lower bound for $C(x)$ so, by the hypothesis, $M(x)$ is a lower bound for $d(x)$. This shows that $C \models^* d$. □

The relation $\trianglelefteq^*$ can be expressed more simply as follows:

Lemma 4. *Suppose $B \cup \{\gamma\} \subseteq A$ with B non-empty. Then $B \trianglelefteq^* \gamma$ if and only if there exists $\beta \in B$ with $\beta \preceq \gamma$.*

Proof. One of the implications is easy: suppose there exists $\beta \in B$ with $\beta \preceq \gamma$. Then for any $(A^*, 0, 1, \preceq^*)$ extending $(A, 0, 1, \preceq)$ and any α' which is a lower

[2] We could change this definition as follows: $(A^*, 0, 1, \preceq^*)$ extends $(A, 0, 1, \preceq)$ if and only if $A^* \supseteq A$, and for $\alpha, \beta \in A$, $\alpha \preceq \beta \iff \alpha \preceq^* \beta$. All the results of this section would then still hold (using the same proofs).

bound in $\mathcal{A}^*$ of B, we have $\alpha' \preceq^* \beta$. Since $\preceq^*$ extends $\preceq$ we have $\beta \preceq^* \gamma$, so by transitivity of $\preceq^*$ we have $\alpha' \preceq^* \gamma$, proving that $B \trianglelefteq^* \gamma$.

For the converse, for any $B \subseteq A$ not containing 0 we generate the extension $\mathcal{A}^* = (A^*, 0, 1, \preceq^*)$ of $(A, 0, 1, \preceq)$ defined as follows. We add extra symbol B^* to A to give $A^* = A \cup \{B^*\}$. Define $\preceq^*$ by: $\alpha_1 \preceq^* \alpha_2$ if and only if either (a) $\alpha_1, \alpha_2 \in A$ and $\alpha_1 \preceq \alpha_2$ or (b) $\alpha_1 \in \{0, B^*\}$ and $\alpha_2 = B^*$; or (c) $\alpha_1 = B^*$ and there exists $\beta \in B$ such that $\beta \preceq \alpha_2$. Since B is non-empty, $B^* \preceq^* 1$ so for all $\delta \in A^*$, $0 \preceq^* \delta \preceq^* 1$. Clearly, $\mathcal{A}^*$ extends $(A, 0, 1, \preceq)$. We need to show that $\preceq^*$ is a partial order: (i) $\preceq^*$ is reflexive because $\preceq$ is reflexive. (ii) Transitivity: suppose $\delta_1 \preceq^* \delta_2 \preceq^* \delta_3$, but it is not the case that $\delta_1 \preceq^* \delta_3$. Then all three must be different (since $\preceq^*$ is reflexive). Also, none can be 0. Because $\preceq$ is transitive, all three cannot be in A, so one of them must be B^*. But δ_1 is the only one that could possibly be B^* (as e.g., $\delta_2 = B^*$ would imply $\delta_1 = B^*$, contradicting $\delta_1 \neq \delta_2$). So $\delta_1 = B^*$ and there exists $\beta \in B$ with $\beta \preceq \delta_2$. But $\delta_2 \preceq \delta_3$, so $\beta \preceq \delta_3$, by transitivity of $\preceq$. This implies $B^* \preceq^* \delta_3$, i.e., $\delta_1 \preceq^* \delta_3$, which is a contradiction. (iii) Antisymmetry: assume $\delta_1 \preceq^* \delta_2$ and $\delta_2 \preceq^* \delta_1$. Suppose it were the case that $\delta_1 \neq \delta_2$. Then by antisymmetry of $\preceq$, one of the two must be B^*, which would imply the other was also B^* (since $B \not\ni 0$ and so $B^* \not\preceq^* 0$), contradicting them being different. This proves antisymmetry. So $(A^*, 0, 1, \preceq^*)$ is a partially ordered set extending $(A, 0, 1, \preceq)$.

For $\gamma \in A$, if there does not exist $\beta \in B$ with $\beta \preceq \gamma$ then $B \not\ni 0$ and it is not the case that $B^* \preceq^* \gamma$, even though B^* is a lower bound in A^* for B. This shows that it is not the case that $B \trianglelefteq^* \gamma$, as required. □

Bringing the proposition and lemma together we have the following theorem, which characterises this new semantic entailment relation $\models^*$. This result indicates that the consequence relation $\models^*$ will often be much weaker than the consequence $\models$ defined in Section 2.2, since $\trianglelefteq^*$ will often be a much weaker relation than $\trianglelefteq$ (the antecedents B for $\trianglelefteq^*$ don't interact).

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

Another consequence of Lemma 4 and Proposition 8 is that if all the $\mathcal{A}$-constraints in $C \cup \{d\}$ only take values in $\{0, 1\}$ then $C \models d$ if and only if $C \models^* d$. This is because the lemma then implies for such C and d that $C(x) \trianglelefteq^* d(x)$ if and only if $C(x) \trianglelefteq d(x)$, which is if and only if either $d(x) = 1$ or $C(x) \ni 0$. Proposition 8 and Proposition 1 then imply the result. Consequently, $\models$ and $\models^*$ both extend deduction for (standard) constraints.

We can also use these results to show that for any given $\mathcal{A}$, the deduction problem with respect to $\models^*$ for $\mathcal{A}$-constraints is coNP-complete, similarly to the corresponding result for $\models$, Proposition 2. Theorem 4 can be used to show that the complement of the problem is in NP, and the same reduction from 3SAT can be used to show the complement is NP-hard, since the constraints in the constructed $C \cup \{d\}$ only take values in $\{0, 1\}$.

For each $z \in D(V_d)$ and $c \in C$ define the (standard) constraint $c_{\not\preceq d(z)}$ consisting of all tuples $y \in D(V_c)$ such that $c(y) \not\preceq d(z)$, i.e., $\neg(c(y) \preceq d(z))$. Complete tuple x satisfies $c_{\not\preceq d(z)}$ if and only if $c(x) \not\preceq d(z)$. We also use $V_d = z$ to mean the constraint with scope V_d which is satisfied if and only if $V_d = z$.

Theorem 4 can be written in a different way, leading to a computational procedure to determine the entailment $C \models^* d$: for each assignment z to V_d we check whether a set of constraints is satisfied. This can be done using any of the usual CSP methods.

Proposition 9. *$C \models^* d$ if and only if for all $z \in D(V_d)$, the set of (standard) constraints $\{c_{\not\preceq d(z)} : c \in C\} \cup \{V_d = z\}$ is unsatisfiable.*

Proof. Let z be any assignment to V_d. The set of constraints $\{c_{\not\preceq d(z)} : c \in C\} \cup \{V_d = z\}$ is unsatisfiable if and only if there does not exist complete tuple x satisfying all the constraints, i.e., if for all $x \in D(V)$ with $x^{\downarrow V_d} = z$, there exists $c \in C$ with $c(x) \preceq d(z)$. Therefore [for all $z \in D(V_d)$, the set of constraints $\{c_{\not\preceq d(z)} : c \in C\} \cup \{V_d = z\}$ is unsatisfiable] if and only if for all $x \in D(V)$, there exists $c \in C$ with $c(x) \preceq d(x)$. Theorem 4 then proves the result. □

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 consequence being the same as that defined in Section 2, since the properties of Lemma 1 (in particular, part (ii)) still hold, and hence so does the counterpart of Theorem 1.

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

6 Finding Good and Optimal Complete tuples

Let $\mathcal{A} = (A, \preceq, 0, 1)$ be a partially ordered set, and let C be a set of $\mathcal{A}$-constraints. We will consider the problem of finding complete assignments $x \in D(V)$, which may be considered as being among the better or best assignments.

For $\alpha \in A$, we say that x *is* α*-satisfactory* if x satisfies each $\mathcal{A}$-constraint at least to degree α. This happens if and only if x is a solution of the set of (standard) constraints $C_{\succeq\alpha} = \{c_{\succeq\alpha} : c \in C\}$, where $c_{\succeq\alpha} = \{y \in D(V_c) : c(y) \succeq \alpha\}$. So standard CSP solving techniques can be used to find tuples which satisfy each constraint to a (relatively) high degree.

This can also be used to generate an ordering $\succeq_1$ on $D(V)$ by $x \succeq_1 x'$ if and only if x is α-satisfactory for every α such that x' is α-satisfactory. $\succeq_1$ is then a pre-order on $D(V)$ (a reflexive and transitive relation). x is α-satisfactory if and only if α is a member of $\mathrm{LB}(C(x))$, the set of lower bounds of $C(x)$. This implies that $x \succeq_1 x'$ if and only if $\mathrm{LB}(C(x)) \supseteq \mathrm{LB}(C(x'))$. When $\mathcal{A}$ is a lattice then $x \succeq_1 x'$ if and only if $(\bigwedge C)(x) \succeq (\bigwedge C)(x')$; if $\mathcal{A}$ is totally ordered, $\succeq_1$ is just the ordering $\succeq_{min}$ given by $x \succeq_{min} x'$ if and only if $\min\{c(x) : c \in C\} \succeq \min\{c(x') : c \in C\}$. We can define optimal assignments x to be maximal elements with respect to $\succeq_1$, i.e. x such that there exists no $x' \in D(V)$ with $x' \succeq_1 x$ and $x \not\succeq_1 x'$.

A simple alternative definition of optimality is that x satisfies $C_{\succeq\alpha}$ for some maximal α such that $C_{\succeq\alpha}$ is satisfiable. Such maximal α can be found using an iterative algorithm, which, given an appropriate representation of $\mathcal{A}$, can involve checking the satisfiability of *of order* $\log|A|$ CSPs. For the case when $\mathcal{A}$ is a lattice, both these definitions of x being optimal reduce to $(\bigwedge C)(x)$ being maximal.

Ordering relation $\succeq_1$ might be criticised for the same reasons as $\succeq_{min}$ is criticised in e.g., [15]: it suffers from the 'drowning problem'. However, a natural condition that one may wish to impose on the ordering on $D(V)$, and which has perhaps surprisingly strong consequences, is the following: (I) If we replace C by semantically equivalent C' then it doesn't change the ordering. This implies that adding implied constraints doesn't change the ordering. We can define "semantically equivalent" according to the semantics of either Section 2.2 or of Section 5. If we use the first semantics, (I) means that the ordering just depends on the set of models $\{M : M \models C\}$. We have: $M \models C$ if and only for all $x \in D(V)$, $M(x) \in \mathrm{LB}(C(x))$. Another (less precise) natural condition is the following: (II) The comparison between x and x' depends only on the parts of the problem related to x and x', and not on the values of the $\mathcal{A}$-valuations on any other complete assignments. This condition then suggests that the comparison between x and x' should depend only on the relationship between the two sets $\mathrm{LB}(C(x))$ and $\mathrm{LB}(C(x'))$. In the case of a lattice, the set $\mathrm{LB}(C(x))$ is characterised by a single semiring value, $(\bigwedge C)(x) = \bigwedge_{c \in C} c(x)$, with $\mathrm{LB}(C(x))$ being all lower bounds of this value, so the ordering between x and x' is determined by comparing values $(\bigwedge C)(x)$ and $(\bigwedge C)(x')$. For the general case, the obvious way of comparing $\mathrm{LB}(C(x))$ and $\mathrm{LB}(C(x'))$ is by determining if one is a subset of the other, leading to $\succeq_1$ as defined above.

If instead we were to interpret condition (I) on the basis of the semantics of Section 5 then we can use a similar argument to suggest that the ordering between x and x' just depends on the set of minimal elements in $C(x)$, and the

set of minimal elements in $C(x')$. This also leads in the totally ordered case to relation $\succeq_{min}$.

One may also consider syntactic definitions of the ordering on $D(V)$, allowing semantically equivalent sets of $\mathcal{A}$-valuations to lead to different orderings. In particular, one might generalise approaches for the totally ordered case such as in [15]. One approach of this kind is to associate a *multiset* $C^*(x) = \{c(x) : c \in C\}$ to each $x \in D(V)$, and use a method for comparing sub-multisets of A; for example, we can say that x is at least as good as x' if for all $\alpha \in C^*(x) - C^*(x')$ there exists element $\beta \in C^*(x') - C^*(x)$ with $\beta \preceq \alpha$.

7 Summary And Discussion

The paper defines a logic of soft constraints, based on a very simple semantics. This formalism allows one to reason with soft constraints that assign arbitrary partially ordered degrees to tuples. A possibilistic logic based on a partially ordered set is also constructed, which is formally a very closely related system. In the case where the partially ordered set is a distributive lattice, the logic has the same consequence relation as that generated by idempotent semiring-based CSPs, so the system can be considered as an extension of idempotent semiring-based CSPs. However, this logic is far more expressive since distributive lattices are a very special kind of partial order. The properties of a distributive lattice allow a range of computational techniques for computing consequences. One of the main contributions of this paper is to show how to make use of these computational techniques for the general case, by compactly embedding the partially ordered set within a distributive lattice.

Another logic of soft constraints is also defined based on a weaker semantics, and it is shown how to compute consequences in this logic also. The computational properties of the logics seem reasonable. For example, testing if a soft constraint can be derived from a set of soft constraints, will be feasible even for very large problems, if the topological structure is appropriately sparse, and the partially ordered set is fairly small.

A by-product of this work is a new and, in a certain sense, deeper semantics for idempotent semiring-based CSPs. The previous semantics treats the operations of multiplication $\wedge$ and addition $\vee$ as primitives, and defines the semantics in terms of them. The strength of this new semantics is that it assumes so little: essentially the whole system follows from equation (1), saying that the constraints express upper bounds on preference degrees for complete assignments.

Both logics, but especially the second, are somewhat conservative. It would be interesting to look at more adventurous (non-monotonic) extensions where tentative inferences about preferences are suggested. Another valuable extension could be a formalism that represented both absolute preferences (as these do) and relative preferences of the kind that CP-nets express.

Acknowledgements.

I'm grateful for the thought-provoking comments of the referees, and for feedback from Gene Freuder, Barry O'Sullivan, Steve Prestwich and Rick Wallace. Part of this work was done 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.

References

1. S. Benferhat, S. Lagrue, and O. Papini. Reasoning with partially ordered information in a possibilistic logic framework. In *Proceedings of IPMU 2002*, pages 1047–1052, 2002.
2. U. Bertele and F. Brioschi. *Nonserial Dynamic Programming*. Academic Press, 1972.
3. S. Bistarelli. *Semirings for Soft Constraint Solving and Programming*. Springer, 2004.
4. S. Bistarelli, T. Fruewirth, M. Marte, and F. Rossi. Soft constraint propagation and solving in constraint handling rules. *Computational Intelligence, Special Issue on Preferences in AI and CP*, 2004.
5. S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based Constraint Solving and Optimization. *Journal of the ACM (JACM)*, 44(2):201–236, 1997.
6. S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint logic programming: Syntax and semantics. *ACM Transactions on Programming Languages and Systems*, 2001.
7. S. Bistarelli, U. Montanari, and F. Rossi. Soft concurrent constraint programming. In *Proc. 11th European Symposium on Programming (ESOP)*, Lecture Notes in Computer Science (LNCS), pages 53–67. Springer, 2002.
8. S. Bistarelli, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, and H. Fargier. Semiring-based CSPs and Valued CSPs: Frameworks, properties and comparison. *Constraints*, 4(3):199–240, 1999.
9. C. Boutilier, R. Brafman, C. Domshlak, H. Hoos, and D. Poole. CP-nets: a tool for representing and reasoning with conditional ceteris paribus statements. *JAIR*, 21:135–191, 2004.
10. C. Boutilier, R. I. Brafman, H. H. Hoos, and D. Poole. Reasoning with conditional ceteris paribus preference statements. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI99)*, pages 71–80, 1999.
11. R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113, 1999.
12. R. Dechter and I. Rish. Mini-buckets: A general scheme for bounded inference. *Journal of the ACM*, 50(2):107–153, 2003.
13. D. Dubois, J. Lang, and H. Prade. Timed possibilistic logic. *Fundamenta Informaticae*, XV, 1991.
14. D. Dubois, J. Lang, and H. Prade. *Possibilistic Logic*, pages 439–513. In: Handbook of Logic in Artificial Intelligence and Logic Programming Vol. 3, D. Gabbay and C. Hogger and J. Robinson (eds.). Oxford University Press, 1994.
15. H. Fargier, J. Lang, and T. Schiex. Selecting preferred solutions in fuzzy constraint satisfaction problems. In *Proc. of the first European Congress on Fuzzy and Intelligent Technologies*, 1993.

16. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.
17. J. Kohlas. *Information Algebras*. Springer, London, 2003.
18. J. Kohlas and P. Shenoy. *Computation in Valuation Algebras*. In: Kohlas, J., Moral, S., (eds.) Algorithms for Uncertainty and Defeasible Reasoning, Volume 5, Handbook of Defeasible Reasoning. Kluwer Academic Publishers, 2000.
19. T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: hard and easy problems. In *Proc. IJCAI-95*, pages 631–637, 1995.
20. P. P. Shenoy. Valuation-based systems: A framework for managing uncertainty in expert systems. *Fuzzy Logic for the Management of Uncertainty*, 1992.
21. P. P. Shenoy and G. Shafer. Axioms for probability and belief function propagation. In *Uncertainty in Artificial Intelligence 4*, pages 575–610, 1990.
22. N. Wilson. A logic of partially satisfied constraints. In *Soft'03: Fifth International Workshop on Soft Constraints*, 2003.
23. N. Wilson. Extending CP-nets with stronger conditional preference statements. In *Proceedings of AAAI-04*, pages 735–741, 2004.
24. N. Wilson. Soft constraints with partially ordered preferences (short paper). In *Proc. European Conference on Artificial Intelligence*, pages 1111–1112, 2004.
25. N. Wilson. Decision diagrams for the computation of semiring valuations. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 331– 336, 2005.