NP-completeness of small conflict set generation for congruence closure

Andreas Fellner[1,2], Pascal Fontaine[3] and Bruno Woltzenlogel Paleo[2,5*]

[1] Austrian Institute of Technology (Austria)
[2] Vienna University of Technology (Austria)
[3] Inria, Loria, U. of Lorraine (France)
[4] Australian National University (Australia)

Abstract

The efficiency of Satisfiability Modulo Theories (SMT) solvers is dependent on the capability of theory reasoners to provide small conflict sets, i.e. small unsatisfiable subsets from unsatisfiable sets of literals. Decision procedures for uninterpreted symbols (i.e. congruence closure algorithms) date back from the very early days of SMT. Nevertheless, to our best knowledge, the complexity of the smallest conflict set generation for sets of literals with uninterpreted symbols and equalities had not yet been determined, although it is believed to be NP-complete. We provide here an NP-hardness proof, using a simple reduction from SAT.

Introduction

Satisfiability Modulo Theory solvers are nowadays based on a cooperation of a propositional satisfiability (SAT) solver and a theory reasoner for the combination of theories understood by the SMT solver. The propositional structure of the problem is handled by the SAT solver, whereas the theory reasoner only has to deal with conjunctions of literals. Very schematically (we refer to [1] for more details) the Boolean abstraction of the SMT problem is repeatedly refined by adding theory conflict clauses that eliminate spurious models of the abstraction, until either unsatisfiability is reached, or a model of the SMT formula is found. Refinements can be done by refuting models of the propositional abstraction one at a time. It is, however, much more productive to refute all propositional models that are spurious for the same reason. A model of the abstraction is spurious if the set of concrete literals corresponding to the abstracted literals satisfied by this model is unsatisfiable modulo the theory. Given such an unsatisfiable set of concrete literals, the disjunction of the negations of any unsatisfiable subset (a.k.a. *core*) is a suitable conflict clause. By backtracking and asserting the conflict clause, the SAT-solver is prevented from generating the spurious model again. The smaller the clause, the stronger it is and the more spurious models it prevents. Therefore, an optimal conflict clause, corresponding to a minimal unsatisfiable subset of literals (i.e. such that all its proper subsets are satisfiable) or even a minimum one (i.e. smallest among the minimals) is desirable. This feature of the theory reasoners to *generate small conflict sets* (a name adopted in [1]) from their input is also referred to as *proof production* [8, 9] or *explanation generation* [10].

Decision procedures for the theory of uninterpreted symbols and equality can be based on congruence closure [7, 4, 10]. The decision problem is polynomial and even quasi-linear [4] with

*This work has been partially supported by the project ANR-13-IS02-0001 of the Agence Nationale de la Recherche, by the European Union Seventh Framework Programme under grant agreement no. 295261 (MEALS), by an AW APART Stipendium, by the STIC AmSud MISMT, by FWF S11407-N23 (RiSE/SHiNE), and by ERC Start Grant (279307: Graph Games).

respect to the number of terms and literals in the input set. Producing minimal conflict sets also takes polynomial time. Indeed, testing if a set S remains unsatisfiable after removal of one of its literal is also polynomial. It suffices then to repeatedly test the $|S|$ literals of S to check if they can be removed. The set S pruned of its unnecessary literals is minimal. One can also make profit of the incrementality of the decision procedure [6].

It has also been common knowledge that computing minimum conflict clauses for the theory of uninterpreted symbols and equality is a difficult problem. But, to our best knowledge, the complexity of the smallest conflict clause generation for sets of literals with uninterpreted symbols and equalities has never been established. It is mentioned to be NP-complete in [10] — with a reference to a private communication with Ashish Tiwari — but neither the authors of [10] nor Ashish Tiwari published a written proof of this fact[1].

Our interest in this problem arose from our work on Skeptik [2], a tool for the compression of proofs generated by SAT and SMT solvers. For the sake of moving beyond the purely propositional level, we have developed an algorithm for compressing congruence closure proofs, which consists of regenerating (possibly smaller) congruence closure conflict clauses while traversing the proof. Congruence closure conflict clauses are typically generated from paths in the *congruence graph* maintained by the congruence closure algorithm [5, 10, 9]. During the replay, we use a polynomial-time algorithm for searching for short paths in the congruence graph, which is a modification of Dijkstra's shortest path algorithm [3]. This raised the question whether our algorithm could find the shortest conflict clauses, as Dijkstra's algorithm finds shortest paths. We answered this question negatively by proving that the problem of deciding whether a shorter conflict clause exists is NP-hard. The goal of this short paper is to present this proof.

Preliminaries

We assume knowledge of propositional logic and quantifier-free first order logic with equality and uninterpreted symbols. We only enumerate the notions and notations used in this paper. A literal is either a propositional variable or the negation of a propositional variable. A clause is a disjunctive set of literals. A propositional variable x appears positively (negatively) in a clause C if $x \in C$ (resp. $\neg x \in C$). The notations $\{\ell_1, \dots \ell_n\}$ and $\ell_1 \vee \dots \vee \ell_n$ will be used interchangeably. A clause is tautological if it contains a variable both positively and negatively; it is non-tautological otherwise. We will mainly work with non-tautological clauses, except explicitly stated. Clauses being sets, they can not contain multiple occurrences of the same literal. A formula in conjunctive normal form (CNF for short) is a conjunctive set of clauses. A total (partial) assignment $\mathcal{I}$ for a formula in propositional logic associates a value in $\{\top, \bot\}$ to each (resp. some) propositional variable(s) in the formula. An assignment $\mathcal{I}$ for a formula F is a model of F, denoted $\mathcal{I} \models F$, if it makes the formula F true. A formula is satisfiable if it has a model, it is unsatisfiable otherwise. A total or partial assignment can be perfectly defined by the set of literals it makes true. By default, an assignment is total unless explicitly said to be partial.

We use the usual notions of (un)satisfiability and model also for quantifier-free first-order logic. A set of formulas E entails a (set of) formula(s) E', denoted $E \models E'$, if every model of E is a model of E'.

The notion of assignments, models and the $\models$ relation is extended to quantifier free first order logic via the notion of congruence closure, which is defined below. In order to define congruence closure, we first need to define what literals in this logic are.

[1] We contacted both Ashish Tiwari and the authors of [10] who confirmed this.

Definition 1 (Terms and Subterms)**.** *Let $\mathcal{F}$ be a finite set of function symbols and arity* $: \mathcal{F} \to \mathbb{N}$*. A tuple $\Sigma = \langle \mathcal{F}, arity \rangle$ is a* signature*. A function symbol with arity zero is a* constant*, one with arity one is a* unary *function symbol and one with arity two is* binary*. For a given signature Σ, the set of* terms $\mathcal{T}^\Sigma$ *is defined inductively.*

$$
\begin{aligned}
\mathcal{T}_0^\Sigma &= \{a \in \mathcal{F} \mid arity(a) = 0\} \\
\mathcal{T}_{i+1}^\Sigma &= \{g(t_1, \dots, t_n) \mid arity(g) = n \text{ and } t_1, \dots, t_n \in \mathcal{T}_i\} \\
\mathcal{T}^\Sigma &= \bigcup_{i \in \mathbb{N}} \mathcal{T}_i^\Sigma
\end{aligned}
$$

Let $g(t_1, \dots, t_n) \in \mathcal{T}^\Sigma$, then $t_1, \dots, t_n$ are direct subterms *of $g(t_1, \dots, t_n)$. The* subterm *relation is the reflexive, transitive closure of the direct subterm relation. A term of the form $g(t_1, \dots, t_n)$ is a* compound term.

Let $\mathcal{T}$ be a set of terms. An *equation* of termset $\mathcal{T}$ is a tuple of terms $(s, t) \in \mathcal{T} \times \mathcal{T}$, which we write as $s = t$. For a set of equations E we denote by $\mathcal{T}_E$ the set of terms used in E.

$$\mathcal{T}_E := \{t \mid t \text{ is subterm of some } u\text{, such that for some } v : (u, v) \in E \text{ or } (v, u) \in E\}$$

A quantifier free first order logic literal is a propositional variable, an equation or the negation of any of the first two.

Definition 2 (Congruence Relation)**.** *Let $\mathcal{T}$ be a set of terms. A relation $R \subseteq \mathcal{T} \times \mathcal{T}$ is a congruence relation, if has the following four properties:*

- *reflexivity: for all $t \in \mathcal{T} : (t, t) \in R$*
- *symmetry: $(s, t) \in R$ then $(t, s) \in R$*
- *transitivity: $(r, s) \in R$ and $(s, t) \in R$ then $(r, t) \in R$*
- *compatibility: g is a n-ary function symbol and for all $i = 1, \dots, n$ $(t_i, s_i) \in R$ then $(g(t_1, \dots, t_n), g(s_1, \dots, s_n)) \in R$*

Clearly every congruence relation is also an equivalence relation (which is a reflexive, transitive and symmetric relation). Therefore every congruence relation partitions its underlying set of terms $\mathcal{T}$ into congruence classes, such that two terms (s, t) belong to the same class if and only if $(s, t) \in R$. The relations $\emptyset$ and $\mathcal{T} \times \mathcal{T}$ are trivial congruence relations.

Definition 3 (Congruence Closure)**.** *Let E be a set of equations. The set $E^* \supseteq E$ is the* congruence closure *of E, if E^* is a congruence relation on $\mathcal{T}_E$ and for every congruence relation C, such that $C \supseteq E$ follows $C \supseteq E^*$. We write $E \models s = t$ if $(s, t) \in E^*$ and say that E is an* explanation *for $s = t$.*

It is easily seen that congruence relations are closed under intersection. Therefore E^* always exists.

An assignment $\mathcal{I}$ for a quantifier free first order logic maps terms to some universe $\mathcal{U}$ and literals to $\{\top, \bot\}$. Such an assignment is a model for F, if

- $\mathcal{I}$ is a model of formula F, when every equation is treated as a propositional variable

- For every pairs of terms (s,t) in the congruence closure of $\{s = t \mid \mathcal{I} \models s = t\}$ we have $\mathcal{I}(s) = \mathcal{I}(t)$
- For every equation $s = t$, such that $\mathcal{I} \models s \neq t\}$ we have $\mathcal{I}(s) \neq \mathcal{I}(t)$

The algorithms we consider in the next section take as input a set of literals E. When we consider complexity, not only the cardinality of the set is important, but also the number of terms and subterms as well as the number of their occurrences. Congruence closure algorithms in modern SMT solvers also typically work on Directed Acyclic Graphs (DAGs), not on trees, to represent terms. In that case, what matters is not the number of term and subterm occurrences, but only the number of (sub)terms. However, the input is also typically not a set, but successive calls to an assertion function with a literal as argument. In that case, every repetition of the same literal could matter for complexity. Let us assume however here that the input is a set E, that terms are DAGs with maximal sharing (identity of two terms can be checked in constant time), and that identity of two function symbols can be checked in constant time. Therefore, we characterize complexity results in terms of number of literals, terms and subterms of the input set, i.e. $|E| + |\mathcal{T}(E)|$.

NP-completeness of small conflict set generation problem

Definition 4 (Small conflict set generation problem)**.** *Given an unsatisfiable set E of literals in quantifier-free first-order logic with equality and $k \in \mathbb{N}$, the* small conflict set generation problem *is the problem of finding whether there exists an unsatisfiable set $E' \subseteq E$ with $|E'| \leq k$.*

This problem is NP-complete. Our proof reduces the problem of deciding the satisfiability of a propositional logic formula in conjunctive normal form (SAT) to the small *explanation problem*:

Definition 5 (Small explanation problem)**.** *Given a set of equations $E = \{s_1 = t_1, \dots s_n = t_n\}$, $k \in \mathbb{N}$ and a target equation $s = t$, the* small explanation problem *is the problem of answering whether there exists a set E' such that $E' \subseteq E$, $E' \models s = t$ and $|E'| \leq k$.*

The small explanation problem and the small conflict set generation problem are closely related. There is a small explanation of size k of $s = t$ from E if and only if there is a small conflict of size $k+1$ for $E \cup \{s \neq t\}$. The proof of hardness is based on a (polynomial) translation of the propositional satisfiability problem to the small explanation problem.

In the following text we will use notations like $\top_i$ and $\bot_i$ as constants, with no implicit relation to the Boolean constants $\top$ and $\bot$.

Definition 6 (CNF congruence translation)**.** *Let $\mathcal{C}$ be a set of propositional clauses $\{C_1, \dots C_n\}$ using variables $x_1, \dots, x_m$. The* congruence translation *$E_\mathcal{C}$ of $\mathcal{C}$ is defined as the set of equations*

$$E_\mathcal{C} = \mathit{Connect} \cup \bigcup_{1 \leq i \leq n} \mathit{Clause}_i$$

with

$$\begin{array}{rcl} \mathit{Connect} & = & \{c'_i = c_{i+1} \;\mid\; 1 \leq i < n\} \\ \mathit{Clause}_i & = & \{c_i = t_i(\hat{x}_j) \;\mid\; x_j \textit{ appears in } C_i\} \\ & & \cup\ \{t_i(\top_j) = c'_i \;\mid\; x_j \textit{ appears positively in } C_i\} \\ & & \cup\ \{t_i(\bot_j) = c'_i \;\mid\; x_j \textit{ appears negatively in } C_i\} \end{array}$$

where $c_1, \dots c_n, c'_1, \dots c'_n, \hat{x}_1, \dots \hat{x}_m, \top_1, \dots \top_m, \bot_1, \dots \bot_m$ are distinct constants, and $t_1, \dots t_n$ are distinct unary functions.

This translation is illustrated by the following example.

Example 1. *Consider the set of clauses $\mathcal{C}$*

$$\{C_1 = x_1 \vee x_2 \vee \neg x_3, C_2 = \neg x_2 \vee x_3, C_3 = \neg x_1 \vee \neg x_2\}.$$

Figure 1 represents the congruence translation of $\mathcal{C}$ graphically, an edge between two nodes meaning that the set contains an equation between the terms labeling the two nodes.

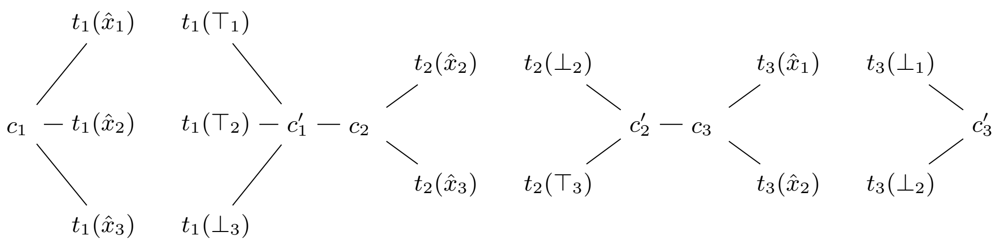

Figure 1: The congruence translation $E_\mathcal{C} = \mathit{Connect} \cup \bigcup_{1 \le i \le n} \mathit{Clause}_i$ of $\mathcal{C}$.

Assignments can also be translated to sets of equations:

Definition 7 (Assignment congruence translation)**.** *Given an assignment $\mathcal{I}$ on propositional variables $x_1, \ldots, x_m$, the set of terms $\mathcal{T}$ is constructed using the following constants and function symbols. The congruence translation $E_\mathcal{I}$ of $\mathcal{I}$ is defined as the set of equations*

$$\begin{aligned} E_\mathcal{I} \quad = \quad & \{\hat{x}_j = \top_j \mid 1 \le j \le m \text{ and } \mathcal{I} \models x_j\} \\ & \cup \{\hat{x}_j = \bot_j \mid 1 \le j \le m \text{ and } \mathcal{I} \models \neg x_j\} \end{aligned}$$

For convenience, we also define the set

$$\mathit{Assignment}^\star = \{\hat{x}_j = \top_j, \hat{x}_j = \bot_j \mid 1 \le j \le m\}.$$

The congruence translation of an assignment is always a subset of $\mathit{Assignment}^\star$. By extension, a subset of $\mathit{Assignment}^\star$ is said to be an assignment if it is the congruence translation of an assignment, that is, if it does not contain both $\hat{x}_j = \top_j$ and $\hat{x}_j = \bot_j$ for some j.

Example 2. *(Example 1 continued) Consider the model $\mathcal{I} = \{x_1, \neg x_2, x_3\}$ of $\mathcal{C}$. Figure 2 gives a graphical representation of $E_\mathcal{I}$, whereas $\mathit{Assignment}^\star$ is described by Figure 3. Notice that $E_\mathcal{C} \cup E_\mathcal{I} \models c_1 = c'_3$, and c_1 and c'_3 are connected in the congruence graph of $E_\mathcal{C} \cup E_\mathcal{I}$ (Figure 4). This is actually the aim of the construction.*

Lemma 1. *Consider a (partial or total) assignment $\mathcal{I}$ for a set of non-tautological clauses $\mathcal{C} = \{C_1, \ldots C_n\}$ using variables $x_1, \ldots, x_m$. $\mathcal{I} \models \mathcal{C}$ if and only if $E_\mathcal{I} \cup E_\mathcal{C} \models c_1 = c'_n$.*

Proof. ($\Leftarrow$) Consider the congruence graph induced by $E_\mathcal{I} \cup E_\mathcal{C}$. Besides edges directly associated to equalities in the set, the only edges are congruence edges between terms $t_i(\hat{x}_j)$ and either $t_i(\top_j)$ or $t_i(\bot_j)$. So any path from c_1 to c'_n would go through such a congruence edge for each i. And such an edge exists for i if and only if the clause i is satisfied by $\mathcal{I}$.

($\Rightarrow$) If $\mathcal{I} \models \mathcal{C}$, then $\mathcal{I} \models C_i$ for each clause $C_i \in \mathcal{C}$. Assume $\mathcal{I}$ makes true a variable x_j, literal of C_i (the case of the negation of a variable is handled similarly). Then $E_\mathcal{I} \models t_i(\hat{x}_j) = t_i(\top_j)$, and $E_\mathcal{I} \cup \mathit{Clause}_i \models c_i = c'_i$. This is true for each i, and thanks to the equations in *Connect*, one can deduce using transitivity that $E_\mathcal{I} \cup E_\mathcal{C} \models c_1 = c'_n$.

□

$$
\begin{array}{ccccc}
 & & \hat{x}_1 & \text{———} & \top_1 \\
\bot_2 & \text{———} & \hat{x}_2 & & \\
 & & \hat{x}_3 & \text{———} & \top_3
\end{array}
$$

Figure 2: Congruence translation of $\mathcal{I}$

$$
\begin{array}{ccccc}
\bot_1 & \text{——} & \hat{x}_1 & \text{——} & \top_1 \\
\bot_2 & \text{——} & \hat{x}_2 & \text{——} & \top_2 \\
\bot_3 & \text{——} & \hat{x}_3 & \text{——} & \top_3
\end{array}
$$

Figure 3: *Assignment**

Lemma 2. *Consider a (partial or total) assignment $\mathcal{I}$ for a set of non-tautological clauses $\mathcal{C} = \{C_1, \dots C_n\}$ using variables $x_1, \dots, x_m$. $|E_\mathcal{I} \cup E_\mathcal{C}|$ and $|\mathcal{T}(E_\mathcal{I} \cup E_\mathcal{C})|$ are polynomial in n and m.*

Proof. $E_\mathcal{I}$ contains at most m equations, since for no j both $\mathcal{I} \models x_j$ and $\mathcal{I} \models \neg x_j$. The set *Connect* contains exactly $n-1$ equations. For every i, the set Clause_i contains at most $2m$ equations, resulting in $2mn$ equations for all clauses. In total, we thus have $|E_\mathcal{I} \cup E_\mathcal{C}| \leq n - 1 + m + 2mn$.

$E_\mathcal{I} \cup E_\mathcal{C}$ contains at most $2n + 3m + 3mn$ terms, which are $2n$ for c_i, c_i', $3m$ for $\hat{x}_j, \top_j, \bot_j$ and $3mn$ for all possible combinations of $t_i(\hat{x}_j), t_i(\top_j), t_i(\bot_j)$. □

Considering again Example 2, and particularly Figure 4, any transitivity chain from c_1 to c_3' will pass through c_1', c_2, c_2' and c_3. Any acyclic path from c_1 to c_3' will contain 11 edges: 3 congruence edges, $3 * 2$ edges in Clause_i for $i = 1, \dots 3$ and 2 edges from *Connect*.

Since every interpretation $\mathcal{I}$ is such that $E_\mathcal{I} \subset \mathit{Assignment}^\star$, one can try to relate the propositional satisfiability problem for a set of clauses $\mathcal{C} = \{C_1, \dots C_n\}$ to finding an explanation of $c_1 = c_n'$ in $\mathit{Assignment}^\star \cup E_\mathcal{C}$. However, it is necessary that this explanation does not set $\hat{x}_j$ equal both to $\top_j$ and $\bot_j$, i.e. at most one of the two equations $\hat{x}_j = \top_j$ and $\hat{x}_j = \bot_j$ should be in the explanation. By restricting assignments to total ones, i.e. by enforcing that at least one of the two equations $\hat{x}_j = \top_j$ and $\hat{x}_j = \bot_j$ belongs to the explanation, it is also possible, with a single cardinality constraint on the explanation, to require that at most one of them belong to the explanation.

Lemma 3. *A set of non-tautological clauses $\mathcal{C} = \{C_1, \dots C_n\}$ using variables $x_1, \dots, x_m$ is satisfiable if and only if there is a subset $E' \subseteq \mathit{Assignment}^\star \cup E_{\mathcal{C}'}$ such that $E' \models c_1 = c_{n+m}'$ and $|E'| \leq 3n + 4m - 1$, where $\mathcal{C}'$ is $\mathcal{C}$ augmented with the tautological clauses $C_{n+i} = x_i \vee \neg x_i$ for $i = 1, \dots m$.*

Proof. ($\Rightarrow$) Consider a total model $\mathcal{I}$ for $\mathcal{C}$: $E_\mathcal{I} \subset \mathit{Assignment}^\star$ contains exactly m elements. For each clause i ($i = 1 \dots n+m$), collect in $E \subset E_{\mathcal{C}'}$ both equations relative to one satisfied literal (that is $2(n+m)$ equations in total), as well as the $n+m-1$ equations from *Connect*. The

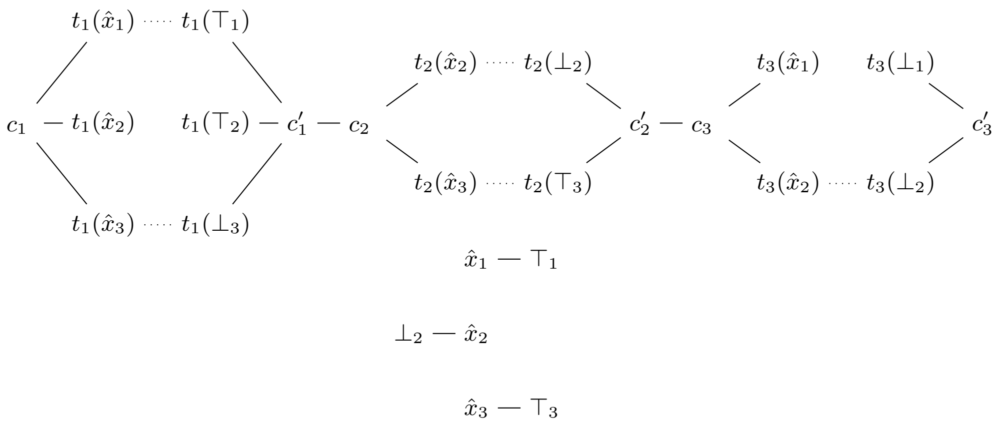

Figure 4: The congruence graph for $E_{\mathcal{C}} \cup E_{\mathcal{I}}$

equation $c_1 = c'_{n+m}$ is a consequence of the disjoint union $E \cup E_{\mathcal{I}}$, which contains $3n + 4m - 1$ elements.

($\Leftarrow$) An explanation of $c_1 = c'_{n+m}$ has to contain $2(n + m)$ equations from $Clause_i$ ($i = 1 \dots n + m$) and $n + m - 1$ equations from *Connect*. Thanks to the tautological clauses, any explanation also has to contain at least $\hat{x}_j = \top_j$ or $\hat{x}_j = \bot_j$ for each $j \in \{1 \dots m\}$. Therefore, the cardinality constraint requires that the explanation contains at most one $\hat{x}_j = \top_j$ or $\hat{x}_j = \bot_j$ for each $j \in \{1 \dots m\}$. If such an explanation exists, Lemma 1 guarantees the existence of a model for $\mathcal{C}'$, or equivalently for the original set of clauses $\mathcal{C}$. □

Corollary 1 (NP-hardness)**.** *The small explanation problem is NP-hard.*

Proof. Propositional satisfiability is NP-hard, and can be reduced in polynomial time to the small explanation problem. □

Lemma 4 (NP)**.** *The small explanation problem is in NP.*

Proof. Let E be a set of equations and $s = t$ be a target equation. A solution to the explanation problem for some $k \in \mathbb{N}$ is a subset $E' \subseteq E$, such that $|E'| \leq k$. Let $n = |\mathcal{T}(E)| + |E|$ and $n' = |\mathcal{T}(E')| + |E'|$. We have $n' \leq n$, since $E' \subseteq E$ and every term in E' appears also in E. Checking whether E' is an explanation of $s = t$ can be done by computing its congruence closure, which is possible in polynomial time in n' [7] and thereby also in n.

□

Theorem 1 (NP-completeness)**.** *The small explanation problem is NP-complete.*

Proof. By corollary 1 and lemma 4. □

Theorem 2 (NP-completeness)**.** *The small conflict set problem is NP-complete.*

Proof. The small conflict set problem is at least as hard as the small explanation problem since the small explanation problem has been showed to be reducible to the small conflict set problem. It is also in NP for exactly the same reason that the small explanation problem is. □

Conclusion

The conflict set generation feature of congruence algorithms is essential for practical SMT solving. Although one could argue that the important property of the generated conflicts is minimality (i.e. no useless literal is in the conflict), it is also interesting to consider producing the smallest conflict. We have shown that the problem of deciding whether a conflict of a given size exists is NP-complete. Therefore, it is generally intractable to obtain the smallest conflict.

In [6, 8, 9], methods to obtain small conflicts, but not necessarily the smallest, are discussed. In practice, it pays off to prioritize speed of the congruence closure algorithm and conflict generation over succinctness of conflicts. However, other applications sensitive to proof size may benefit from other methods prioritizing small conflict size, at a cost of less efficient solving. Thanks to the NP-completeness, one option could be to iteratively encode the small conflict problem into SAT, and use SAT-solvers to find successively smaller conflicts, until the smallest is found. An algorithm based on a variant of the SAT problem (i.e. Max-SAT) may provide a technique to avoid repeated calls to the solver, and find the smallest conflict in one run.

Acknowledgment. We would like to thank Roberto Nieuwenhuis and Ashish Tiwari for discussions and some preliminary ideas that led us to this proof.

References

[1] Clark Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In Armin Biere, Marijn J. H. Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 26, pages 825–885. IOS Press, February 2009.

[2] Joseph Boudou, Andreas Fellner, and Bruno Woltzenlogel Paleo. Skeptik: A proof compression system. In Stéphane Demri, Deepak Kapur, and Christoph Weidenbach, editors, *International Joint Conference on Automated Reasoning (IJCAR)*, volume 8562 of *Lecture Notes in Computer Science*, pages 374–380. Springer, 2014.

[3] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[4] Peter J. Downey, Ravi Sethi, and Robert E. Tarjan. Variations on the common subexpressions problem. *Journal of the ACM*, 27(4):758–771, October 1980.

[5] Pascal Fontaine. *Techniques for verification of concurrent systems with invariants.* PhD thesis, PhD thesis, Institut Montefiore, Université de Liege, Belgium, 2004.

[6] Pascal Fontaine and E. Pascal Gribomont. Using BDDs with combinations of theories. In Matthias Baaz and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, volume 2514 of *Lecture Notes in Computer Science*, pages 190–201. Springer-Verlag, 2002.

[7] Greg Nelson and Derek C. Oppen. Fast decision procedures based on congruence closure. *Journal of the ACM*, 27(2):356–364, April 1980.

[8] Robert Nieuwenhuis and Albert Oliveras. Union-find and congruence closure algorithms that produce proofs. In Cesare Tinelli and Silvio Ranise, editors, *Pragmatics of Decision Procedures in Automated Reasoning (PDPAR)*, 2004.

[9] Robert Nieuwenhuis and Albert Oliveras. Proof-producing congruence closure. In Jürgen Giesl, editor, *Rewriting Techniques and Applications (RTA)*, volume 3467 of *Lecture Notes in Computer Science*, pages 453–468. Springer, 2005.

[10] Robert Nieuwenhuis and Albert Oliveras. Fast congruence closure and extensions. *Information and Computation*, 205(4):557–580, 2007.