The Logic behind Weighted CSP*

Carlos Ansótegui
DIEI, UdL
Jaume II 69, Lleida, Spain

María Luisa Bonet
LSI, UPC
Barcelona, Spain

Jordi Levy and **Felip Manyà**
IIIA, CSIC
Campus UAB, Bellaterra, Spain

Abstract

We define a translation from Weighted CSP to signed Max-SAT, and a complete resolution-style calculus for solving signed Max-SAT. Based on these results, we then describe an original exact algorithm for solving Weighted CSP. Finally, we define several derived rules and prove that they enforce the main soft arc consistency defined in the literature when applied to Weighted CSP instances.

1 Introduction

The Weighted Constraint Satisfaction Problem (WCSP) is a well known soft constraint framework for modelling over-constrained problems with practical applications in domains such as resource allocation, combinatorial auctions and bioinformatics. WCSP is an optimization version of the CSP framework in which constraints are extended by associating *costs* to tuples. Solving a WCSP instance, which is NP-hard, consists in finding a complete assignment of minimal cost.

Global consistency WCSP algorithms such as Bucket Elimination [Dechter, 1999] solve WCSP instances without search. They obtain an optimal solution by applying, to the original instance, transformations that preserve cost distributions. On the other hand, WCSP Branch and Bound (BnB) based algorithms such as PFC [Freuder and Wallace, 1992], PFC-MRDAC [Larrosa and Meseguer, 1996], Russian Doll Search [Verfaillie *et al.*, 1996], MAC* [Larrosa and Schiex, 2004], MFDAC* [Larrosa and Schiex, 2003] and MEDAC* [de Givry *et al.*, 2005] perform a systematic search in the space of all possible assignments. They differ in the method of computing a lower bound at each node of the proof tree to prune some parts of the search space. Modern algorithms such as MAC*, MFDAC* and MEDAC* enforce some extension of Arc Consistency (AC) to WCSP —Soft AC (AC*), Full Directional AC (FDAC*) or Existential Directional AC (EDAC*)— when computing that lower bound.

In this paper we relate ideas from three different research communities —Multiple-Valued Logic, Satisfiability and Constraint Processing— with the aim of describing the underlying logic of WCSP. First, we define an encoding, called *signed encoding*, that transforms any WCSP instance to a *Signed Max-SAT* instance, where Signed Max-SAT is the Max-SAT problem of the multiple-valued clausal forms known as Signed CNF formulas (see [Ansótegui and Manyà, 2003] and references therein). Second, we define a complete resolution calculus for solving Signed Max-SAT. Third, we devise an exact algorithm for solving WCSP from the completeness proof of the resolution calculus. Fourth, we define several sound inference rules for Signed Max-SAT that enforce some known arc consistency properties when applied to the signed encoding of any binary WCSP instance.

The structure of the paper is as follows. Section 2 contains preliminary definitions and the signed encoding. Section 3 defines the inference rule for signed Max-SAT and proves its soundness and completeness. Section 4 describes an exact algorithm for solving Weighted CSP. Section 5 defines four derived rules that enforce soft local consistency properties. Finally, Section 6 presents the conclusions of our work.

2 Preliminaries

Definition 1 *A* truth value set*, or* domain*, N is a non-empty finite set $\{i_1, i_2, \ldots, i_n\}$ where n denotes its cardinality. A* sign *is a subset $S \subseteq N$ of truth values. A* signed literal *is an expression of the form $S{:}p$, where S is a sign and p is a propositional variable. The* complement *of a signed literal l of the form $S{:}p$, denoted by $\overline{l}$, is $\overline{S}{:}p = (N \setminus S){:}p$. A* signed clause *is a disjunction of signed literals. A* signed CNF formula *is a* multiset *of signed clauses.*

Definition 2 *An* assignment *for a signed CNF formula is a mapping that assigns to every propositional variable an element of the truth value set. An assignment I* satisfies *a signed literal $S{:}p$ iff $I(p) \in S$,* satisfies *a signed clause C iff it satisfies at least one of the signed literals in C, and* satisfies *a signed CNF formula Γ iff it satisfies all clauses in Γ. A signed CNF formula is* satisfiable *iff it is satisfied by at least one assignment; otherwise it is* unsatisfiable*.*

Definition 3 *The* Signed Max-SAT problem *for a signed CNF formula consists in finding an assignment that minimizes the number of falsified signed clauses.*

Definition 4 *A* constraint satisfaction problem (CSP) *instance is defined as a triple $\langle X, D, C\rangle$, where X =*

*This research was founded by the MEC research projects iDEAS (TIN2004-04343), Mulog (TIN2004-07933-C03-01/03) and SofSAT (TIC2003-00950), and the program Ramón y Cajal.

$\{x_1, \ldots, x_n\}$ is a set of variables, $D = \{d(x_1), \ldots, d(x_n)\}$ is a set of domains containing the values the variables may take, and $C = \{C_1, \ldots, C_m\}$ is a set of constraints. Each constraint $C_i = \langle S_i, R_i \rangle$ is defined as a relation R_i over a subset of variables $S_i = \{x_{i_1}, \ldots, x_{i_k}\}$, called the constraint scope*. The relation R_i may be represented extensionally as a subset of the Cartesian product $d(x_{i_1}) \times \cdots \times d(x_{i_k})$.*

Definition 5 *An* assignment *v for a CSP instance $\langle X, D, C \rangle$ is a mapping that assigns to every variable $x_i \in X$ an element $v(x_i) \in d(x_i)$. An assignment v satisfies a constraint $\langle \{x_{i_1}, \ldots, x_{i_k}\}, R_i \rangle \in C$ iff $\langle v(x_{i_1}), \ldots, v(x_{i_k}) \rangle \in R_i$.*

Definition 6 *A* Weighted CSP (WCSP) *instance is defined as a triple $\langle X, D, C \rangle$, where X and D are variables and domains as in CSP. A constraint C_i is now defined as a pair $\langle S_i, f_i \rangle$, where $S_i = \{x_{i_1}, \ldots, x_{i_k}\}$ is the constraint scope and $f_i : d(x_{i_1}) \times \cdots \times d(x_{i_k}) \to \mathbb{N}$ is a* cost function*. The cost of a constraint C_i induced by an assignment v in which the variables of $S_i = \{x_{i_1}, \ldots, x_{i_k}\}$ take values $b_{i_1}, \ldots, b_{i_k}$ is $f_i(b_{i_1}, \ldots, b_{i_k})$. An optimal solution to a WCSP instance is a complete assignment in which the sum of the costs of the constraints is minimal.*

Definition 7 *The Weighted Constraint Satisfaction Problem (WCSP) for a WCSP instance consists in finding an optimal solution for that instance.*

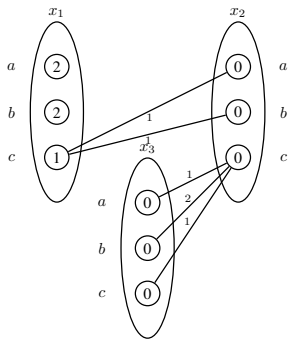
Definition 8 *The* signed encoding *of a WCSP instance $\langle X, D, C \rangle$ is the signed CNF formula over the domain $N = \bigcup_{x_i \in D} d(x_i)$ that contains for every possible tuple $\langle b_{i_1}, \ldots, b_{i_k} \rangle \in d(x_{i_1}) \times \ldots \times d(x_{i_k})$ of every constraint $\langle \{x_{i_1}, \ldots, x_{i_k}\}, f_i \rangle \in C$, $f_i(b_{i_1}, \ldots, b_{i_k})$ copies of the signed clause:*

$$\overline{\{b_{i_1}\}}{:}x_{i_1} \vee \cdots \vee \overline{\{b_{i_k}\}}{:}x_{i_k}.$$

An alternative encoding is to consider signed clauses with weights instead of allowing multiple copies of a clause. For the sake of clarity we use unweighted clauses. Nevertheless, any efficient implementation of the algorithms proposed should deal with weighted clauses. The extension of our theoretical results to weighted clauses is straightforward.

Proposition 9 *Solving a WCSP instance P is equivalent to solving the Signed Max-SAT problem of its signed encoding; i.e., the optimal cost of P coincides with the minimal number of unsatisfied signed clauses of the signed encoding of P.*

PROOF: For every combination of values to the variables of the scope of a constraint $C_i = \langle S_i, f_i \rangle$, the signed encoding contains as many clauses as the cost associated with that combination. If an assignment of the signed encoding restricted to the variables of S_i coincides with a combination of C_i with cost 0, then all the clauses of the signed encoding introduced by C_i are satisfied because there is no clause forbidding that combination. If an assignment of the signed encoding restricted to the variables of S_i coincides with a combination $\langle b_{i_1}, \ldots, b_{i_k} \rangle$ of C_i with cost u, where $u > 0$, then, by construction of the signed encoding, only the u clauses of the form $\overline{\{b_{i_1}\}}{:}x_{i_1} \vee \cdots \vee \overline{\{b_{i_k}\}}{:}x_{i_k}$ are falsified among the clauses introduced by C_i. ■



$1:\ \overline{\{a\}}{:}x_1$
$2:\ \overline{\{a\}}{:}x_1$
$3:\ \overline{\{b\}}{:}x_1$
$4:\ \overline{\{b\}}{:}x_1$
$5:\ \overline{\{c\}}{:}x_1$
$6:\ \overline{\{c\}}{:}x_1 \vee \overline{\{a\}}{:}x_2$
$7:\ \overline{\{c\}}{:}x_1 \vee \overline{\{b\}}{:}x_2$
$8:\ \overline{\{a\}}{:}x_3 \vee \overline{\{c\}}{:}x_2$
$9:\ \overline{\{b\}}{:}x_3 \vee \overline{\{c\}}{:}x_2$
$10:\ \overline{\{b\}}{:}x_3 \vee \overline{\{c\}}{:}x_2$
$11:\ \overline{\{c\}}{:}x_3 \vee \overline{\{c\}}{:}x_2$

Figure 1: A WCSP instance and its signed encoding

Example 10 Figure 1 shows a WCSP instance $\langle X, D, C \rangle$ and its signed encoding. The WCSP has the set of variables $X = \{x_1, x_2, x_3\}$ with domains $d(x_1) = d(x_2) = d(x_3) = \{a, b, c\}$. There is a binary constraint between variables x_1 and x_2, a binary constraint between variables x_2 and x_3, and a unary constraint for every variable. Unary costs are depicted inside small circles. Binary costs are depicted as labeled edges connecting the corresponding pair of values. The label of each edge is the corresponding cost. If two values are not connected, the binary cost between them is 0. In this instance, the optimal cost is 2.

3 The Inference Rule. Soundness and Completeness

We define a resolution rule for solving signed Max-SAT, called *Signed Max-SAT Resolution*, and prove its soundness and completeness. This rule was inspired by previous works [Larrosa and Heras, 2005; Bonet *et al.*, 2006] for Max-SAT. The completeness proof for signed CNF formulas is technically more involved than the proof for Boolean CNF formulas.

Definition 11 *The* Signed Max-SAT Resolution *rule is defined as follows*

$$\frac{\begin{array}{l} S{:}x \vee a_1 \vee \ldots \vee a_s \\ S'{:}x \vee b_1 \vee \ldots \vee b_t \end{array}}{\begin{array}{l} S \cap S'{:}x \vee a_1 \vee \ldots \vee a_s \vee b_1 \vee \ldots \vee b_t \\ S \cup S'{:}x \vee a_1 \vee \ldots \vee a_s \vee b_1 \vee \ldots \vee b_t \\ S{:}x \vee a_1 \vee \ldots \vee a_s \vee \overline{b_1} \\ S{:}x \vee a_1 \vee \ldots \vee a_s \vee b_1 \vee \overline{b_2} \\ \ldots \\ S{:}x \vee a_1 \vee \ldots \vee a_s \vee b_1 \vee \ldots \vee b_{t-1} \vee \overline{b_t} \\ S'{:}x \vee b_1 \vee \ldots \vee b_t \vee \overline{a_1} \\ S'{:}x \vee b_1 \vee \ldots \vee b_t \vee a_1 \vee \overline{a_2} \\ \ldots \\ S'{:}x \vee b_1 \vee \ldots \vee b_t \vee a_1 \vee \ldots \vee a_{s-1} \vee \overline{a_s} \end{array}}$$

This inference rule is applied to multisets of clauses, and replaces *the premises of the rule by its conclusions.*
We say that the rule resolves *the variable x.*

The tautologies concluded by the rule like $N{:}x \vee A$ are removed from the resulting multiset. Also we substitute clauses like $S{:}x \vee S'{:}x \vee A$ by $(S \cup S'){:}x \vee A$, and clauses like $\emptyset{:}x \vee A$ by A.

We would like to emphasize that the rule does not add the conclusions to the premises. It replaces the clauses in the premises by the clauses in the conclusions.

Definition 12 *We write $\mathcal{C} \vdash \mathcal{D}$ when the multiset of clauses $\mathcal{D}$ can be obtained from the multiset $\mathcal{C}$ applying the rule finitely many times. We write $C \vdash_x C'$ when this sequence of applications only resolves the variable x.*

In the context of Max-SAT problems, an inference rule is *sound* iff the number of falsified clauses in the premises is equal to the number of falsified clauses in the conclusions for any assignment.

Theorem 13 (Soundness) *The signed Max-SAT resolution rule is sound.*

PROOF: Let I be an arbitrary assignment. There are four cases:
1. If I falsifies the two premises, then I also falsifies the first two conclusions, and only them.
2. If I satisfies the two premises, then it also trivially satisfies the last $s+t$ clauses of the conclusion, because they are either implied by one or the other premise. The second clause of the conclusion is implied by each one of the premises. Therefore, it is also satisfied by I.

The first clause of the conclusion is not implied by the premises. However, if both premises are satisfied then we have two cases. If $S{:}x$ and $S'{:}x$ are both satisfied, then so it is $(S \cap S'){:}x$. Otherwise, either some a_i's or some b_j's is satisfied, thus also the first clause of the conclusion.
3. If I satisfies the first premise, but not the second one, then the second clause of the conclusion as well as the t following clauses are satisfied, because all them are implied by the first premise.

For the rest of conclusions, there are two cases: If some of the a_i's is satisfied, then let i be the index of such a. The assignment will satisfy the first clause of the conclusion and the last s conclusions, except $S'{:}x \vee b_1 \vee \ldots \vee b_t \vee a_1 \vee \ldots \vee a_{i-1} \vee \overline{a_i}$ that is falsified. Otherwise none of the a_i's is satisfied, and therefore, $S{:}x$ is satisfied. Hence, the first conclusion is falsified, and the last s conclusions are satisfied.
4. If I satisfies the second premise, but not the first one, the situation is analogous to previous case. ■

Definition 14 *A multiset of clauses $\mathcal{C}$ is said to be* saturated w.r.t. *x if, for every pair of clauses $C_1 = S{:}x \vee A$ and $C_2 = S'{:}x \vee B$ of $\mathcal{C}$,*
i) there are literals $S_1{:}y$ in A and $S_2{:}y$ in B such that $S_1 \cup S_2 = N$, or
ii) $S \cap S' = S$ or $S \cap S' = S'$.
A multiset of clauses $\mathcal{C}'$ is a saturation of *$\mathcal{C}$ w.r.t. x if $\mathcal{C}'$ is saturated w.r.t. x and $\mathcal{C} \vdash_x \mathcal{C}'$, i.e. $\mathcal{C}'$ can be obtained from $\mathcal{C}$ applying the inference rule resolving x finitely many times.*

We assign to every clause C a score $s(C)$ equal to the number of assignments to the variables that falsify C. The score of a multiset of clauses is the sum of scores of the clauses contained in it.

Lemma 15 *For every multiset of clauses $\mathcal{C}$ and variable x, there exists a multiset $\mathcal{C}'$ such that $\mathcal{C}'$ is a saturation of $\mathcal{C}$ w.r.t. x.*

PROOF: We proceed by applying nondeterministically the inference rule resolving x, until we obtain a saturated multiset. We only need to prove that this process terminates in finitely many inference steps, i.e that there does not exist infinite sequences $\mathcal{C} = \mathcal{C}_0 \vdash \mathcal{C}_1 \vdash \ldots$, where at every inference we resolve the variable x and none of the sets $\mathcal{C}_i$ are saturated. Let M be the score of $\mathcal{C}$.

Let us partition the multiset $\mathcal{C}$ of clauses into n multisets (n is the size of the domain), $\{B_0, B_1, \ldots, B_{n-1}\}$, where B_i contains the clauses where the cardinality of the support of x is i. Notice that B_0 is the multiset of clauses that do not contain the variable x. Let us denote by $s(B_i)$ the score of the multiset B_i.

We will look at this n multisets as a word of length n and base $M+1$. So our multiset will be represented by the number $s(B_0)\, s(B_1) \cdots s(B_{n-1})$, taking $s(B_0)$ as the most significant digit. Since B_i is a subset of $\mathcal{C}$, for $i = 0, \ldots, n-1$, $s(B_i) \leq M$.

When we apply our inference rule, we take two clauses, say one from B_i and one from B_j and substitute them by a set of clauses that we will distribute among the different B_k's. Now we have a new multiset of clauses and by the soundness of our rule the score of the new multiset is the same. But, if we again look at the multiset as a number in base M, the number will be different. We will argue that for each inference step, the number increases. Say that the clauses we do inference are $S{:}x \vee A \in B_{|S|}$ and $S'{:}x \vee B \in B_{|S'|}$. By the inference step we remove these clauses and add some clause in $B_{|S \cap S'|}$, and maybe also some clauses in $B_{|S|}$, $B_{|S'|}$ and $B_{|S \cup S'|}$. Since, by definition of saturation $S \cap S' \neq S$ and $S \cap S' \neq S'$, we know that $|S \cap S'| < |S|, |S'| < |S \cup S'|$, hence the digit of $B_{|S \cap S'|}$ is more significant than the digits of $B_{|S|}$, $B_{|S'|}$ and $B_{|S \cup S'|}$. We have to conclude that the new M-base number after the inference step is larger than before. Since the largest possible number we can obtain is the one represented as $s(B_0)s(B_1) \cdots s(B_{n-1}) = M0 \cdots 0$ the saturation procedure for x has to finish before M^n steps. ■

Lemma 16 *Let $\mathcal{E}$ be a saturated multiset of clauses w.r.t. x. Let $\mathcal{E}'$ be the subset of clauses of $\mathcal{E}$ not containing x. Then, any assignment I satisfying $\mathcal{E}'$ (and not assigning x) can be extended to an assignment satisfying $\mathcal{E}$.*

PROOF: We have to extend I to satisfy the whole $\mathcal{E}$. In fact we only need to set the value of x. Let us partition the multiset $(\mathcal{E} - \mathcal{E}')$ (multiset of clauses that contain the variable x) into two multisets: $(\mathcal{E} - \mathcal{E}')_T$ the multiset already satisfied by I, and $(\mathcal{E} - \mathcal{E}')_F$ the multiset such that the partial assignment I doesn't satisfy any of the clauses. Our aim is to show that the intersection of all the supports of x in $(\mathcal{E} - \mathcal{E}')_F$ is nonempty. This way we will extend I by assigning x to a value in the intersection of all the supports.

Since $\mathcal{E}$ is saturated, for every pair of clauses $C_1 = S{:}x \vee A$ and $C_2 = S'{:}x \vee B$ in $(\mathcal{E} - \mathcal{E}')_F$ either condition i) or ii) of the definition happens. Condition i) cannot happen because C_1 and C_2 cannot both be in $(\mathcal{E} - \mathcal{E}')_F$. Therefore, for every pair of clauses, $C_1 = S{:}x \vee A$ and $C_2 = S'{:}x \vee B$ in $(\mathcal{E} - \mathcal{E}')_F$, $S \cap S' = S$ or $S \cap S' = S'$. Now, we order all the supports of x appearing in $(\mathcal{E} - \mathcal{E}')_F$ in decreasing order of their cardinality. It is straightforward to see that every support is contained or equal to its predecessor. Particularly, the last support is equal to the intersection of all the supports, and it is non-empty. ■

Theorem 17 (Completeness) *For any multiset of clauses $\mathcal{C}$, we have*

$$\mathcal{C} \vdash \underbrace{\square, \ldots, \square}_{m}, \mathcal{D}$$

where $\mathcal{D}$ is a satisfiable multiset of clauses, and m is the minimum number of unsatisfied clauses of $\mathcal{C}$.

PROOF: Let $x_1, \ldots, x_n$ be any list of the variables of $\mathcal{C}$. We construct two sequences of multisets $\mathcal{C}_0, \ldots, \mathcal{C}_n$ and $\mathcal{D}_1, \ldots, \mathcal{D}_n$ such that

1. $\mathcal{C} = \mathcal{C}_0$,
2. for $i = 1, \ldots, n$, $\mathcal{C}_i \cup \mathcal{D}_i$ is a saturation of $\mathcal{C}_{i-1}$ w.r.t. x_i, and
3. for $i = 1, \ldots, n$, $\mathcal{C}_i$ is a multiset of clauses not containing $x_1, \ldots, x_i$, and $\mathcal{D}_i$ is a multiset of clauses containing the variable x_i.

By lemma 15, this sequences can effectively be computed: for $i = 1, \ldots, n$, we saturate $\mathcal{C}_{i-1}$ w.r.t. x_i, and then we partition the resulting multiset into a subset $\mathcal{D}_i$ containing x_i, and another $\mathcal{C}_i$ not containing this variable.

Notice that, since $\mathcal{C}_n$ does not contain any variable, it is either the empty multiset $\emptyset$, or it only contains (some) empty clauses $\{\square, \ldots, \square\}$.

Now we are going to prove that the multiset $\mathcal{D} = \bigcup_{i=1}^{n} \mathcal{D}_i$ is satisfiable by constructing an assignment satisfying it. For $i = 1, \ldots, n$, let $\mathcal{E}_i = \mathcal{D}_i \cup \ldots \cup \mathcal{D}_n$, and let $\mathcal{E}_{n+1} = \emptyset$. Notice that, for $i = 1, \ldots, n$,

1. the multiset $\mathcal{E}_i$ only contains the variables $\{x_i, \ldots, x_n\}$,
2. $\mathcal{E}_i$ is saturated w.r.t. x_i, and
3. $\mathcal{E}_i$ decomposes as $\mathcal{E}_i = \mathcal{D}_i \cup \mathcal{E}_{i+1}$, where all the clauses of $\mathcal{D}_i$ contain x_i and none of $\mathcal{E}_{i+1}$ contains x_i.

Now, we construct a sequence of assignments $I_1, \ldots, I_{n+1}$, where I_{n+1} is the empty assignment, hence satisfies $\mathcal{E}_{n+1} = \emptyset$. Now, I_i is constructed from I_{i+1} as follows. Assume by induction hypothesis that I_{i+1} satisfies $\mathcal{E}_{i+1}$. Since $\mathcal{E}_i$ is saturated w.r.t. x_i, and decomposes into $\mathcal{D}_i$ and $\mathcal{E}_{i+1}$, by lemma 16, we can extend I_{i+1} with an assignment for x_i to obtain I_i satisfy $\mathcal{E}_i$. Iterating, we get that I_1 satisfies $\mathcal{E}_1 = \mathcal{D} = \bigcup_{i=1}^{n} \mathcal{D}_i$.

Concluding, since by the soundness of the rule (Theorem 13) the inference preserves the number of falsified clauses for every assignment, $m = |\mathcal{C}_n|$ is the minimum number of unsatisfied clauses of $\mathcal{C}$. ■

4 Global Consistency in WCSP

From the proof of Theorem 17, we can extract the following exact algorithm for solving WCSP.

input: A WCSP instance P
$C_0 :=$ *signed_encoding*(P)
for $i := 1$ **to** k
  $C := saturation(C_{i-1}, x_i)$
  $\langle C_i, D_i \rangle := partition(C, x_i)$
endfor
$m := |C_k|$
$I := \emptyset$
for $i := k$ **downto** 1
  $I := I \cup [x_i \mapsto extension(x_i, I, D_i)]$
output: m, I

Given an initial WCSP instance P with k variables, this algorithm returns the minimal cost m of P and an optimal solution I.

The function $saturation(C_{i-1}, x_i)$ computes a saturation of C_{i-1} w.r.t. x_i applying the resolution rule resolving x until it gets a saturated set. Lemma 15 ensures that this process terminates, in particular that it does not cycle. As we have already said, the saturation of a multiset is not unique, but the proof of Theorem 17 does not depend on which particular saturation we take.

The function $partition(C, x_i)$ computes a partition of C, already saturated, into the subset of clauses containing x_i and the subset of clauses not containing x_i.

The function $extension(x_i, I, D_i)$ computes an assignment for x_i extending the assignment I, to satisfy the clauses of D_i according to Lemma 16. The function filters all clauses of D_i that are not satisfied by I. Then it computes the intersection of the supports for x_i of all of them, and returns one of the values of such an intersection. It returns a value from

$$\cap\{S \mid S{:}x_i \vee A \in D_i \text{ and } I \text{ falsifies } A\}$$

The argumentation of the proof of Lemma 16 ensures that this intersection is not empty.

The order on the saturation of the variables can be freely chosen, i.e. the sequence $x_1, \ldots x_n$ can be any enumeration of the variables.

A similar approach to this algorithm was defined using bucket elimination [Dechter, 1999]. Even though both procedures have the same exponential worst-case complexity, we believe that our algorithm can give rise to a better performance profile, in the sense that our computation of the joint operation is incremental.

5 Local Consistency in WCSP

In WCSP a number of local consistency properties have been proposed. These local properties do not ensure the global consistency of a set of constraints. However, they can be enforced very efficiently and used to find a lower bound of the cost.

In this section we focus on binary WCSP instances as in [Larrosa and Schiex, 2003; de Givry *et al.*, 2005]. We assume the existence of a unary constraint for every variable x_i. If no such a constraint is defined, we can always define

a dummy constraint as $f(a_k) = 0$ for every $a_k \in d(x_i)$. We will use the standard notation for binary WCSP in the literature: C_i will denote a unary constraint over a variable x_i, and C_{ij} will denote a binary constraint between variables x_i and x_j; $C_i(a_k)$, where $a_k \in d(x_i)$, will denote $f(a_k)$, and $C_{ij}(a_k, b_l)$, where $a_k \in d(x_i)$ and $b_l \in d(x_j)$, will denote $f(a_k, b_k)$.

Definition 18 *Variable x_i is* node consistent *if there exists a value $a_k \in d(x_i)$ such that $C_i(a_k) = 0$. A WCSP is node consistent (NC*) if every variable is node consistent.*

Definition 19 *Given a binary constraint C_{ij}, the value $b \in d(x_j)$ is a* simple support *for $a \in d(x_i)$ if $C_{ij}(a, b) = 0$, and is a* full support *if $C_{ij}(a, b) + C_j(b) = 0$*

Definition 20 *Variable x_i is* arc consistent *if every value $a \in d(x_i)$ has a simple support in every constraint C_{ij}. A WCSP is arc consistent (AC*) if every variable is node and arc consistent.*

Definition 21 *Variable x_i is* full arc consistent *if every value $a \in d(x_i)$ has a full support in every constraint C_{ij}. A WCSP is full arc consistent (FAC*) if every variable is node and full arc consistent.*

Definition 22 *Let $>$ be a total ordering over the variables of a WCSP. Variable x_i is* directional arc consistent *(DAC) if every value $a \in d(x_i)$ has a full support in every constraint C_{ij} such that $x_j > x_i$. It is* full directional arc consistent *(FDAC) if, in addition, every value $a \in d(x_i)$ has a simple support in every constraint C_{ij} such that $x_j < x_i$. A WCSP is full directional arc consistent (FDAC*) if every variable is node and full directional arc consistent.*

Definition 23 *Let $>$ be a total ordering over the variables of a WCSP. Variable x_i is* existential arc consistent *if there is at least one value $a \in d(x_i)$ such that $C_i(a) = 0$ and has a full support in every constraint C_{ij}. A WCSP is existential arc consistent (EAC*) if every variable is node and existential arc consistent. A WCSP is* existential directional arc consistent *(EDAC*) if it is FDAC* and EAC*.*

In what follows we define four sound inference rules for a sublanguage of signed formulas. We just consider clauses with at most two literals and whose signs are complements of singletons. This language captures binary WCSP instances. As we will see bellow, the rules enforce some known local consistency properties. For lack of space we do not show how these rules may be derived from the signed Max-SAT resolution rule.

In the next rules we assume that $N = \{i_1, \ldots, i_n\} = \{j_1, \ldots, j_n\}$ and $j \in N$.

Rule 1:

$$\frac{\overline{\{i_1\}}{:}x \\ \cdots \\ \overline{\{i_n\}}{:}x}{\square}$$

Rule 2:

$$\frac{\overline{\{i_1\}}{:}x \vee \overline{\{j\}}{:}y \\ \cdots \\ \overline{\{i_n\}}{:}x \vee \overline{\{j\}}{:}y}{\overline{\{j\}}{:}y}$$

Rule 3:

$$\frac{\overline{\{i_1\}}{:}x \vee \overline{\{j_1\}}{:}y \\ \cdots \\ \overline{\{i_s\}}{:}x \vee \overline{\{j_1\}}{:}y \\ \overline{\{i_{s+1}\}}{:}x \\ \cdots \\ \overline{\{i_n\}}{:}x}{\begin{array}{l}\overline{\{j_1\}}{:}y \\ \overline{\{i_{s+1}\}}{:}x \vee \overline{\{j_2\}}{:}y \\ \cdots \\ \overline{\{i_{s+1}\}}{:}x \vee \overline{\{j_n\}}{:}y \\ \cdots \\ \overline{\{i_n\}}{:}x \vee \overline{\{j_2\}}{:}y \\ \cdots \\ \overline{\{i_n\}}{:}x \vee \overline{\{j_n\}}{:}y\end{array}}$$

Rule 4:

$$\frac{\overline{\{i_1\}}{:}x \vee \overline{\{j_1\}}{:}y \\ \cdots \\ \overline{\{i_s\}}{:}x \vee \overline{\{j_1\}}{:}y \\ \overline{\{i_{s+1}\}}{:}x \\ \cdots \\ \overline{\{i_n\}}{:}x \\ \overline{\{j_2\}}{:}y \\ \cdots \\ \overline{\{j_n\}}{:}y}{\begin{array}{l}\square \\ \overline{\{i_{s+1}\}}{:}x \vee \overline{\{j_2\}}{:}y \\ \cdots \\ \overline{\{i_{s+1}\}}{:}x \vee \overline{\{j_n\}}{:}y \\ \cdots \\ \overline{\{i_n\}}{:}x \vee \overline{\{j_2\}}{:}y \\ \cdots \\ \overline{\{i_n\}}{:}x \vee \overline{\{j_n\}}{:}y\end{array}}$$

Lemma 24 *Star node consistency (NC*) can be enforced applying rule 1.*

PROOF: Say x_i is a variable of a WCSP that is not star node consistent. Then for every $j \in N$, $C_i(j) > 0$. Let be $w = \min\{C_i(j) \,|\, j \in N\}$ and k such that $C_i(k) = w$. This means that in the corresponding signed encoding we have $C_i(j)$ copies of $\overline{\{j\}}{:}x_i$, for all $j \in N$. Rule 1 applied to the encoding w many times will remove w copies of $\overline{\{j\}}{:}x_i$, for all $j \in N$, hence all the copies of $\overline{\{k\}}{:}x_i$. Therefore, the WCSP equivalent to the new encoding has the star node consistency property of the variable x_i. ■

Lemma 25 *Arc consistency (AC*) can be enforced applying rule 2.*

PROOF: Say x_i is a variable that is not arc consistent with respect to a constraint C_{ij}, for some variable x_j. This means that there is a value $a \in N$ such that for all $b \in N$, $C_{ij}(a, b) > 0$. Let be $w = \min\{C_{ij}(a, b) \,|\, b \in N\}$. The constrain C_{ij} will generate among others $C_{ij}(a, b)$ copies of $\overline{\{a\}}{:}x_i \vee \overline{\{b\}}{:}x_j$, for every $b \in N$. Applying rule 2 w many times, we substitute these clauses by w copies of $\overline{\{a\}}{:}x_i$ and $C_{ij}(a, b) - w$ copies of $\overline{\{a\}}{:}x_i \vee \overline{\{b\}}{:}x_j$, for every $b \in N$. Since there is one value k such that $C_{ij}(a, k) - w = 0$, this new set of clauses indicates that now variable x_i is arc consistent with respect to C_{ij}, for the value a. Arc consistency for other values would be obtained in the same way. ■

The previous two lemmas where proved for domains of size two in [Larrosa and Heras, 2005].

Lemma 26 *Fixed a total ordering $>$ on the variables, Directional Arc Consistency (DAC) can be enforced from rule 3 applied with the restriction $x > y$.*

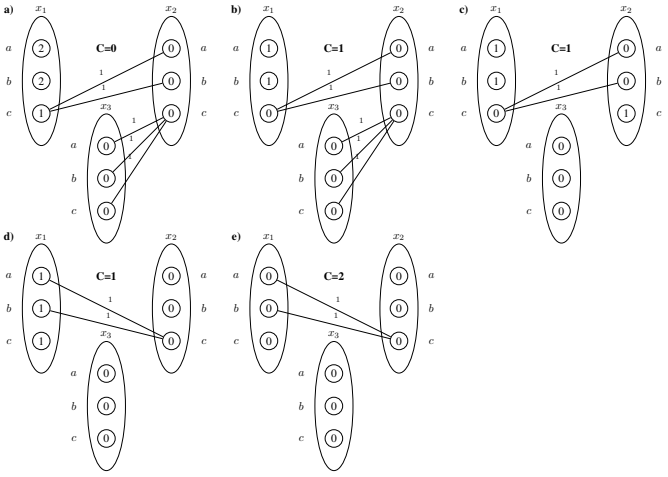
PROOF: Let $>$ be a total ordering on the variables. Say x_i is a variable that is not directional arc consistent with respect to a restriction C_{ij} for some variable x_j where

$x_j > x_i$. This means that there is a value $a \in N$ such that for all $b \in N$, $C_{ij}(a,b) + C_j(b) > 0$. Suppose that there is some b such that $C_{ij}(a,b) = 0$, otherwise we can use rule 2 to enforce arc consistency. W.l.o.g. suppose that, for $s \in \{k \dots n\}$, $C_{ij}(a,s) = 0$. So, for the same subdomain, $C_j(s) > 0$. This ensures that we have the following subset of clauses $\{\overline{\{a\}}{:}x_i \vee \overline{\{1\}}{:}x_j, \dots, \overline{\{a\}}{:}x_i \vee \overline{\{k-1\}}{:}x_j, \overline{\{k\}}{:}x_j, \dots, \overline{\{n\}}{:}x_j\}$. Rule 2 allows us to substitute this set of clauses by $\{\overline{\{a\}}{:}x_i\} \cup \bigcup_{c \neq a, b \in \{1 \dots k-1\}} \{\overline{\{c\}}{:}x_i \vee \overline{\{b\}}{:}x_j\}$. Applying rule 3 repeatedly the values $\{k \dots n\}$ of x_j become the full support for x_i. ■

Rule 4 enforces DAC*, since it combines DAC and NC*.

Rule 3 also enforces full arc consistency (FAC*), but then it must be applied without the limitation $x > y$.

Existential Arc Consistency (EAC*) can also be obtained from rule 3 but with different limitations than DAC* (in order to avoid cycles).



a)
$1 : \overline{\{a\}} : x_1$
$2 : \overline{\{a\}} : x_1$
$3 : \overline{\{b\}} : x_1$
$4 : \overline{\{b\}} : x_1$
$5 : \overline{\{c\}} : x_1$
$6 : \overline{\{c\}} : x_1 \vee \overline{\{a\}}x_2$
$7 : \overline{\{c\}} : x_1 \vee \overline{\{b\}}x_2$
$8 : \overline{\{a\}} : x_3 \vee \overline{\{c\}}x_2$
$9 : \overline{\{b\}} : x_3 \vee \overline{\{c\}}x_2$
$10 : \overline{\{c\}} : x_3 \vee \{c\}x_2$

b)
$2 : \overline{\{a\}} : x_1$
$4 : \overline{\{b\}} : x_1$
$6 : \overline{\{c\}} : x_1 \vee \overline{\{a\}}x_2$
$7 : \overline{\{c\}} : x_1 \vee \overline{\{b\}}x_2$
$8 : \overline{\{a\}} : x_3 \vee \overline{\{c\}}x_2$
$9 : \overline{\{b\}} : x_3 \vee \overline{\{c\}}x_2$
$10 : \overline{\{c\}} : x_3 \vee \overline{\{c\}}x_2$
$11 : \square$

c)
$2 : \overline{\{a\}} : x_1$
$4 : \overline{\{b\}} : x_1$
$6 : \overline{\{c\}} : x_1 \vee \overline{\{a\}}x_2$
$7 : \{c\} : x_1 \vee \{b\}x_2$
$11 : \square$
$12 : \overline{\{c\}}x_2$

d)
$2 : \overline{\{a\}} : x_1$
$4 : \overline{\{b\}} : x_1$
$11 : \square$
$13 : \overline{\{c\}} : x_1$
$14 : \overline{\{a\}} : x_1 \vee \overline{\{c\}}x_2$
$15 : \overline{\{c\}} : x_1 \vee \{c\}x_2$

e)
$11 : \square$
$14 : \overline{\{a\}} : x_1 \vee \overline{\{c\}}x_2$
$15 : \{c\} : x_1 \vee \overline{\{c\}}x_2$
$16 : \square$

Figure 2: Example of the application of the rules

Example 27 Figure 2 shows a sequence of equivalent signed Max-SAT instances obtained by applying rules 1, 2 and 3. The sequence of transformations is: a) original formula, b) application of rule 1 to clauses 1, 3 and 5, c) application of rule 2 substituting clauses 8, 9 and 10 by clause 12, d) application of rule 3 substituting clauses 6, 7 and 12 by 13, 14 and 15, and e) application of rule 1 to clauses 2, 4 and 13. The minimal cost is 2. The corresponding sequence of WCSP instances is: a) original instance, b) is NC* but not AC*, c) is AC* but not DAC*, d) is DAC* but not NC* and e) is DAC* and NC*.

6 Conclusions

We have proved that the logic of signed CNF formulas provides the underlying logic of WCSP. On the one hand, this language allows a compact and natural representation of WCSP with not only binary but n-ary constraints. On the other hand, the inference system captures and formalizes the algorithms for global as well as local consistency that have been described in the WCSP community.

We are currently investigating new derived rules that could be applied in polynomial time and enforce stronger forms of soft local consistency.

References

[Ansótegui and Manyà, 2003] C. Ansótegui, and F. Manyà. New logical and complexity results for Signed-SAT. In *Proc. of ISMVL'03*, pages 181–187, 2003.

[Bonet *et al.*, 2006] M. L. Bonet, J. Levy, and F. Manyà. A complete calculus for Max-SAT. In *Proc. of SAT'06*, pages 240–251. Springer LNCS 3569, 2006.

[de Givry *et al.*, 2005] S. de Givry, F. Heras, J. Larrosa, and M. Zytnicki. Existencial arc consistency: Getting closer to full arc consistency in weighted CSPs. In *Proc. of IJCAI'05*, pages 84–89, 2005.

[Dechter, 1999] R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1–2):41–85, 1999.

[Freuder and Wallace, 1992] E. Freuder and R. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 58:21–71, 1992.

[Larrosa and Heras, 2005] J. Larrosa and F. Heras. Resolution in Max-SAT and its relation to local consistency in weighted CSPs. In *Proc. of IJCAI'05*, pages 193–198, 2005.

[Larrosa and Meseguer, 1996] J. Larrosa and P. Meseguer. Exploiting the use of DAC in Max-CSP. In *Proc. of CP'96*, pages 308–322, 1996.

[Larrosa and Schiex, 2003] J. Larrosa and T. Schiex. In the quest of the best form of local consistency for weighted CSP. In *Proc. of IJCAI'03*, pages 239–244, 2003.

[Larrosa and Schiex, 2004] J. Larrosa and T. Schiex. Solving weighted CSP by maintaining arc-consistency. *Artificial Intelligence*, 159(1–2):1–26, 2004.

[Verfaillie *et al.*, 1996] G. Verfaillie, M. Lemaitre, and T. Schiex. Russian doll search. In *Proc. of AAAI'96*, pages 181–187, 1996.