Mapping CSP into Many-Valued SAT*

Carlos Ansótegui[1], María Luisa Bonet[2], Jordi Levy[3], and Felip Manyà[1]

[1] Universitat de Lleida (DIEI, UdL)
[2] Universitat Politècnica de Catalunya (LSI, UPC)
[3] Artificial Intelligence Research Institute (IIIA, CSIC)

Abstract. We first define a mapping from CSP to many-valued SAT which allows to solve CSP instances with many-valued SAT solvers. Second, we define a new many-valued resolution rule and prove that it is refutation complete for many-valued CNF formulas and, moreover, enforces CSP (i, j)-consistency when applied to a many-valued SAT encoding of a CSP. Instances of our rule enforce well-known local consistency properties such as arc consistency and path consistency.

1 Introduction

SAT and CSP are problem solving paradigms which have been shown to be competitive in a wide range of domains. Both the SAT community and the CSP community have devised a number of solving techniques that have been incorporated into state-of-the-art solvers. SAT techniques are better than CSP techniques for some problems, and vice versa. In this paper, we focus on inference and our goal is to explore how CSP inference can be defined in a way similar to SAT inference, which is usually defined via resolution-like inference rules. To this end, we use the formalism provided by the many-valued clausal forms known as signed CNF formulas, and define a number of resolution rules that enforce the most important local consistency properties defined in the literature.

First, we define a mapping from CSP to signed-SAT, which is the satisfiability problem of the many-valued clausal forms known as signed CNF formula [BHM00]. A CSP instance is now represented as a list of clauses, where each clause represents a no-good of a constraint. We use signed-SAT instead of SAT to capture in a natural way the information provided by the domains of CSP variables. Second, we define a new resolution rule and prove that it is refutation complete for signed CNF formulas and, moreover, enforces (i, j)-consistency when applied to a signed-SAT encoding of a CSP. Third, we show how instances of the (i, j)-consistency rule enforce well-known local consistency properties such as arc consistency and path consistency.

The fact of reformulating the main CSP consistency properties as resolution-like inference rule has some advantages: (i) they are easier to understand, at least for the SAT community; (ii) the machinery and techniques for resolution

* Research partially supported by projects iDEAS (TIN2004-04343), Mulog (TIN2004-07933-C03-01/03) and IEA (TIN2006-15662-C02-02) funded by the MEC.

J. Marques-Silva and K.A. Sakallah (Eds.): SAT 2007, LNCS 4501, pp. 10–15, 2007.
© Springer-Verlag Berlin Heidelberg 2007

developed by the automated deduction community can be easily applied to CSP; (iii) CSP and SAT inference can be compared by restricting to domains of cardinality two; and (iv) a signed-SAT solver allowing to apply different resolution rules at each node of the search tree provides a framework for analysing CSP local consistency, as well as for comparing SAT and CSP inference and eventually devise new solvers.

Our work is closely related to previous attempts to understand the relation between CSP and SAT, and vice versa (see [BHW04, Gen02, Wal00]). The advantage of our approach is the use of a formalism in which we can reformulate the inference of both SAT and CSP, instead of mapping CSP into SAT and SAT into CSP as in [AM04, BHM99, BHW04, FP01].

The results of this paper can provide new insights to the existing results about exploiting the structure of CSPs into SAT solvers [ALM03, Bac06, DS06].

2 Preliminaries

2.1 Signed CNF Formulas

Definition 1. *A* truth value set*, or* domain*, N is a non-empty finite set. A* sign *is a subset $S \subseteq N$ of truth values. The* complement *of a sign S, denoted by $\overline{S}$, is $N \setminus S$. A* signed literal *is an expression of the form $S{:}x$, where S is a sign and x is a propositional variable. The set S is also called the* support *of x. The* complement *of a signed literal l of the form $S{:}x$, denoted by $\overline{l}$, is $\overline{S}{:}x$. A* signed clause *is a disjunction of signed literals. A* signed CNF formula *is a set of signed clauses (or a conjunction of 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 $S{:}x$, if $I(x) \in S$. It* satisfies *a signed clause C, if it satisfies at least one of the signed literals in C. It* satisfies *a signed CNF formula Γ, if it satisfies all clauses in Γ.*

A signed CNF formula is satisfiable*, if it is satisfied by at least one assignment; otherwise it is* unsatisfiable*. The* signed-SAT problem *for a signed CNF formula ϕ consists of determining whether ϕ is satisfiable.*

We give now two refutationally complete inference systems for signed-SAT. The first one is defined by the next two rules on the left [Häh93], while the second one is defined by the rule on the right [Häh94].

Signed Binary Resolution

$$\frac{S{:}x \vee A \qquad S'{:}x \vee B}{S \cap S'{:}x \vee A \vee B}$$

Simplification

$$\frac{\emptyset{:}x \vee D}{D}$$

Signed Parallel Resolution

$$\frac{S_1{:}x \vee A_1 \quad \cdots \quad S_k{:}x \vee A_k}{A_1 \vee \cdots \vee A_k} \quad \text{whenever } \bigcap_{i=1}^{k} S_i = \emptyset$$

Also we assume w.l.o.g. that every variable in a clause appears only once collapsing different occurrences of a literal making the union of the supports.

2.2 Constraint Satisfaction Problems

Definition 3. *A* constraint satisfaction problem (CSP) *instance, or* constraint network*, is defined as a triple* $\langle X, D, C\rangle$*, where* $X = \{x_1, \ldots, x_n\}$ *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* $C = \{C_1, \ldots, C_p\}$ *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 4. *An* assignment *for a CSP instance* $\langle X, D, C\rangle$ *is a mapping that assigns to each variable* $x_i \in Y$*, where* $Y \subseteq X$*, a value from* $d(x_i)$*. An assignment* I *satisfies a constraint* $\langle\{x_{i_1}, \ldots, x_{i_k}\}, R_i\rangle \in C$*, if* $\langle I(x_{i_1}), \ldots, I(x_{i_k})\rangle \in R_i$*. An assignment* I *over the set of variables* Y *is consistent, if for every constraint* $C_i \in C$ *defined on variables* $Y' \subseteq Y$*,* I *restricted to* Y' *satisfies* C_i*.*

The Constraint Satisfaction Problem (CSP) consists of, given a CSP instance, finding an assignment that satisfies the instance, if it exists, or showing that it is unsatisfiable.

We next define the main local consistency properties that have been defined in the literature.

Definition 5. *A CSP is* (i, j)-consistent*, for* $i \geq 0$ *and* $j \geq 1$*, if it has non-empty domains and any consistent instantiation of* i *variables can be extended to a consistent instantiation of* j *additional variables.*

A CSP is node consistent *if it is* $(0, 1)$*-consistent, it is* arc consistent *if it is* $(1, 1)$*-consistent, and it is* path consistent *if it is* $(2, 1)$*-consistent.*

A CSP is k-consistent*, for* $k \geq 1$*, if it is* $(k-1, 1)$*-consistent.*

A CSP is strong k-consistent*, for* $k \geq 1$*, if it is* i*-consistent for every* $i \in \{1, \ldots, k\}$*.*

2.3 Mapping CSP into Signed-SAT

We define a mapping that translates a CSP instance P into a signed-SAT instance P' in such a way that P is satisfiable iff P' is satisfiable [ABLM07]. The encoding basically translates no-goods into clauses.

Definition 6. *The* signed encoding *of a CSP instance* $\langle X, D, C\rangle$ *is the signed CNF formula over the truth value set* $N = \bigcup_{x_i \in D} d(x_i)$ *that contains, for every constraint* $\langle\{x_1, \ldots, x_k\}, R\rangle \in C$ *and every possible tuple* $\langle b_1, \ldots, b_k\rangle \in d(x_1) \times \cdots \times d(x_k)$ *such that* $(b_1, \ldots, b_k) \notin R$*, the clause:*

$$\overline{\{b_1\}}{:}x_1 \vee \cdots \vee \overline{\{b_k\}}{:}x_k$$

Moreover, for every variable x *and every value* $b \in N$ *such that* $x \notin d(x)$*, we add the unary clause* $\overline{\{b\}}{:}x$ *.*

3 CSP Inference as Signed Resolution

In this section we define a sound and complete signed resolution rule, called signed (i,j)-consistency, that enforces CSP (i,j)-consistency when applied to a signed-SAT encoded CSP. Then, we show that instances of the rule enforce arc consistency and path consistency. The next lemma will help us understand the rule.

Lemma 1. *Let* $\phi = \{ S_{1,1}{:}y_1 \vee \cdots \vee S_{1,p}{:}y_p, \ldots, S_{k,1}{:}y_1 \vee \cdots \vee S_{k,p}{:}y_p \}$ *be a set of signed clauses. Then, the set of assignments that satisfies all the clauses of* ϕ *can be characterized by the set* $\bigcap_{r=1}^{k} \overline{\overline{S_{r,1}} \times \cdots \times \overline{S_{r,p}}}$.

Proof: The set $\overline{S_{r,1}} \times \cdots \times \overline{S_{r,p}}$ is exactly the set of assignments that falsify the clause $S_{r,1}{:}y_1 \vee \cdots \vee S_{r,p}{:}y_p$. Therefore $\overline{\overline{S_{r,1}} \times \ldots \times \overline{S_{r,p}}}$ is the set of assignments that satisfy it. As a conclusion, the set of assignments that satisfy all the clauses is $\bigcap_{r=1}^{k} \overline{\overline{S_{r,1}} \times \cdots \times \overline{S_{r,p}}}$. ■

Signed (i,j)-Consistency Rule:

$$\frac{\begin{array}{c} S_{1,1}{:}x_1 \vee \cdots \vee S_{1,i}{:}x_i \vee S_{1,i+1}{:}x_{i+1} \vee \cdots \vee S_{1,i+j}{:}x_{i+j} \\ \cdots \\ S_{k,1}{:}x_1 \vee \cdots \vee S_{k,i}{:}x_i \vee S_{k,i+1}{:}x_{i+1} \vee \cdots \vee S_{k,i+j}{:}x_{i+j} \end{array}}{\bigcup_{r=1}^{k} S_{r,1}{:}x_1 \vee \cdots \vee \bigcup_{r=1}^{k} S_{r,i}{:}x_i}$$

whenever $\bigcap_{r=1}^{k} \overline{\overline{S_{r,i+1}} \times \cdots \times \overline{S_{r,i+j}}} = \emptyset$, $i \geq 0$ and $j \geq 1$

Remark 1. Since we start with a no-good representation of the constraints, the initial clauses will have all the supports of the form $\overline{\{b\}}$, for some $b \in N$. Then, when we apply the rule, for every $l = 1, \ldots, i$, there exists a $b \in N$ such that, for all $r = 1, \ldots, k$, we have either $S_{r,l} = \overline{\{b\}}$ or $S_{r,l} = \emptyset$; otherwise the rule concludes a tautology. Therefore, in the conclusion of the rule $\bigcup_{r=1}^{k} S_{r,l}{:}x_l$ is either empty or has the form $\overline{\{b\}}$ for some $b \in N$; thus, the conclusion of the rule also preserves the no-good representation form.

In the (i,j)-consistency rule, the last j variables $x_{i+1}, \ldots, x_{i+j}$ are called *resolving variables*. In the (i,j)-consistency rule we can add the restriction that all variables appear in at least one clause ($\bigcup_{r=1}^{k} S_{r,l} \neq \emptyset$, for $l = 1, \ldots, i+j$). We call this version of the rule *non-strong*.

Lemma 2. *The signed* (i,j)*-consistency rule enforces CSP* (i,j)*-consistency, i.e. if the signed encoding of a CSP instance is closed by the* (i,j)*-consistency rule, then the CSP is* (i,j)*-consistent.*

The [non] strong signed $(i-1,1)$*-consistency rule enforces CSP [non] strong* i*-consistency.*

PROOF: Suppose that a set of clauses is closed by the rule, but its corresponding constraint network is not (i,j)-consistent. We have some tuple of i variables $\overline{x}$ and i consistent values $\overline{a}$ of their domains, and there exists also a tuple of j variables $\overline{y}$, such that $\overline{a}$ can not be extended to these new variables consistently. I.e. for any tuple of j values $\overline{b}$, the tuple of $i+j$ values $\overline{a,b}$ for the variables $\overline{x,y}$ falsifies some constraint about a subset of such variables (where at leas one of the y variables is present). Therefore, for any tuple $\langle b_1,\ldots,b_j\rangle$, the tuple $\langle a_1,\ldots,a_i,b_1,\ldots,b_j\rangle$ for $\langle x_1,\ldots,x_i,y_1,\ldots,y_j\rangle$ is not good, and there is a clause whose literals are a subset of $\overline{\{a_1\}}{:}x_1 \vee \cdots \vee \overline{\{a_i\}}{:}x_i \vee \overline{\{b_1\}}{:}y_1 \vee \cdots \vee \overline{\{b_j\}}{:}y_j$. Since the set of clauses is closed by the rule, and we have $\bigcap_{b_1\in N,\ldots,b_j\in N} \overline{\overline{\{b_1\}} \times \cdots \times \overline{\{b_j\}}} = \bigcap_{b_1\in N,\ldots,b_j\in N} \overline{\{b_1\} \times \cdots \times \{b_j\}} = \emptyset$ our set of clauses also contains a subclause of $\overline{\{x_1\}}{:}a_1 \vee \cdots \vee \overline{\{x_1\}}{:}a_i$ witch means that the tuple $\langle a_1,\ldots,a_i\rangle$ is not good for $\langle x_1,\ldots,x_i\rangle$ and this contradicts the assumption. The proof of the second part of the lemma has the same ingredients as the first. ■

Theorem 1. *The signed (i,j)-consistency rule defines a sound and complete resolution system for signed CNF formulas.*

PROOF: When $j=1$, the signed $(i,1)$-consistency rule is the signed parallel resolution rule. So, already the signed $(i,1)$-consistency rule is complete.

To see that it is a sound rule, notice that, by Lemma 1, since $\bigcap_{r=1}^{k} \overline{\overline{S_{r,i+1}} \times \cdots \times \overline{S_{r,i+j}}} = \emptyset$, the set of clauses $\{S_{1,i+1} : x_{i+1} \vee \cdots \vee S_{1,i+j} : x_{i+j}, \ldots, S_{k,i+1} : x_{i+1} \vee \cdots \vee S_{k,i+j} : x_{i+j}\}$ is unsatisfiable. By the completeness of the parallel resolution rule we can obtain the empty clause from them. Now, from this refutation we do the following transformation. We change the set of premises by $\{\, S_{1,1}{:}x_1 \vee \cdots \vee S_{1,i}{:}x_i \vee S_{1,i+1}{:}x_{i+1} \vee \cdots \vee S_{1,i+j}{:}x_{i+j}\,, \ldots,\, S_{k,1}{:}x_1 \vee \cdots \vee S_{k,i}{:}x_i \vee S_{k,i+1}{:}x_{i+1} \vee \cdots \vee S_{k,i+j}{:}x_{i+j}\,\}$. The rest of the proof is identical, but keeping the appended parts along. At this point we will not produce the empty clause, but the clause $\bigcup_{r=1}^{k} S_{r,1}{:}x_1 \vee \cdots \vee \bigcup_{r=1}^{k} S_{r,i}{:}x_i$. ■

Arc Consistency Rule:

$$\frac{\begin{array}{l}\overline{\{a\}}{:}x \vee \overline{\{j_1\}}{:}y \\ \cdots \\ \overline{\{a\}}{:}x \vee \overline{\{j_s\}}{:}y \\ \qquad\quad \overline{\{j_{s+1}\}}{:}y \\ \qquad\quad \cdots \\ \qquad\quad \overline{\{j_m\}}{:}y\end{array}}{\overline{\{a\}}{:}x}$$

where $s \geq 1$ and $\{j_1,\ldots,j_m\} = N$

Path Consistency Rule:

$$\frac{\begin{array}{l}\overline{\{a\}}{:}x \vee \overline{\{j_1\}}{:}z \\ \qquad\quad \cdots \\ \overline{\{a\}}{:}x \vee \overline{\{j_s\}}{:}z \\ \qquad\quad \overline{\{b\}}{:}y \vee \overline{\{j_{s+1}\}}{:}z \\ \qquad\quad \cdots \\ \qquad\quad \overline{\{b\}}{:}y \vee \overline{\{j_r\}}{:}z \\ \qquad\quad \overline{\{j_{r+1}\}}{:}z \\ \qquad\quad \cdots \\ \qquad\quad \overline{\{j_m\}}{:}z\end{array}}{\overline{\{a\}}{:}x \vee \overline{\{b\}}{:}y}$$

where $r > s \geq 1$ and $\{j_1,\ldots,j_m\} = N$

Fig. 1. Instances of the non-strong signed (i,j)-consistency rule

In Figure 1 we present instances of the non-strong signed (i, j)-consistency rule that enforce local consistency properties like arc consistency and path consistency. We use supports of the form $\{\overline{b}\}$ given that we have already shown that this form of signs is preserved by inferences (see Remark 1). Basically, the arc consistency rule reduces the domain of the variable x to exclude the value a when it does not have support in y. The algorithm that enforces path consistency works by removing all the satisfying pairs of a constraint that cannot be extended to another variable in the way just defined.

References

[ABLM07] C. Ansótegui, M. Bonet, J. Levy, and F. Manyà. The logic behind weighted CSP. In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence, IJCAI'07*, pages 32–37, 2007.

[ALM03] C. Ansótegui, J. Larrubia, and F. Manyà. Boosting Chaff's performance by incorporating CSP heuristics. In *Proc. of the 9th Int. Conf. on Principles and Practice of Constraint Programming, CP'03*, pages 96–107. Springer LNCS 2833, 2003.

[AM04] C. Ansótegui and F. Manyà. Mapping problems with finite-domain variables into problems with boolean variables. In *Proc. of the 7th Int. Conf. on Theory and Applications of Satisfiability Testing, SAT'04*, pages 1–15. Springer LNCS 3542, 2004.

[Bac06] F. Bacchus. CSPs: Adding structure to SAT. In *Proc. of the 9th Int. Conf. on Theory and Applications of Satisfiability Testing, SAT'06*, page 10. Springer LNCS 4121, 2006.

[BHM99] B. Beckert, R. Hähnle, and F. Manyà. Transformations between signed and classical clause logic. In *Proc. of the 29th Int. Symp. on Multiple-Valued Logics, ISMVL'99*, pages 248–255, 1999.

[BHM00] B. Beckert, R. Hähnle, and F. Manyà. The SAT problem of signed CNF formulas. In *Labelled Deduction*, volume 17 of *Applied Logic Series*, pages 61–82. Kluwer, Dordrecht, 2000.

[BHW04] C. Bessière, E. Hebrard, and T. Walsh. Local consistencies in SAT. In *Proc. of the 6th Int. Conf. on Theory and Applications of Satisfiability Testing, SAT'03*, pages 299–314. Springer LNCS 2919, 2004.

[DS06] Y. Dimopoulos and K. Stergiou. Propagation in CSP and SAT. In *Proc. of the 12th Int. Conf. on Principles and Practice of Constraint Programming, CP'06*, pages 137–151. Springer LNCS 4204, 2006.

[FP01] A. M. Frisch and T. J. Peugniez. Solving non-boolean satisfiability problems with stochastic local search. In *Proc. of the Int. Joint Conf. on Artificial Intelligence, IJCAI'01*, pages 282–288, 2001.

[Gen02] I. P. Gent. Arc consistency in SAT. In *Proc. of the 15th European Conf. on Artificial Intelligence, ECAI'02*, pages 121–125, 2002.

[Häh93] R. Hähnle. Short CNF in finitely-valued logics. In *Proc., Int. Symp. on Methodologies for Intelligent Systems, ISMIS'93*, pages 49–58. Springer LNCS 689, 1993.

[Häh94] R. Hähnle. Efficient deduction in many-valued logics. In *Proc. of the Int. Symp. on Multiple-Valued Logics, ISMVL'94*, pages 240–249. IEEE Press, 1994.

[Wal00] T. Walsh. SAT v CSP. In *Proc. of the 6th Int. Conf. on Principles of Constraint Programming, CP'00*, pages 441–456. Springer LNCS 1894, 2000.