A Generalized Framework for Conflict Analysis

G. Audemard[1], L. Bordeaux[2], Y. Hamadi[2], S. Jabbour[1], and L. Sais[1]

[1] CRIL - CNRS UMR 8188, Artois, France
{audemard,jabbour,sais}@cril.fr
[2] Microsoft Research Cambridge, UK
{lucasb,youssefh}@microsoft.com

Abstract. This paper presents an extension of Conflict Driven Clauses Learning (CDCL). It relies on an extended notion of implication graph containing additional arcs, called inverse arcs. These are obtained by taking into account the satisfied clauses of the formula, which are usually ignored by conflict analysis. This extension captures more conveniently the whole propagation process, and opens new perspectives for CDCL-based approaches. Among other benefits, our extension leads to a new conflict analysis scheme that exploits the additional arcs to back-jump to higher levels. Experimental results show that the integration of our generalized conflict analysis scheme within two state-of-the-art solvers improves their performance.

1 Introduction

This paper extends *Conflict-Driven Clause-Learning (CDCL)*, which is one of the key components of modern SAT solvers [7,5]. In the CDCL approach a central data-structure is the *implication graph*, which records the partial assignment that is under construction together with its implications. This data-structure enables conflict analysis, which, in turn, is used for intelligent backtracking, clause learning, for the adjustment of the variable selection heuristic. An important observation is that the implication graph built in the traditional way is ”incomplete” in that it only gives a partial view of the actual implications between literals. A solver only keeps track of the first explanation that is encountered for the deduced literal. This strategy is obviously very much dependent on the particular order in which clauses are propagated. We present here an extended notion of implication graph in which a deduced literal can have several explanations. An extended version of our work can be found in [1]. The paper is organized as follows, definitions and notations are presented in the next section. Section three describes classical conflict analysis. Section four presents our extension, and finally before the conclusion, section five presents some experimental results.

2 Preliminary Definitions and Notations

A *CNF formula* $\mathcal{F}$ is a set (interpreted as a conjunction) of *clauses*, where a clause is a set (interpreted as a disjunction) of *literals*. A literal is a positive (x) or negated ($\neg x$) propositional variable. The two literals x and $\neg x$ are called *complementary*. We note $\bar{l}$

the complementary literal of l. For a set of literals L, $\bar{L}$ is defined as $\{\bar{l} \mid l \in L\}$. A *unit* clause is a clause with only one literal (called *unit literal*). An *empty clause*, noted $\perp$, is interpreted as false, while an *empty CNF formula*, noted $\top$, is interpreted as true.

The set of variables occurring in $\mathcal{F}$ is noted $V_{\mathcal{F}}$. A set of literals is *complete* if it contains one literal for each variable in $V_{\mathcal{F}}$, and *fundamental* if it does not contain complementary literals. An *interpretation* ρ of a boolean formula $\mathcal{F}$ associates a value $\rho(x)$ to some of the variables $x \in \mathcal{F}$. An interpretation is alternatively represented by a complete and fundamental set of literals, in the obvious way. A *model* of a formula $\mathcal{F}$ is an interpretation ρ that satisfies the formula; noted $\rho \vDash \Sigma$.

The following notations will be heavily used throughout the paper:

– $\eta[x, c_i, c_j]$ denotes the *resolvent* between a clause c_i containing the literal x and c_j a clause containing the literal $\neg x$. In other words $\eta[x, c_i, c_j] = c_i \cup c_j \backslash \{x, \neg x\}$.
– $\mathcal{F}|_x$ will denote the formula obtained from $\mathcal{F}$ by assigning x the truth-value *true*. Formally $\mathcal{F}|_x = \{c \mid c \in \mathcal{F}, \{x, \neg x\} \cap c = \emptyset\} \cup \{c \backslash \{\neg x\} \mid c \in \mathcal{F}, \neg x \in c\}$. This notation is extended to interpretations: given an interpretation $\rho = \{x_1, \ldots, x_n\}$, we define $\mathcal{F}|_\rho = (\ldots((\mathcal{F}|_{x_1})|_{x_2}) \ldots |_{x_n})$.
– $\mathcal{F}^*$ denotes the formula $\mathcal{F}$ closed under unit propagation, defined recursively as follows: (1) $\mathcal{F}^* = \mathcal{F}$ if $\mathcal{F}$ does not contain any unit clause, (2) $\mathcal{F}^* = \perp$ if $\mathcal{F}$ contains two unit-clauses $\{x\}$ and $\{\neg x\}$, (3) otherwise, $\mathcal{F}^* = (\mathcal{F}|_x)^*$ where x is the literal appearing in a unit clause of $\mathcal{F}$.

Let us now introduce some notations and terminology on SAT solvers based on the DPLL backtrack search procedure [3]. At each node the assigned literals (decision literal and the propagated ones) are labeled with the same *decision level* starting from 1 and increased at each branching. The current decision level is the highest decision level in the assignment stack. After backtracking, some variables are unassigned, and the current decision level is decreased accordingly. At level i, the current partial assignment ρ can be represented as a sequence of decision-propagations of the form $\langle (x_k^i), x_{k_1}^i, x_{k_2}^i, \ldots, x_{k_{n_k}}^i \rangle$ where the first literal x_k^i corresponds to the decision literal x_k assigned at level i and each $x_{k_j}^i$ for $1 \leq j \leq n_k$ represents a propagated (unit) literal at level i. Let $x \in \rho$, we note $l(x)$ the assignment level of x, $d(\rho, i) = x$ if x is the decision literal assigned at level i. For a given level i, we define ρ^i as the projection of ρ to literals assigned at a level $\leq i$.

3 Conflict Analysis Using Implication Graphs

Implication graphs capture the variable assignments ρ made during the search, both by branching and by propagation. This representation is a convenient way to analyze conflicts. In classical SAT solvers, whenever a literal y is propagated, we keep a reference to the clause at the origin of the propagation of y, which we note $\overrightarrow{cla}(y)$. The clause $\overrightarrow{cla}(y)$ is in this case of the form $(x_1 \vee \cdots \vee x_n \vee y)$ where every literal x_i is false under the current partial assignment ($\rho(x_i) = \mathit{false}, \forall i \in 1..n$), while $\rho(y) = \mathit{true}$. When a literal y is not obtained by propagation but comes from a decision, $\overrightarrow{cla}(y)$ is undefined, which we note for convenience $\overrightarrow{cla}(y) = \perp$.

When $\overrightarrow{cla}(y) \neq \perp$, we denote by $exp(y)$ the set $\{\overline{x} \mid x \in \overrightarrow{cla}(y) \setminus \{y\}\}$, called set of *explanations* of y. In other words if $\overrightarrow{cla}(y) = (x_1 \vee \cdots \vee x_n \vee y)$, then the explanations are the literals $\overline{x_i}$ with which $\overrightarrow{cla}(y)$ becomes the unit clause $\{y\}$. Note that for all i we have $l(\overline{x_i}) \leq l(y)$, i.e., all the explanations of the deduction come from a level at most as high. When $\overrightarrow{cla}(y)$ is undefined we define $exp(y)$ as the empty set. The explanations can alternatively be seen as an implication graph, in which the set of predecessors of a node corresponds to the set of explanations of the corresponding literal:

Definition 1 (Implication Graph). *Let $\mathcal{F}$ be a CNF formula, ρ a partial ordered interpretation, and let exp denote a choice of explanations for the deduced literals in ρ. The implication graph associated to $\mathcal{F}$, ρ and exp is $(\mathcal{N}, \mathcal{E})$ where:*

- *$\mathcal{N} = \rho$, i.e. there is exactly one node for every literal, decision or implied;*
- *$\mathcal{E} = \{(x, y) \mid x \in \rho, y \in \rho, x \in \mathrm{exp}(y)\}$*

Example 1. $\mathcal{G}^{\rho}_{\mathcal{F}}$, shown in Figure 1 and restricted to plain arcs is an implication graph for the formula $\mathcal{F}$ and the partial assignment ρ given below : $\mathcal{F} \supseteq \{c_1, \ldots, c_9\}$

$(c_1)\ x_6 \vee \neg x_{11} \vee \neg x_{12}$	$(c_2)\ \neg x_{11} \vee x_{13} \vee x_{16}$	$(c_3)\ x_{12} \vee \neg x_{16} \vee \neg x_2$
$(c_4)\ \neg x_4 \vee x_2 \vee \neg x_{10}$	$(c_5)\ \neg x_8 \vee x_{10} \vee x_1$	$(c_6)\ x_{10} \vee x_3$
$(c_7)\ x_{10} \vee \neg x_5$	$(c_8)\ x_{17} \vee \neg x_1 \vee \neg x_3 \vee x_5 \vee x_{18}$	$(c_9)\ \neg x_3 \vee \neg x_{19} \vee \neg x_{18}$

$\rho = \{\langle \ldots \neg x_6^1 \ldots \neg x_{17}^1 \rangle \langle (x_8^2) \ldots \neg x_{13}^2 \ldots \rangle \langle (x_4^3) \ldots x_{19}^3 \ldots \rangle \ldots \langle (x_{11}^5) \ldots \rangle\}$. The current decision level is 5.

We consider ρ, a partial assignment such that $(\mathcal{F}|_\rho)* = \perp$ and $\mathcal{G}^{\rho}_{\mathcal{F}} = (\mathcal{N}, \mathcal{E})$ the associated implication graph. Assume that the current decision level is m. As a conflict is reached, then $\exists x \in$ st. $\{x, \neg x\} \subset \mathcal{N}$ and $l(x) = m$ or $l(\neg x) = m$. Conflict analysis is based on applying resolution from the top to the bottom of the implication graph using the different clauses of the form $(\overline{exp(y)} \vee y)$ implicitly encoded at each node $y \in \mathcal{N}$. We call this process a conflict resolution proof. More formally,

Definition 2 (Asserting clause). *A clause c of the form $(\alpha \vee x)$ is called an asserting clause iff $\rho(c) =$ false, $l(x) = m$ and $\forall y \in \alpha, l(y) < l(x)$. x is called asserting literal, which we note in short $\mathcal{A}(c)$. We can define $jump(c) = max\{l(\neg y) \mid y \in \alpha\}$.*

Definition 3 (Conflict resolution proof). *A conflict resolution proof π is a sequence of clauses $\langle \sigma_1, \sigma_2, \ldots \sigma_k \rangle$ satisfying the following conditions :*

1. *$\sigma_1 = \eta[x, \overrightarrow{cla}(x), \overrightarrow{cla}(\neg x)]$, where $\{x, \neg x\}$ is the conflict.*
2. *σ_i, for $i \in 2..k$, is built by selecting a literal $y \in \sigma_{i-1}$ for which $\overrightarrow{cla}(\overline{y})$ is defined. We then have $y \in \sigma_{i-1}$ and $\overline{y} \in \overrightarrow{cla}(\overline{y})$: the two clauses resolve. The clause σ_i is defined as $\eta[y, \sigma_{i-1}, \overrightarrow{cla}(\overline{y})]$;*
3. *σ_k is, moreover an asserting clause.*

It is called elementary iff $\nexists i < k$ s.t. $\langle \sigma_1, \sigma_2, \ldots \sigma_i \rangle$ is also a conflict resolution proof.

4 Extended Implication Graph

In modern SAT solvers, clauses containing a literal x that is implied at the current level are essentially ignored by the propagation. More precisely, because the solver does not maintain the information whether a given clause is satisfied or not, a clause containing x may occasionally be considered by the propagation, but only when another literal y of the clause becomes false. When this happens the solver typically skips the clause. However, in cases where x is true *and all the other literals are false*, an "arc" was revealed for free that could as well be used to extend the graph. Such arcs are those we propose to use in our extension.

To explain our idea let us consider, again, the formula $\mathcal{F}$ and the partial assignments given in the example 1. We define a new formula $\mathcal{F}'$ as follow : $\mathcal{F}' \supseteq \{c_1, \ldots, c_9\} \cup \{c_{10}, c_{11}, c_{12}\}$ where $c_{10} = (\neg x_{19} \vee x_8), c_{11} = (x_{19} \vee x_{10})$ and $c_{12} = (\neg x_{17} \vee x_{10})$.

The three added clauses are satisfied under the instantiation ρ. c_{10} is satisfied by x_8 assigned at level 2, c_{11} is satisfied by x_{19} at level 3, and c_{12} is satisfied by $\neg x_{17}$ at level 1. This is shown in the extended implication graph (see Figure 1) by the doted edges. Let us now illustrate the usefulness of our proposed extension. Let us consider again the the asserting clause Δ_1 corresponding to the classical first UIP. We can generate the following strong asserting clause: $c_{13} = \eta[x_8, \Delta_1, c_{10}] = (x_{17}^1 \vee \neg x_{19}^3 \vee x_{10}^5)$, $c_{14} = \eta[x_{19}, c_{13}, c_{11}] = (x_{17}^1 \vee x_{10}^5)$ and $\Delta_1^s = \eta[x_{17}, c_{14}, c_{12}] = x_{10}^5$. In this case we backtrack to the level 0 and we assign x_{10} to *true*. Indeed $\mathcal{F}' \models x_{10}$.

As we can see Δ_1^s subsumes Δ_1. If we continue the process we also obtain other strong asserting clauses $\Delta_2^s = (\neg x_4^3 \vee x_2^5)$ and $\Delta_3^s = (\neg x_4^3 \vee x_{13}^2 \vee x_6^1 \vee \neg x_{11}^5)$ which subsume respectively Δ_2 and Δ_3. This first illustration gives us a new way to minimize the size of the asserting clauses.

If we take a look to the clauses used in the implication graph $\mathcal{G}_{\mathcal{F}}^{\rho}$ (plain edges) all have the following properties: (1) $\forall x \in \mathcal{N}$ the clause $c = (\overline{exp(x)} \vee x)$ is satisfied by only one literal i.e. $\rho(x) =$ *true* and $\forall y \in exp(x)$, we have $\rho(y) =$ *true* and (2) $\forall y \in exp(x), l(\neg y) \leq l(x)$. Now in the extended implication graph, the added clauses satisfy property (1) and, in addition, the property (2') $\exists y \in exp(x)$ st. $l(\neg y) > l(x)$.

Let us now explain briefly how the extra arcs can be computed. Usually unit propagation does not keep track of implications from the satisfiable sub-formula. In this extension the new implications (deductions) are considered. For instance in the previous

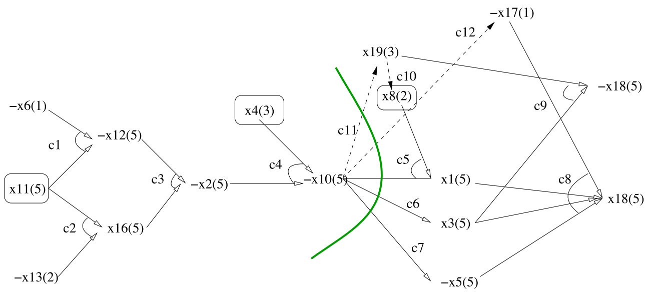

Fig. 1. Implication graph / extended implication graph

example, when we deduce x_{19} at level 3, we "rediscover" the deduction x_8 (which was a choice (decision literal) at level 2). Our proposal keeps track of these re-discoveries.

Before introducing the formal definition of our extended Implication Graph, we introduce the concept of inverse implication (inverse edge).

We maintain additionally to the classical clause $\overrightarrow{cla}(x)$ a new clause $\overleftarrow{cla}(x)$ of the form $(x \vee y_1 \vee \cdots \vee y_n)$. This clause is selected so that $\rho(y_i) = \textit{false}$ for $i \in 1..n$; $\rho(x) = \textit{true}$; and $\exists i.\ l(y_i) > l(x)$. This clause can be undefined in some cases (which we note $\overleftarrow{cla}(x) = \perp$). Several clauses of this form can be found for each literal, in which case one is selected arbitrarily: one can choose to consider the first one in the ordering. (It is easy to define a variant where we would take into account all of them, in which case $\overleftarrow{cla}(x)$ is a set of clauses; but we won't develop this variant).

We denote by $\overleftarrow{exp}(x)$ the set $\{\overline{y} \mid y \in \overleftarrow{cla}(x) \setminus \{x\}\}$, and, for clarity, by $\overrightarrow{exp}(x)$ the set that was previously noted exp. An extended implication graph is defined as follows (note that this graph is now not acyclic in general):

Definition 4 (Extended Implication Graph). *Let $\mathcal{F}$ be a CNF formula and ρ an ordered partial interpretation. We define the extended implication Graph associated to $\mathcal{F}$ and ρ as $\mathcal{G}s^{\rho}_{\mathcal{F}} = (\mathcal{N}, \mathcal{E} \cup \mathcal{E}')$ where, $\mathcal{N} = \rho$, $\mathcal{E} = \{(x, y) \mid x \in \rho, y \in \rho, x \in \overrightarrow{exp}(y)\}$ and $\mathcal{E}' = \{(x, y) \mid x \in \rho, y \in \rho, x \in \overleftarrow{exp}(y)\}$*

4.1 Learning to Back-Jump : A First Extension

In this section, we describe a first possible extension of CDCL approach using extended implication graph. Our approach makes an original use of inverses arcs to back-jump farther, i.e. to improve the back-jumping level of the classical asserting clauses.

Let us illustrate the main idea behind our proposed extension. Our approach works in three steps. In the first step (1) : an asserting clause, say $\sigma_1 = (\neg x^1 \vee \neg y^3 \vee \neg z^7 \vee \neg a^9)$ is learnt using the usual learning scheme where 9 is the current decision level. As $\rho(\sigma_1) = false$, usually we backtrack to level $jump(\sigma_1) = 7$. In the second step (2): our approach aims to eliminate the literal $\neg z^7$ from σ_1 using the new arcs of the extended graph. Let us explain this second and new processing. Let $c = (z^7 \vee \neg u^2 \vee \neg v^9)$ such that $\rho(z) = true$, $\rho(u) = true$ and $\rho(v) = true$. The clause c is an inverse arc i.e. the literal z assigned at level 7 is implied by the two literals u and v respectively assigned at level 2 and 9. From c and σ_1, a new clause $\sigma_2 = \eta[z, c, \sigma_1] = (\neg x^1 \vee \neg u^2 \vee \neg y^3 \vee \neg v^9 \vee \neg a^9)$ is generated. We can remark that the new clause σ_2 contains two literals from the current decision level 9. In the third step (3), using classical learning, one can search from σ_2 for another asserting clause σ_3 with only one literal from the current decision level. Let us note that the new asserting clause σ_3 might be worse in terms of back-jumping level. To avoid this main drawback, the inverse arc c is chosen if the two following conditions are satisfied : i) the literals of c assigned at the current level (v^9) has been already visited during the first step and ii) all the other literals of c are assigned before the level 7 i.e. level of z. In this case, we guaranty that the new asserting clause satisfies the following property : $jump(\sigma_3) \leq jump(\sigma_1)$. Moreover, the asserting literal of σ_3 is $\neg a$.

One can iterate the previous process on the new asserting clause σ_3 to eliminate the literals of σ_3 assigned at level $jump(\sigma_3)$ (for more details see [1]).

5 Experiments

Our extended learning scheme can be crafted to any CDCL based solver. See [1] for details. The experimental results reported in this section are obtained on a Xeon 3.2 GHz (2 GB RAM) and performed on a large panel of SAT instances (286) coming from SAT RACE2006 and SAT07 (industrial). All instances are simplified by the satellite preprocessor [4]. Time limit is set to 1800 seconds and results are reported in seconds. We implement our proposed extension to Minisat [5] and Rsat [8] and make a comparison between original solvers and extended ones (called Minisat^E and Rsat^E). Figure 2 shows the $time$ (t) (figure on the left-hand side) and the $cumulated\ time$ (ct) (figure on the right hand side) needed to solve a given number of instances ($nb\ instances$). t and ct represent respectively the number of instances with running time less than t seconds and the number of solved instances if we consider that all the instances are run sequentially within a time limit of t seconds. This global view clearly shows that as t or ct increase the extended versions solve more instances than the original ones.

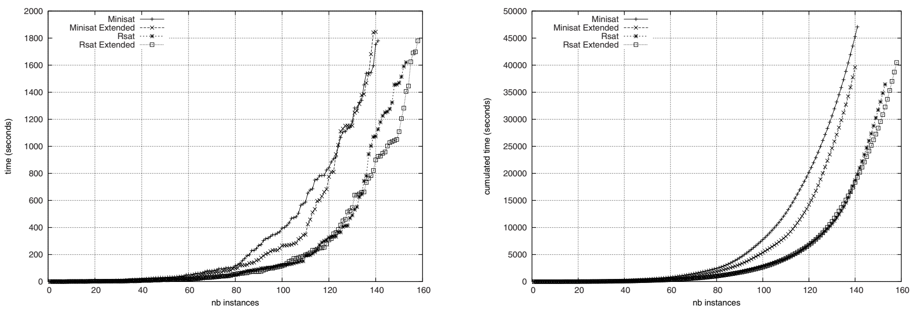

Fig. 2. Time and cumulated time for solving a given number of instances

6 Conclusion

In this paper, we have proposed a generalized framework for conflict analysis. This generalization is obtained by an original extension of the classical implication graph. This extension is obtained by considering clauses that come from the satisfiable part of the formula. Several learning schemes can be defined from this extension. The first extension of learning that improves the classical asserting clauses in term of back-jumping level shows the great potential of the new framework. Despite the different restrictions, our approach achieves interesting improvements of the SAT solvers (Rsat and MiniSat).

References

1. Audemard, G., Bordeaux, L., Hamadi, Y., Jabbour, S., Sais, L.: A Generalized Framework for Conflict Analysis. Microsoft Research, MSR-TR-2008-34 (2008)
2. Beame, P., Kautz, H., Sabharwal, A.: Towards understanding and harnessing the potential of clause learning. JAIR 22, 319–351 (2004)

3. Davis, M., Logemann, G., Loveland, D.W.: A machine program for theorem-proving. Communications of the ACM 5(7), 394–397 (1962)
4. Eén, N., Biere, A.: Effective preprocessing in SAT through variable and clause elimination. In: Bacchus, F., Walsh, T. (eds.) SAT 2005. LNCS, vol. 3569, pp. 61–75. Springer, Heidelberg (2005)
5. Eén, N., Sörensson, N.: An extensible sat-solver. In: Giunchiglia, E., Tacchella, A. (eds.) SAT 2003. LNCS, vol. 2919, Springer, Heidelberg (2004)
6. Marques-Silva, J.: Personal communication
7. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: Engineering an efficient SAT solver. In: Proceedings of (DAC 2001), pp. 530–535 (2001)
8. Pipatsrisawat, K., Darwiche, A.: Rsat 2.0: Sat solver description. Technical Report D–153, Automated Reasoning Group, Computer Science Department, UCLA (2007)