Minimally Unsatisfiable Boolean Circuits

Anton Belov and Joao Marques-Silva

Complex and Adaptive Systems Laboratory
School of Computer Science and Informatics
University College Dublin, Ireland
{anton.belov,jpms}@ucd.ie

Abstract. Automated reasoning tasks in many real-world domains involve analysis of redundancies in unsatisfiable instances of SAT. In CNF-based instances, some of the redundancies can be captured by computing a minimally unsatisfiable subset of clauses (MUS). However, the notion of MUS does not apply directly to non-clausal instances of SAT, particularly those that are represented as Boolean circuits. In this paper we identify certain types of redundancies in unsatisfiable Boolean circuits, and propose a number of algorithms to compute minimally unsatisfiable, that is, irredundant, subcircuits.

1 Introduction

Understanding the causes of unsatisfiability of sets of Boolean constraints is a problem of both theoretical and practical interest. Over the last decade, a large number of algorithms for identifying *minimally unsatisfiable subformulas (MUSes)* of CNF formulas have been developed. Recent accounts of practical algorithms can be found in [3, 2, 8], and the current theory of CNF-based MUSes in [6]. However, in many settings the original problem representation is not CNF formulas, but arbitrary Boolean formulas or circuits. For example, in hardware model checking, the next state logic can be represented as a Boolean circuit. Also, for predicate-based abstraction [10], it is necessary to compute MUSes starting from circuit structures. A fairly straightforward observation is that computing an MUS from a clausal representation of a circuit may not result in a circuit. In some contexts this is not a significant issue, but in others it can represent an important drawback. For example, circuit designers are likely to prefer to analyze a Boolean circuit than a set of apparently unrelated clauses. As a result, it is of interest to be able to compute a non-clausal Boolean formula or circuit that represents a minimal source of unsatisfiability. Early examples of work addressing minimal sources of unsatisfiability in non-clausal formulas include [6, 11]. However, these early attempts are only applicable in restricted cases, and so do not provide a general solution.

This paper contains the following main contributions. First, the paper formalizes the notion of *minimally unsatisfiable circuits*. Second, the paper proposes algorithms for the computation of minimally unsatisfiable subcircuits of Boolean circuits (*circuit MUSes*). Third, the paper investigates the relationship between circuit MUSes and the recently proposed notion of group oriented MUSes [10]. Experimental results confirm the practical efficiency of the proposed algorithms, and the usefulness of dedicated techniques.

2 Preliminaries

Propositional formulas are constructed in terms of a countably infinite set of propositional variables, logical constants F and T and a set logical connectives (in this paper, we assume this set to be $\{\neg, \vee, \wedge\}$). We denote the set of all propositional formulas by PROP, and when $\alpha \in \mathrm{PROP}$ the set of propositional variables that occur in α by $Var(\alpha)$. A *truth-value assignment* (or simply, *assignment*) for $\alpha \in \mathrm{PROP}$ is function h mapping $Var(\alpha) \cup \{F, T\}$ into the set $\{0, 1\}$ in such a way that $h(F) = 0$ and $h(T) = 1$. An assignment h is extended naturally to all subformulas of α. A formula α is satisfiable if there is an assignment h such that $h(\alpha) = 1$.

Propositional formulas in which the negation connective applies only to variables are said to be in the *Negation Normal Form (NNF)*. A *literal* is a propositional variable or its negation, a *clause* is a disjunction of literals. A formula is said to be in the *Conjunctive Normal Form (CNF)* if it is a conjunction of clauses. Formulas in CNF are often represented using set notation, and treated as sets of clauses – we use this representation in this paper.

For a formula $\alpha \in \mathrm{PROP}$, the *polarity* of a subformula α' of α is positive (resp. negative) if α' is in the scope of an even (resp. odd) number of negation connectives. We write $pol(\alpha') = 1$ (resp. $pol(\alpha') = -1$) when the polarity of α' is positive (resp. negative). Recall that if $pol(\alpha') = 1$, then for any assignment h we have $h(\alpha(\alpha'/F)) \leq h(\alpha) \leq h(\alpha(\alpha'/T))$ – the inequalities are reversed when $pol(\alpha') = -1$. Here $\alpha(\alpha'/\gamma)$ denotes the formula obtained from α by replacing the subformula α' with the logical constant γ.

Let G be a countably infinite set of *gate variables* (or simply *gates*). A *Boolean circuit* over G is a finite set C of equations of the form $g = f(g_1, \ldots, g_n)$, where $g, g_1, \ldots, g_n \in G$, and $f : \{0, 1\}^n \to \{0, 1\}$ is a Boolean function, with the additional requirements that *(i)* each $g \in G$ appears at most once as the left hand side in the equations in C, and *(ii)* the underlying directed graph $\langle G, E(C)\rangle$, where $E(C) = \{\langle g, g'\rangle \in G \times G \mid g = f(\ldots, g', \ldots) \in C\}$, is acyclic. We refer to the elements of $E(C)$ as *wires*, and to the graph $\langle G, E(C)\rangle$ as the *circuit graph* of C. If the equation $g = f(g_1, \ldots, g_n)$ is in C then g is an f-gate (or, of type f), the equation is denoted by eq_g. When no ambiguity is possible, we write $g \in C$ to denote $eq_g \in C$.

For a gate g, the set of its children (resp. parents) in the circuit graph is called the *fanin* (resp. *fanout*) of g and is denoted by $FI(g)$ (resp. $FO(g)$). A gate with the empty fanin (resp. fanout) is an *input gate* (resp. *output gate*). A gate that is neither an input nor an output is an *internal gate*. The sets of input gates and output gates in C are denoted by $Inputs(C)$ and $Outputs(C)$, respectively.

An *assignment* for a circuit C, is a function $h : Inputs(C) \to \{0, 1\}$ extended in the natural way to all gates in C – that is, for each $g = f(g_1, \ldots, g_n) \in C$, $h(g) = f(h(g_1), \ldots, h(g_n))$. Satisfiability for Boolean circuits can be defined in the following way: for each circuit C fix a designated output gate $out_C \in Outputs(C)$. Then C is *satisfiable* (with respect to out_C) if there exists an assignment h such that $h(out_C) = 1$, otherwise C is *unsatisfiable*. The polarity of gates in C with respect to the designated output out_C can be defined in terms of paths in the circuit graph and the monotonicity of functions that appear on these paths.

3 Definitions of minimal unsatisfiability

We begin by reviewing the well-known definition of minimal unsatisfiability for formulas in CNF, and some of the existing proposals for generalization of minimal unsatisfiability to non-clausal propositional formulas.

Definition 1. *A CNF formula F is* minimally unsatisfiable *if F is unsatisfiable, and for any clause $c \in F$, the formula $F \setminus \{c\}$ is satisfiable.*

As in [6], by MU we denote the set of minimally unsatisfiable formulas in CNF. The set of *minimally unsatisfiable subformulas* of a CNF formula F, in symbols $MUS(F)$ is defined as $MUS(F) = \{F' \mid F' \subseteq F$ and $F' \in MU\}$.

A definition of minimal unsatisfiability for propositional formulas in NNF has been proposed in [6]. Let α be an NNF formula, and T_α be the tree representation α. Consider any subtree $T_{\alpha'}$ of T_α whose root is either an $\vee$-node or a literal, and that is a successor of an $\wedge$-node. Then, formula α' represented by $T_{\alpha'}$ is called an *or-subformula* of α. The minimal unsatisfiability can be defined with respect to the elimination of or-subformulas.

Definition 2 (cf. [6]). *An NNF formula α is* minimally unsatisfiable *if α is unsatisfiable, and for any or-subformula α' of α, the formula produced by the elimination of α' from α is satisfiable.*

We denote the set of minimally unsatisfiable, according to Definition 2, NNF formulas by MU_{NNF} (it is MU^* in [6]). Note that a syntactic elimination of a subformula does not necessarily yield a well-formed formula, as such, the term "elimination" in this definition implies an additional simplification. Nevertheless, given an NNF formula α one can define a set $MUS_{\mathrm{NNF}}(\alpha)$ by analogy with CNF – this set contains all formulas in MU_{NNF} that can be obtained from α via elimination of any number of or-subformulas (including none). It is not difficult to see that on the domain of formulas in CNF, Definition 2 captures the same set of formulas as Definition 1.

A number of notions of minimal unsatisfiability for temporal formulas in LTL have been proposed in [11]. One of these notions, when specialized to the (classical) propositional logic (which is a fragment of LTL) results in the following definition:

Definition 3 (cf. [11]). *A propositional formula α is* minimally unsatisfiable *if α is unsatisfiable, and the replacement of any of its positively (resp. negatively) polarized subformula by the logical constant T (resp. F) produces a satisfiable formula.*

By MU_{PROP} we denote the set of minimally unsatisfiable, according to Definition 3, propositional formulas. Given a propositional formula α, the set $MUS_{\mathrm{PROP}}(\alpha)$ can be defined by analogy with the definition of $MUS_{\mathrm{NNF}}(\alpha)$. Note that Definition 3 captures the same set of formulas in NNF as Definition 2, that is $MU_{\mathrm{NNF}} = MU_{\mathrm{PROP}} \cap \mathrm{NNF}$.

The notions of minimal unsatisfiability presented in this section, and the related notions of MUS, both in clausal and non-clausal domains, rely on some *basic operation* with certain properties. Formally, we can describe such operation by a binary relation on a set of formulas. For example, in the case of CNF the basic operation is the removal of a single clause, and the corresponding relation $\mathcal{R}_{\mathrm{CNF}}$ on the set of CNF formulas is:

$$\langle F, F' \rangle \in \mathcal{R}_{\mathrm{CNF}} \text{ if and only if } \exists c\, (F' = F \setminus \{c\}). \qquad (1)$$

Then, a CNF $F \in MU$ if F is unsatisfiable, and any $F' \in \mathcal{R}_{\mathrm{CNF}}(F)$ is satisfiable. In addition, given a CNF F, the set $MUS(F)$ is defined simply as $\mathcal{R}^{*}_{\mathrm{CNF}}(F) \cap MU$. The relations $\mathcal{R}_{\mathrm{NNF}}$ and $\mathcal{R}_{\mathrm{PROP}}$ on the set of propositional formulas that describe the operations used in the definitions of MU_{NNF} and MU_{PROP} can be defined analogously.

In general, given a set LE of logical entities with a defined notion of satisfiability (for example, logical formulas, or Boolean circuits), and a binary relation $\mathcal{R}$ on LE, we can define the set $MU_{LE}(\mathcal{R})$ of minimally unsatisfiable, with respect to $\mathcal{R}$, members of LE as

$$MU_{LE}(\mathcal{R}) = \{L \in \mathrm{LE} \mid L \text{ is unsatisfiable, and any } L' \in \mathcal{R}(L) \text{ is satisfiable } \},$$

and, given $L \in \mathrm{LE}$, the related set

$$MUS_{LE}(L, \mathcal{R}) = \mathcal{R}^{*}(L) \cap MU_{LE}(\mathcal{R}).$$

However, for some relations $\mathcal{R}$, the sets defined in this way might not capture the intuitive meaning of minimal unsatisfiability – the *irredundancy*. As such, we propose a number of characteristic properties of $\mathcal{R}$ that, albeit somewhat imprecise, aid in constructing intuitively meaningful definitions of minimal unsatisfiability and MUS.

Property 1. $\mathcal{R}$ has to be *satisfiability preserving*, that is if $L \in \mathrm{LE}$ is satisfiable, then any $L' \in \mathcal{R}(L)$ is satisfiable.

This property ensures that minimal unsatisfiability defined using $\mathcal{R}$ captures a strong notion of irredundancy – if L is in $MU_{LE}(\mathcal{R})$, then *every* L' in $\mathcal{R}^{+}(L)$ is satisfiable.

Property 2. For any unsatisfiable $L \in \mathrm{LE}$, $\mathcal{R}(L) \neq \emptyset$.

This property of $\mathcal{R}$ prevents definitions of minimal unsatisfiability that are vacuous – that is, elements L of $MU_{LE}(\mathcal{R})$ that are minimally unsatisfiable simply because the basic operation captured by the relation $\mathcal{R}$ cannot be applied to L.

Property 3. Every $L' \in \mathcal{R}(L)$ is in some sense "smaller" than, and is "close" to L.

This property can be made precise by defining a suitable order and a metric on LE, however for this paper we will rely on its intuitive meaning. Note that in general $L' \in \mathcal{R}(L)$ is not a necessarily a "sub-object" of L – in fact, among the definitions presented above, it is only in the case of CNF, where L' is a sub-formula of L.

In the next section we propose two relations on the set of Boolean circuits that satisfy the above requirements, and give rise to intuitively meaningful definitions of minimally unsatisfiable Boolean circuits and circuit MUSes.

4 Minimally Unsatisfiable Boolean Circuits

Consider a Boolean circuit C over a set of gates G – recall that C is a finite set of equations eq_g of the form $g = f(g_1, \ldots, g_n)$. Let out_C be the designated output of C, and assume that C is unsatisfiable, that is, for every assignment h for C, $h(out_C) = 0$. Consider the situation when there exists a gate $g \in C$ such that $C \setminus \{g\}$ is unsatisfiable. Then the gate g, or more precisely the equation eq_g, is redundant with respect to the

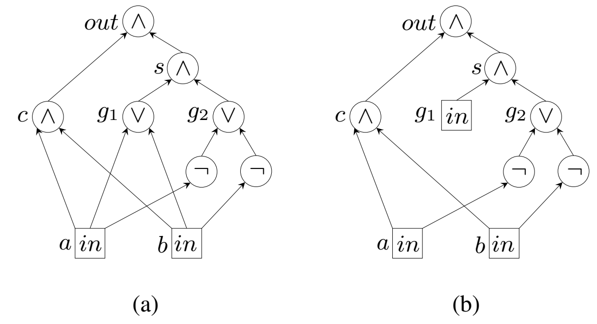


Fig. 1. (a) An example circuit C (a half-adder with both the carry c and the sum s set to 1). C is unsatisfiable. (b) The circuit $C' = C \setminus \{g_1\}$ is also unsatisfiable, and since $C' \in \mathcal{R}_g(C)$, the circuit C is not gate-minimally unsatisfiable. However, the circuit C' is, and is a gate-MUS of C.

unsatisfiability of C. This suggests a possible *gate-based* definition of minimal unsatisfiability for circuits: a circuit is unsatisfiable, and no gate equation can be removed without making it satisfiable. We will formalize this definition shortly, but first present a slightly different perspective.

Each gate equation $g = f(g_1, \ldots, g_k)$ in the circuit captures a relationship, a constraint, between the values of g and the values of gates in $FI(g)$. When the gate is redundant, the relationship of g with all these values is redundant. However, when the gate is not redundant, it does not necessarily mean that the relationship of g with *all* gates is not redundant. It is possible that only some of these relationships are important for unsatisfiability, while others can be dropped. These individual relationships corresponds to the wires in the circuit – i.e. the edges in the circuit graph – that connect the gate to the gates in its fanin. This suggests a different, *wire-based*, definition of minimal unsatisfiability for circuits – it is more refined than the gate-based, in that a minimally unsatisfiable circuit from the gate view, is not necessarily minimally unsatisfiable from the wire point of view.

With this motivation, we proceed to formalizing the two proposed types of minimal unsatisfiability.

4.1 Gate-based minimal unsatisfiability

Let CIRC be a set of Boolean circuits over the set of gates G, and let $\mathcal{R}_g \subseteq \text{CIRC}^2$ be defined as follows:

$$\langle C, C' \rangle \in \mathcal{R}_g \text{ if and only if } out_C = out_{C'} \text{ and } \exists g \in C \;\; C' = C \setminus \{g\}.$$

When $C' = C \setminus \{g\}$ we have $Inputs(C') = Inputs(C) \cup \{g\}$, thus the effect of the removal of eq_g from C is that g becomes an unconstrained input. As an example, consider the circuit C in Figure 1(a), and the circuit C' obtained from C by removing gate g_2 (Figure 1(b)). We now establish the basic properties of the relation $\mathcal{R}_g$, that, as argued in Section 3, will afford a meaningful definition of minimally unsatisfiable circuit and circuit MUS based on $\mathcal{R}_g$.

Proposition 1. *Let C be a satisfiable Boolean circuit. Then, for any $C' \in \mathcal{R}_g(C)$, C' is satisfiable.*

Proof. Let h be satisfying assignment for C, and let $C' = C \setminus \{g\}$. Then, $h' = h \cup \{\langle g, h(g)\rangle\}$ is a satisfying assignment for C'. $\square$

Hence, $\mathcal{R}_g$ is satisfiability preserving (Property 1). Further, we have that for every unsatisfiable circuit C, $\mathcal{R}_g(C) \neq \emptyset$, because C must have at least one gate (Property 2). Finally, when $C' \in \mathcal{R}_g(C)$, $C' \subset C$, as such C' is smaller than C in terms of the number of gate definitions, and is close to C as the two circuits differ by exactly one gate (Property 3). Thus, paraphrasing the definitions of $MU_{\mathrm{CIRC}}(\mathcal{R}_g)$ and $MUS_{\mathrm{CIRC}}(C, \mathcal{R}_g)$ we have:

Definition 4. *A Boolean circuit C is* gate-minimally unsatisfiable, *if C is unsatisfiable and for every gate $g \in C$, the circuit $C \setminus \{g\}$ is satisfiable.*

Definition 5. *Let C be an unsatisfiable Boolean circuit. Then the circuit C' is a* gate-MUS *of C if $C' \subseteq C$ and C' is gate-minimally unsatisfiable.*

We denote the set $MU_{\mathrm{CIRC}}(\mathcal{R}_g)$ as MU_g, and a set $MUS_{\mathrm{CIRC}}(C, \mathcal{R}_g)$ as $MUS_g(C)$ for the rest of this paper. The circuit C' in Figure 1(b) is gate minimally unsatisfiable, and is the gate-MUS of the circuit C in Figure 1(a).

In Section 3 we have emphasized the fact that the presented definitions of minimal unsatisfiability are strict generalizations: on the domain of CNF formulas, the set MU_{NNF} coincides with the set MU, while on the domain of NNF formulas, the set MU_{PROP} coincides with the set MU_{NNF}. We now demonstrate that the proposed definition of gate-based minimal unsatisfiability for Boolean circuits, despite its intuitive appeal, is in a sense too *coarse*.

Take any $\alpha \in \mathrm{PROP}$, and let $sm : Var(\alpha) \mapsto G$ be some injective function ("subformula map"). We are going to extend sm to all subformulas of α, and, simultaneously, associate each subformula α' of α with a Boolean circuit $C_{\alpha'}$. The construction is defined inductively on the structure of α as follows:

(i) if $\alpha = p$, then let $C_\alpha = \emptyset$, and $out_C = sm(p)$;
(ii) if $\alpha = \beta \wedge \gamma$, then take a fresh $g_\alpha \in G$, and let

$$C_\alpha = C_\beta \cup C_\gamma \cup \{g_\alpha = \wedge(out_{C_\beta}, out_{C_\gamma})\},$$

and let $out_{C_\alpha} = g_\alpha$, and $sm = sm \cup \{\langle \alpha, g_\alpha\rangle\}$.
(iii) the constructions for the cases $\alpha = \beta \vee \gamma$ and $\alpha = \neg\beta$ are analogous to (ii).

Thus, the circuit graph of C_α is tree-like, with the possible exception of the inputs. Furthermore, for every gate $g \in C_\alpha$, the polarity $pol(g)$ is the same as $pol(sm^{-1}(g))$, and for every *non-variable* subformula α' of α, $pol(\alpha') = pol(sm(\alpha'))$.

Let h be an assignment to $Var(\alpha)$, then we can define define a corresponding assignment for C_α in a straightforward manner:

$$h_{sm} = \{\langle\, sm(p), h(sm(p))\,\rangle \mid p \in Var(\alpha)\}.$$

Clearly, for any subformula α' of α, $h(\alpha') = h_{sm}(sm(\alpha'))$. Similarly, given an assignment h for circuit C_α we can define the corresponding assignment $h_{sm^{-1}}$ with the analogous property.

Theorem 1. *For every propositional formula α, if $\alpha \in MU_{PROP}$, then $C_\alpha \in MU_g$, however the converse doesn't hold.*

Proof. It is easy to see that formula α is unsatisfiable if and only if so is the circuit C_α.

Towards a contradiction, assume that $\alpha \in MU_{\mathrm{PROP}}$, but $C_\alpha \notin MU_g$. Since α is unsatisfiable, so is C_α and we conclude that there exists $g \in C_\alpha$ such that the circuit $C' = C_\alpha \setminus \{g\}$ is unsatisfiable. Let $\alpha' = sm^{-1}(g)$, and assume $pol(\alpha') = 1$. Then the formula $\alpha(\alpha'/T)$ must be unsatisfiable, as otherwise, if h is a satisfying assignment for $\alpha(\alpha'/T)$, then $h_{sm} \cup \{\langle g, 1\rangle\}$ satisfies C'. Since $\alpha(\alpha'/T) \in \mathcal{R}_{\mathrm{PROP}}$ we have $\alpha \notin MU_{\mathrm{PROP}}$.

One of the reasons that the reverse implication does not hold is that the operation $\mathcal{R}_{\mathrm{PROP}}$ allows to substitute the constants T/F for variables in the formula, while an equivalent of such operation is not captured by $\mathcal{R}_g$. Consider for example the formula $\alpha = q \wedge (\neg q \wedge r)$. Then, $\alpha \notin MU_{\mathrm{PROP}}$ because the formula $\alpha' = q \wedge (\neg q \wedge T)$ is still unsatisfiable. However, the corresponding circuit $C_\alpha = \{out = \wedge(q, g_1), g_1 = \wedge(g_2, r), g_2 = \neg(q)\}$ is gate-minimally unsatisfiable. □

Note that the formula α used as a counterexample in the above proof is a propositional representation of the CNF formula $F = \{q, \neg q, r\}$ and so $\mathcal{R}_g$ gives rise to the definition of minimal unsatisfiability that is too coarse on domain of CNF formulas as well.

The counterexample in the proof of Theorem 1 might suggest that $\mathcal{R}_g$ could be refined if it were to allow the replacement of inputs by constants according to their polarity. Unfortunately, this suggestion poses an immediate problem: in unsatisfiable circuits there must be non-polarized inputs. Selecting an arbitrary constant for non-polarized inputs results in satisfiability non-preserving operation. Further, the following, intuitively minimally unsatisfiable example circuit $C = \{out = \wedge(p, g_1), g_1 = \neg(p)\}$ would not be minimally unsatisfiable, as we could replace p with 0 (or 1) and still obtain an unsatisfiable circuit.

The real reason for the fact that in certain cases unsatisfiable formulas that are not in MU_{PROP} map to gate-minimally unsatisfiable circuits (i.e. MU_g is in this sense coarser than MU_{PROP}) is that $\mathcal{R}_{\mathrm{PROP}}$ allows the replacement of an *individual occurrence* of a variable in the formula without affecting other occurrences of this variable. Multiple occurrences of a variable p in formula α are represented by multiple wires connecting the input $sm(p)$ in the circuit C_α, hence an operation that would allow to "break" wires in the circuit would address this weakness of $\mathcal{R}_g$. Thus, in conjunction with the discussion at the beginning of this section, we have a strong motivation for the definition of wire-based minimal unsatisfiability of Boolean circuits.

4.2 Wire-minimal unsatisfiable circuits

Let I be a set of gates disjoint from G, let CIRC be a set of Boolean circuits over $G \cup I$, and let $\mathcal{R}_w \subseteq \mathrm{CIRC}^2$ be defined as follows:

$$\langle C, C'\rangle \in \mathcal{R}_w \text{ if and only if } out_C = out_{C'} \text{ and } \exists g, g_k \in G,\ \exists i \in I \text{ such that}$$
$$g = f(\ldots, g_k, \ldots) \in C \text{ and } C' = C \setminus \{g\} \cup \{g = f(\ldots, i, \ldots)\}$$

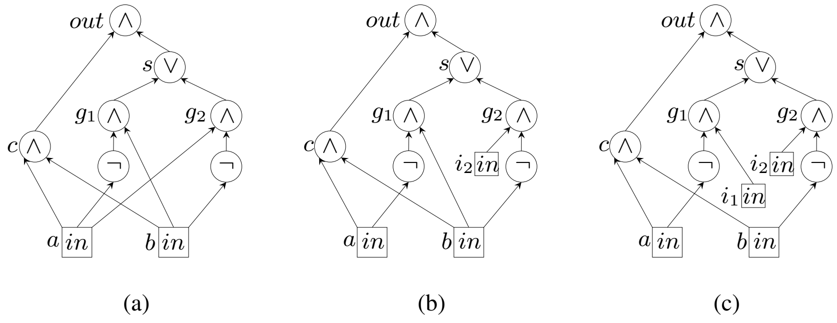


Fig. 2. (a) An example circuit C (also a half-adder with both the carry c and the sum s set to 1). C is unsatisfiable. (b) The circuit C_1 obtained from C by removing the wire $\langle g_2, a\rangle$, hence $C_1 \in \mathcal{R}_w(C)$; C_1 is also unsatisfiable. (c) The circuit C_2 obtained from C_1 by removing the wire $\langle g_1, b\rangle$, hence $C_2 \in \mathcal{R}_w(C)$. $C_2 \in MU_w$, and is a wire-MUS of C (and C_1)

In words, when $C' \in \mathcal{R}_w(C)$, the circuit C' can be obtained by replacing some wire $\langle g, g_k\rangle$ in the circuit graph of C with a wire $\langle g, i\rangle$, where i is a *fresh input gate*. Thus, effectively the operation eliminates the connection, or constraint, between the values of g_k and g in C. Note that only that wires that connect gates in G can be replaced, as such, once replaced, a wire cannot be replaced again. As such we will often refer to this operation as the *removal* of the wire $\langle g, g_k\rangle$ from C.

As an example, consider the circuit C depicted in Figure 2(a), and the circuit C_1 in Figure 2(b) that is obtained from C by removing the wire $\{g_2, a\}$. Thus, $C_1 \in \mathcal{R}_w(C)$. The circuit C_2 in Figure 2(c) is obtained from C_1 by removing the wire $\langle g_1, b\rangle$, as such $C_2 \in \mathcal{R}_w(C_1)$. We use this opportunity to point out that both C_1 and C_2 are unsatisfiable, but C is gate minimally unsatisfiable. In order to further motivate the definition of minimal unsatisfiability based on the relation $\mathcal{R}_w$, we establish the basic properties of this relation outlined in Section 3.

Proposition 2. *Let C be a satisfiable Boolean circuit. Then, for any $C' \in \mathcal{R}_w(C)$, C' is satisfiable.*

Proof. Let h be a satisfying assignment for C, and assume that C' was obtained from C by replacing some wire $\langle g, g_k\rangle$ with $\langle g, i\rangle$, where i is a fresh input gate. Then, the assignment $h' = h \cup \{\langle i, h(g_k)\rangle\}$ is satisfying for C'. □

Thus, $\mathcal{R}_w$ is satisfiability preserving (Property 1). Further, every unsatisfiable circuit, with the exception of $C_d = \{out = 0\}$, has at least one wire, as such for every unsatisfiable $C \neq C_d$, $\mathcal{R}_w(C) \neq \emptyset$. Hence, Property 2 *almost* holds – the rather degenerate circuit C_d is the only case that violates this property, note, however, that $C_d \in MU_{\mathrm{CIRC}}(\mathcal{R}_w)$, albeit vacuously. Finally, with respect to Property 3, when $\langle C, C'\rangle \in \mathcal{R}_w$, C' is not a subcircuit of C (but neither are the formulas related by $\mathcal{R}_{\mathrm{NNF}}$ or $\mathcal{R}_{\mathrm{PROP}}$). It is however, smaller than C in the sense that it has one less constraint between the values of gates. Thus, paraphrasing the definitions of $MU_{\mathrm{CIRC}}(\mathcal{R}_w)$ and $MUS_{\mathrm{CIRC}}(C, \mathcal{R}_w)$ we have:

Definition 6. *A Boolean circuit C is* wire-minimally unsatisfiable *if C is unsatisfiable and for any wire $\langle g, g_k \rangle$ in C, the circuit C' obtained by the replacement of this wire with $\langle g, i \rangle$ for a fresh input i is satisfiable.*

Definition 7. *Let C be an unsatisfiable Boolean circuit. Then the circuit C' is a* wire-MUS *of C, if C' can be obtained from C by removing zero or more wires, and C' is wire-minimally unsatisfiable.*

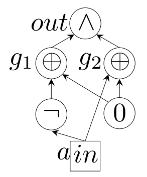
We denote the sets $MU_{\mathrm{CIRC}}(\mathcal{R}_w)$ and $MUS_{\mathrm{CIRC}}(C, \mathcal{R}_w)$ as MU_w and $MUS_w(C)$, respectively. The circuit C_2 depicted in Figure 2(c) is wire-minimally unsatisfiable, and is a wire-MUS of both the circuits C and C_1 in Figures 2(a),2(b).

The example in Figure 2 demonstrates that there are gate-minimally unsatisfiable circuits that are not wire-minimally unsatisfiable. The following theorem shows that, with the exception of circuits with constant gates, every wire-minimally unsatisfiable circuit is gate-minimally unsatisfiable. Let $\mathrm{CIRC}_{\mathrm{nc}} \subset \mathrm{CIRC}$ be the set of Boolean circuits without constant gates. Then,

Theorem 2. *For every Boolean circuit $C \in \mathrm{CIRC}_{\mathrm{nc}}$, if $C \in MU_w$, then $C \in MU_g$, however the converse does not hold.*

Proof. We prove the contrapositive: assume that $C \in \mathrm{CIRC}_{\mathrm{nc}}$ is unsatisfiable and $C \notin MU_g$, we show that $C \notin MU_w$. By assumption $\exists g \in C$, such that $C_g = C \setminus \{g\}$ is unsatisfiable. Let $g = f(g_1, \ldots, g_k)$, let $\{i_1, \ldots, i_k\}$ be a set of fresh input gates, and consider the circuit $C_w = C \setminus \{g\} \cup \{g = f(i_1, \ldots, i_k)\}$. Then, C_w must be unsatisfiable, as otherwise a satisfying assignment h for C_w can be used to construct a satisfying assignment $h \cup \{\langle g, f(h(i_1), \ldots, h(i_k))\rangle\}$ for C_g. Note that $C_w \in \mathcal{R}_w^k(C)$, and therefore for some $C' \in \mathcal{R}_w(C)$, C' is unsatisfiable because $\mathcal{R}_w$ is satisfiability preserving. We conclude that $C \notin MU_w$.

The fact that the converse does not hold is demonstrated in Figure 2. □



The circuit depicted on the right margin illustrates the issue with the constant gates. While the constant gate 0 can be removed from this circuit (recall that this is equivalent to replacing it by an unconstrained input) without breaking its unsatisfiability, removing any of the wires leading from this gate will make the circuit satisfiable. Thus, by allowing the removal of wires, we get an *almost* refinement of MU_g. The following theorem shows that we also gain the equivalence with MU_{PROP}, and as such, with MU_{NNF} and MU (the sets of minimally unsatisfiable formulas in NNF and CNF, respectively).

Theorem 3. *For every propositional formula α, $\alpha \in MU_{\mathrm{PROP}}$, if and only if $C_\alpha \in MU_w$.*

Proof. We prove the contrapositives. Recall that C_α is a circuit constructed using the structure of α (Section 4.1).

Assume α is unsatisfiable and $\alpha \notin MU_{\mathrm{PROP}}$, we show that $\alpha \notin MU_w$. Without the loss of generality, let α' be a positively polarized subformula of α such that the formula $\alpha(\alpha'/T)$ is unsatisfiable. Since $pol(\alpha') = 1$, the formula $\alpha(\alpha'/F)$ is also unsatisfiable.

Note that α' must be a proper subformula of α because the formula T, which is the result of substitution of α itself by T, is obviously satisfiable. As such, let β be the parent of α' in the formula tree of α. Then the circuit C' obtained from C by removing the wire $\langle sm(\beta), sm(\alpha')\rangle$ is also unsatisfiable. Hence, $C \notin MU_w$.

Assume now that C_α is unsatisfiable and $C_\alpha \notin MU_w$, we show that $\alpha \notin MU_{\mathrm{PROP}}$. Let $\langle g_1, g_2\rangle$ be the wire in C_α that can be removed to obtain an unsatisfiable circuit C'. If g_2 is not an input gate, then let $\alpha_2 = sm^{-1}(g_2)$, otherwise let α_2 be the occurrence of the variable $sm^{-1}(g_2)$ in the subformula $sm^{-1}(g_1)$. Then, both $\alpha(\alpha_2/T)$ and $\alpha(\alpha_2/F)$ must be unsatisfiable. Therefore, $\alpha \notin MU_{\mathrm{PROP}}$. $\square$

5 Computing Circuit MUSes

In this section we propose possible solutions to the problem of computing a gate-MUS or a wire-MUS of a given unsatisfiable circuit C. For reasons of clarity, in this section we assume that C does not have constant gates, that is $C \in \mathrm{CIRC}_{\mathrm{nc}}$.

Most of the high-performing algorithms for computation of CNF-based MUSes are based on the identification of so called *transition clauses* [3] in unsatisfiable CNF formulas. A clause $c \in F$ is called a transition clause, if F is unsatisfiable but $F \setminus \{c\}$ is satisfiable. The key property of the transition clauses is that if c is a transition clause for F, then c belongs to *all* MUSes of F. Then, given an unsatisfiable formula F, a *deletion-based* MUS extractor picks a clause $c \in F$, and tests the formula $F' = F \setminus \{c\}$ for satisfiability. If F is satisfiable, then c is *final* – it is a part of constructed MUS. Otherwise, the algorithm continues with the formula F'. When all clauses are final, the current formula is an MUS of the initial formula F. In most cases, this basic extraction algorithm can be accelerated significantly when the underlying SAT solver supports incremental SAT solving, and is capable of producing proofs of unsatisfiability.

It is not difficult to see that in the case of Boolean circuits, the analogous concepts – *transition gates*, and *transition wires* – can be defined, and possess similar properties. As such, the existing CNF MUS algorithms, such as the deletion-based algorithm described above, can be adapted to the circuit MUS problem. It is plausible, however, that in the case of circuits the structure can be used to accelerate the circuit MUS computation – we present empirical data to support this claim in Section 6.

Unfortunately, the publicly available efficient circuit SAT solvers, such as [4], neither expose an incremental interface, nor produce proofs of unsatisfiability. Thus, it is advantageous to develop CNF-based techniques for circuit MUS extraction in order to capitalize on the continuing progress of CNF-based SAT technology.

Let C be a Boolean circuit. For each $g \in C$, let $Ts(g)$ be the set of clauses obtained by the Tseitin transformation of g to CNF [12], and thus, $Ts(C) = \{out_C\} \cup \bigcup_{g \in C} Ts(g)$ be the Tseitin encoding of C. It is tempting to compute CNF-based $M = MUS(Ts(s))$, and then "inflate" each clause in M to obtain the circuit

$$C_M = \bigcup_{c \in M} \{g \mid c \in Ts(g)\}.$$

In general, the resulting circuit C_M is not a gate-MUS of C, and so CNF MUS extractors are not applicable to circuit MUS problem. However, circuit MUSes can be computed using the tools developed for the recently proposed problem of *group oriented MUS* extraction [10]:

Definition 8 ([5]). *Given an explicitly partitioned unsatisfiable CNF formula $F = D \cup \bigcup_{G \in \mathcal{G}} G$, where $\mathcal{G} = \{G_1, \ldots, G_k\}$, and D and each G_i are disjoint sets of clauses, a* group oriented MUS *of F is a subset $\mathcal{G}'$ of $\mathcal{G}$ such that $D \cup \bigcup_{G \in \mathcal{G}'} G$ is unsatisfiable, and, for every $\mathcal{G}'' \subset \mathcal{G}'$, we have that $D \cup \bigcup_{G \in \mathcal{G}''} G$ is satisfiable.*

It is not difficult to see that if we let $\mathcal{G}_C = \{Ts(g) \mid g \in C\}$, and let $D = \{out_C\}$, then the group oriented MUS of the formula $F_C = D \cup \bigcup_{G \in \mathcal{G}_C} G$ corresponds to a gate-MUS of the circuit C, and vice-versa. Thus, gate-MUSes for circuits can be computed using group oriented MUS extractors, for example `SAT4J` [7].

The problem of wire-MUS computation for a given circuit C, however, cannot be solved directly by computation of group oriented MUS of the formula F_C. Consider for example a gate $g \in C$ defined as $g = \wedge(g_1, g_2, g_3)$. The set $Ts(g)$ contains four clauses c_m, c_1, c_2, c_3, where

$$c_m = g \vee \neg g_1 \vee \neg g_2 \vee \neg g_3,\ c_1 = \neg g \vee g_1,\ c_2 = \neg g \vee g_2,\ c_3 = \neg g \vee g_3.$$

Assume now that we remove the wire $\langle g, g_1 \rangle$ from C, that is we replace $\langle g, g_1 \rangle$ with $\langle g, i_1 \rangle$, where i_1 is a fresh input, to obtain a circuit C'. Then, in C' the set $Ts(g)$ contains the clauses c'_m, c'_1, c_2, c_3, where

$$c'_m = g \vee \neg i_1 \vee \neg g_2 \vee \neg g_3,\ c'_1 = \neg g \vee i_1.$$

Since the variable i_1 does not appear in any other clause of $Ts(C')$ except c'_m and c'_1, from the perspective of the satisfiability the net effect of removing the wire $\langle g, g_1 \rangle$ from C on $Ts(C)$ is simply the *removal of clauses* c_m and c_1 from $Ts(C)$. Formally, the CNF formula $Ts(C')$ is satisfiable if and only if the formula $F = Ts(C) \setminus \{c_m, c_1\}$ is satisfiable. The "only if" direction is obvious, since $F \subset Ts(C')$. For the "if" direction, let h be satisfying for F – if $h(g) = 1$, then c'_m is satisfied in $Ts(C')$, and we assign 1 to i_1 to satisfy c'_1; if $h(g) = 0$, then c'_1 is satisfied, and we assign 0 to i_1 to satisfy c'_m. Similarly, the effect of removing the wire $\langle g, g_2 \rangle$ from C' amounts to removing the clause c_2 from $Ts(C')$. Finally, the subsequent removal of $\langle g, g_3 \rangle$ results in the formula with all clauses c_m, c_1, c_2, c_3 removed – note that this is equivalent to removing all clauses in $Ts(g)$ from $Ts(C)$, and as such, removing the gate g from C. Indeed, as the proof of Theorem 2 shows, removing all wires in the fanin of $g \in C$ is equivalent to removing the gate g itself.

We now point out that the problem of computing a wire-MUS of C could have been mapped to group oriented MUS directly, if in Definition 8 the groups G_i were not required to be disjoint. If this were the case, then we could set $G_1 = \{c_m, c_1\}$, $G_2 = \{c_m, c_2\}$, and $G_3 = \{c_m, c_3\}$ – note that the groups intersect on c_m. Then, removing the group G_1 from the formula F, would result in the removal the clause c_m from G_2 and G_3. As an aside, this suggests a possible generalization of the definition of the group oriented MUS problem. Meanwhile, we can still map wire-MUS problem to the group oriented MUS problem, though at the cost of adding extra variables.

We demonstrate the mapping using the previous example of $g = \wedge(g_1, g_2, g_3)$ with $Ts(g) = \{c_m, c_1, c_2, c_3\}$.

The idea is to add one extra variable for every intersecting group. Let l_1, l_2 and l_3 be three fresh variables, and let G_1^*, G_2^* and G_3^* be the groups of clauses defined in the following way:

$$\begin{aligned} G_1^* &= \{c_m \vee \neg l_2 \vee \neg l_3,\ c_1 \vee \neg l_1,\ l_1\} \\ G_2^* &= \{c_m \vee \neg l_1 \vee \neg l_3,\ c_2 \vee \neg l_2,\ l_2\} \\ G_3^* &= \{c_m \vee \neg l_1 \vee \neg l_2,\ c_1 \vee \neg l_3,\ l_3\} \end{aligned}$$

Then, for example, the removal of group G_1^* makes the variable l_1 unconstrained, and, as such, the first clause in both groups G_2^* and G_3^* effectively becomes satisfied. It is easy to see that this has the exact effect of the removal of the group G_1 under the generalized definition of group MUS that allows non-disjoint groups. The demonstrated technique for mapping non-disjoint group MUS problem to (disjoint) group MUS problem can be applied in a general setting. We omit the formal definition of such mapping, and the proofs of its correctness from this paper.

Intuitively, it seems plausible that the structure of a given instance of group oriented MUS problem can provide additional information that allows to accelerate group-MUS extraction. In the application of group-MUS to circuit-MUS computation problem, the circuit structure can be used to deduce the relationships between groups which, in turn, can be used to guide a group-MUS extractor. We propose two such techniques, and, in the following section, demonstrate empirical evidence to their effectiveness.

One of the techniques is based on the following observation. Let g be a gate in C. By $D(g)$ let us denote the set of gates dominated by g, that is

$$D(g) = \{g' \in C \mid \text{ every path from } g' \text{ to } out_C \text{ in the graph of } C \text{ includes } g\ \}.$$

Note that $g \in D(g)$. Then the circuit $C' = C \setminus \{g\}$ is satisfiable if and only if the circuit $C'' = C \setminus \{D(g)\}$ is satisfiable. As such, during the gate-MUS extraction, rather than testing the circuit C' for satisfiability, we can test the circuit C''. Since C'' is smaller than C' the SAT test might be faster. In addition, if C'' is unsatisfiable, we remove a potentially large set of gates at once, thus reducing the number of SAT checks. This, *domination based* optimization can be improved further by the analysis of the satisfying assignment for C'' in case it is satisfiable.

6 Empirical Study

To evaluate some of the ideas presented in this paper empirically, we implemented a prototype circuit MUS extractor `ncmuser`. The extractor computes gate-MUSes by mapping the gate MUS problem to group oriented MUS in the manner described in the previous section. `ncmuser` interfaces with the group-MUS extractor (the group-oriented version of `MUSer` [9]) by controlling the order in which the latter selects the groups for removal. The mapping of the wire-MUS computation problem to the group oriented MUS extraction problem described in the previous section is currently not used

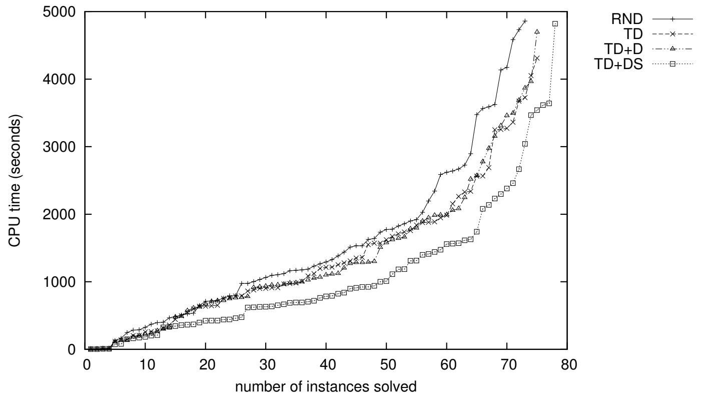


Fig. 3. Effects of gate selection strategies on gate-MUS computation times.

– instead, for wire-MUS extraction `ncmuser` interfaces directly with an incremental SAT solver (`picosat` version 935 [1]). The benchmark circuits for our experiments were selected from the following sets:

(i) unsatisfiable (i.e. correct) sequential designs from the Hardware Model Checking Competition 2010 (http://fmv.jku.at/hwmcc10/) – combinational And-Inverter-Graph (AIG) circuits were generated using `aigtobmc` (http://fmv.jku.at/aiger);
(ii) AIGs generated using Boolector (http://fmv.jku.at/boolector/) to bit-blast `QF_BV` (theory of bit-vectors) instances of the SMT Competition 2009; (http://www.smtcomp.org/2009/)
(iii) unsatisfiable circuits in ISCAS format from the `fvp-unsat-1.0` and `fvp-unsat-2.0` benchmark suites of M. Velev (http://www.miroslav-velev.com/sat_benchmarks.html).

The objective of the first part of our empirical study was to investigate the effectiveness of the structure-based techniques for gate-MUS extraction described in Section 5. We implemented four gate selection strategies in `ncmuser`: the random selection (RND), the top-down traversal of the circuit (i.e. reverse topological order, TD), the top-down traversal with the domination based optimization (TD+D), and, finally, the strategy TD-D with the addition of the analysis of satisfying assignments (TD+DS). From our set of benchmarks we selected a subset of 245 instances solvable with top-down (TD) strategy given 5000 seconds of CPU time and 4 GB of RAM on HPC cluster nodes consisting of two quad-core Intel Xeon E5450's with 32 GB of RAM. From this subset we selected 75 instances that were found to have between 10% to 90% of redundant gates, and added 25 randomly selected timed-out instances. The results of the comparative evaluation of the four gate-selection strategies are presented in Figure 3. We note that the performance of gate-MUS extraction clearly improves with the amount of the circuit-based structural information used to aid the computation.

The goal of the second part of our empirical study was to find out whether the redundant wires do occur in practice. During the computation of wire-MUSes `ncmuser` uses the top-down circuit traversal strategy. As wire-MUS extraction may require more SAT calls than gate-MUS extraction, in wire-MUS extraction mode `ncmuser` was able to solve 228 instances out of 245 described above. We found that out of these 228 instances

30 had over 50%, and 70 had over 10% of redundant wires *after* all the redundant gates have been removed.

7 Conclusion

This paper addresses the problem of minimal unsatisfiability in Boolean circuits. The paper starts by formalizing the gate-based and wire-based notions of *minimally unsatisfiable circuits*, and then proposes algorithms for the computation of gate-MUSes and wire-MUSes of Boolean circuits. One key aspect is the tight relationship between circuit and group-oriented MUS extraction [10, 5]. This applies both to gate-based and wire-based minimal unsatisfiability. Another key aspect is that the extraction can be accelerated by exploiting circuit structure. Experimental results, obtained on Boolean circuits from different application domains, confirm the practical efficiency of the proposed algorithms, and the usefulness of dedicated techniques. Finally, the general treatment of minimal unsatisfiability used in this paper appears to be quite convenient. Future work will investigate further the relationship between circuit and group-oriented MUS extraction.

Acknowledgements We thank the anonymous referees for helpful comments. This work is partially supported by SFI PI grant BEACON (09/IN.1/I2618).

References

1. A. Biere. Picosat essentials. *Journal on Satisfiability, Boolean Modeling and Computation*, 4:75–97, 2008.
2. C. Desrosiers, P. Galinier, A. Hertz, and S. Paroz. Using heuristics to find minimal unsatisfiable subformulas in satisfiability problems. *J. Comb. Optim.*, 18(2):124–150, 2009.
3. É. Grégoire, B. Mazure, and C. Piette. On approaches to explaining infeasibility of sets of Boolean clauses. In *Int'l. Conf. on Tools with Artificial Intelligence*, pages 74–83, 2008.
4. H. Jain and E. M. Clarke. Efficient SAT solving for non-clausal formulas using DPLL, graphs, and watched cuts. In *Proc. of the 46th Annual Design Automation Conference*, pages 563–568, 2009.
5. M. Järvisalo, D. Le Berre, and O. Roussel. Rules of the 2011 SAT Competition. http://www.satcompetition.org/2011/.
6. H. Kleine Büning and O. Kullmann. Minimal unsatisfiability and autarkies. In A. Biere, M. J. H. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability*, chapter 11, pages 339–401. IOS Press, 2009.
7. D. Le Berre and A. Parrain. The Sat4j library, release 2.2. *Journal on Satisfiability, Boolean Modeling and Computation*, 7:59–64, 2010.
8. J. Marques-Silva. Minimal unsatisfiability: Models, algorithms and applications. In *Int'l Symposium on Multiple-Valued Logic*, pages 9–14, 2010.
9. J. Marques-Silva and I. Lynce. On improving MUS extraction algorithms. In *Proc. of SAT 2011*, 2011.
10. A. Nadel. Boosting minimal unsatisfiable core extraction. In *Formal Methods in Computer-Aided Design*, 2010.
11. V. Schuppan. Towards a notion of unsatisfiable cores for LTL. In *Fundamentals of Software Engineering, Third IPM Int'l Conference*, pages 129–145, 2010.
12. G. S. Tseitin. On the complexity of derivations in the propositional calculus. *Studies in Mathematics and Mathematical Logic*, Part II:115–125, 1968.