Complete Trigger Selection in Satisfiability Modulo First-Order Theories

Christopher Lynch[1,*,†], Stephen Miner[1]

[1]Clarkson University, 8 Clarkson Avenue, Potsdam NY 13699-5815

Abstract

Let T be an SMT solver with no theory solvers except for Quantifier Instantiation. Given a set of first-order clauses S saturated by Resolution (with a valid literal selection function) we show that T is complete if its Trigger function is the same as the literal selection function. So if T halts with a ground model G, then G can be extended to a model in the theory of S. In addition for a suitable ordering, if all maximal literals are selected in each clause, then T will halt on G, so it is a decision procedure for the theory S. Also, for a suitable ordering, if all clauses are Horn, or all clauses are 2SAT, then T solves the theory S in polynomial time.

Keywords

SMT, Trigggers, Quantifier Instantiation, First-order Theorem Proving

1. Introduction

SMT solvers [1] are efficient at satisfiability problems over several theories with specialized decision procedures. For first-order theories where a specialized decision procedure has not been implemented, a background theory can often be represented by quantified first-order clauses.[1] The SMT solver instantiates universally quantified FO clauses into ground clauses, which are handled by a SAT solver. To decide which instances are useful the SMT solver can use *triggers* [2, 3]. A trigger function maps each FO clause to a set of terms in the clause. If the terms in this set match existing ground terms, that triggers an instantiation.

Researchers have studied practical methods of selecting triggers. If triggers are selected well, the SMT solver can quickly solve unsatisfiable problems. However, if the problem is satisfiable, the SMT solver will often run forever or halt with a partial propositional model. If the SMT solver halts with a partial propositional model, it will not know if that propositional model will extend to a model of the FO clauses.

This paper is a result of our initial efforts to understand in what instances an SMT solver can be assured that enough instances have been generated to determine satisfiability. The subject is first-order logic without equality. We are motivated by completeness results involving selection functions in resolution-based first-order theorem proving [4]. We show a relationship between selection functions and the trigger functions of SMT solvers. We have started to extend these

SMT 2023: 21st International Workshop on Satisfiability Modulo Theories July 5–6, 2023, Rome Italy

clynch@clarkson.edu (C. Lynch); minersj@clarkson.edu (S. Miner)
0000-0003-1141-0665 (C. Lynch); 0009-0001-8679-036X (S. Miner)

[1]In this paper, the word "theory" refers to a satisfiable set of first-order clauses. We abbreviate it as FO-theory.

results to equational logic [5], and future research will be to extend them to specialized theories.

As an example of the problem faced by SMT solvers, consider the following FO theory represented by clauses, where capital letters are universally quantified variables. This example shows that even if the FO theory has no disjunction, SMT solvers already have trouble:

Example 1.

$$g(s(X), X)$$

$$\neg g(X, X)$$

If we give this theory to z3 [6] and assert $g(a, b)$, z3 returns "unknown" when using the default mbqi (model-based quantifier instantiation) [7]. If we turn off mbqi and set $g(s(X), X)$ and $g(X, X)$ as triggers, z3 will quickly halt and say "unknown".[2]. The SMT solver will have generated enough instances to determine satisfiability, but it is not aware of that. SMT solvers do well with conjunctive normal form problems without uninterpreted function symbols, but may have trouble with satisfiable problems with uninterpreted function symbols.

We now consider an FO theory that contains disjunction, to discuss trigger selection:

Example 2.

$$C_1 : \neg p(X_1, Y_1) \vee q(f(X_1), Y_1)$$

$$C_2 : \neg q(X_2, Y_2) \vee p(X_2, f(Y_2))$$

This is another theory that z3 cannot solve when presented with ground clause $p(a, b)$. The previous example only consisted of unit clauses, so there was no question of which literals to select for triggers. But in this example, we need to decide which literals to select for triggers. So we now consider three possible trigger selection strategies.

1. If we select $q(f(X_1), Y_1)$ and $p(X_2, f(Y_2))$ as triggers, we show whenever an SMT solver halts without saying "unsatisfiable", the ground model it has created is actually a model of the FO theory. In fact, for a fragment of FO logic to which this theory with this selection function belongs, we show that the SMT solver is a polynomial-time decision procedure.
2. If we select $\neg p(X_1, Y_1)$ and $\neg q(X_2, Y_2)$ as triggers,[3] a halting SMT solver can determine satisfiability. Unfortunately, given ground clause $p(a, b)$ the procedure will not halt.
3. If we select $q(f(X_1), Y_1)$ and $\neg q(X_2, Y_2)$, then the SMT solver will return "unknown", because ground clauses $p(a, b)$ and $\neg p(f(a), f(b))$ are unsatisfiable in that theory, but the SMT solver will not generate any instances. However, if we add the FO clause $\neg p(X_3, Y_3) \vee p(f(X_3), f(Y_3))$ to the FO theory, and select either literal in that clause, the SMT solver is complete.

In the first trigger selection (and the third one with the extra clause), the SMT solver creates enough instantiations to guarantee satisfiability. Unfortunately, the instantiations will not halt

[2]We don't mean to pick on z3. We also ran this on cvc5 [8], veriT [9] and SMTInterpol [10]. They all returned "unknown" or ran forever.

[3]To reduce instantiation, we use entire literals as triggers.

for the second one. For all but the second case, there is an ordering where we selected all the maximal literals in each clause. For the second case, there is no such ordering.

Our purpose is not to create a new inference system, but to understand when existing SMT solvers could answer "satisfiable" instead of "unknown", such as the above examples. If a set of FO clauses is saturated under Resolution with a valid selection function (as defined below), we choose the literals selected during Resolution to be the triggers. If an SMT solver halts with "unsatisfiable", the problem is unsatisfiable. But if the SMT solver halts without detecting unsatisfiability, most SMT solvers would say "unknown". However, using our method of trigger selection, we can know the problem is satisfiable. Furthermore, the partial ground model that the SMT solver has constructed can be extended to a model of the FO theory. In Example 1, the FO theory is saturated. In Example 2, the first-order theory is saturated under all three selection functions, assuming that the additional clause is added in the third case.

If, in addition, a single maximum literal is selected in each clause in the saturation of the FO theory, the SMT procedure will halt, and therefore the SMT procedure is a decision procedure. Alternatively, if the order is isomorphic to ω, [4] then selecting all maximal literals will give a decision procedure. If, in addition, the chosen order is a polynomial ordering which is totalizable on ground terms, the SMT solver is guaranteed to decide satisfiability in polynomial time if all clauses are Horn or all clauses contain at most two literals.

In Section 2 of this paper, we give some well-known definitions and some definitions specific to this paper. In Section 3 we define the inference rules used to model our procedure. Section 4 proves the completeness. Section 5 shows cases where we are guaranteed to have a decision procedure and where it is guaranteed to run in polynomial time. Section 6 gives related work, and Section 7 summarizes the paper and gives some important future work. All proofs can be found in [11].

2. Preliminaries

We consider a set of ground formulas modulo a set of first-order formulas, which are in conjunctive normal form. We follow standard definitions for Resolution theorem proving [4], plus some new definitions that are specific to this paper.

We assume we are given a set of variables, which we represent with capital letters, and a set of uninterpreted function symbols of various arities, represented with lower case letters. An arity is a non-negative integer. *Terms* are defined recursively in the following way: each variable is a term, and if $t_1, \cdots, t_n$ are terms, and f is of arity $n \geq 0$, then $f(t_1, \cdots, t_n)$ is a term. If P is a predicate symbol of arity n, and if $t_1, \cdots, t_n$ are terms, then $P(t_1, \cdots, t_n)$ is an *atom.* Any atom or negation of an atom is a *literal.* A literal is called *negative* if it is negated, and *positive* otherwise. For all literals L, we define $\bar{L}$ so that $\bar{L} = \neg L$ and $\bar{\neg L} = L$. A *clause* is a multiset of literals, representing a disjunction of literals. If L is a literal, and Γ is a set of literals, we will write $L \vee \Gamma$ to represent $\{L\} \cup \Gamma$. We use $\perp$ to represent the empty clause. We will use "$-$" to denote multiset difference. For any object C, define $Vars(C)$ as the set of variables in C. If $Vars(C) = \emptyset$ we say that C is *ground*, otherwise we say that C is non-ground.

[4]There are only finitely many atoms smaller than any given atom.

A *substitution* is a mapping from the set of variables to the set of terms, which is almost everywhere the identity. We identify a substitution with its homomorphic extension. Composition of substitutions σ and ρ is defined so that $X(\sigma\rho) = (X\sigma)\rho$ for all variables X. If θ is a substitution then $Dom(\theta) = \{X \mid X\theta \neq X\}$, and $Ran(\theta) = \{X\theta \mid X \in Dom(\theta)\}$. A substitution θ matches A to B if $A\theta = B$, and is a *unifier* of A and B, if $A\theta = B\theta$. σ is a *most general unifier* of A and B, written $\sigma = mgu(A, B)$ if σ is a unifier of A and B, and for all unifiers θ of A and B, there is a substitution ρ such that $X\sigma\rho = X\theta$ for all X in $Vars(A \cup B)$. Given a clause C, define $Gr(C) = \{C\theta | C\theta \text{ is ground }\}$. Given a set of clauses S, let $Gr(S) = \bigcup_{C \in S} Gr(C)$.

We assume an ordering $<$ is a well-founded ordering which is *stable*, meaning that if $s < t$ then $s\theta < t\theta$. We assume the ordering is totalizable on all ground terms and atoms. This means the ordering can be extended to an ordering that is total on ground terms. It can be extended to literals in any way such that $A < \neg A$ for all atoms A. We also assume the ordering is an *atom ordering* meaning that for all literals L and M, $L > M$ implies $L > \bar{M}$. Clauses are compared using the multiset ordering. A literal L is said to be *maximum* in a clause C if L is larger than all other literals in C, and *maximal* in C if no other literal in C is larger than L. An order is a *polynomial ordering* if each atom only has polynomially many smaller atoms.

An *partial interpretation* (or just *interpretation*) I is defined as a consistent set of ground literals such that $I \models L$ if and only if $L \in I$. Therefore, an atom A is undefined in I if $A \notin I$ and $\neg A \notin I$. This differs with some definitions of interpretations where just the true positive literals are given. Since clauses are multisets representing disjunctions, $I \models C$ if $I \cap C \neq \emptyset$, otherwise C is either false or undefined in I. If C is not ground then we say $I \models C$ if $I \models Gr(C)$. If I is an interpretation and S is a set of clauses, then I is a *model* of S if $I \models C$ for all $C \in S$. Interpretations I_1 and I_2 are *compatible* if there is no literal L such that $L \in I_1$ and $\bar{L} \in I_2$. If I_1 and I_2 are compatible then $I_1 \cup I_2$ is also an interpretation, furthermore, for any literal L, $I_1 \cup I_2 \models L$ if and only if $I_1 \models L$ or $I_2 \models L$.

Given an interpretation I and a ground clause C, let $Filter(C, I) = \{L \in C \mid I \not\models \bar{L}\}$. If S is a set of ground clauses, let $Filter(S, I) = \{Filter(C) \mid C \in S\, I \not\models C\}$. i.e., $Filter(S, I)$ is created from S by removing all clauses true in I, and then removing all literals false in I from the remaining clauses.

Example 3. *Consider interpretation $I = \{\neg p(a), p(b)\}$ where S is the set of clauses in the following example:*

$$C_1 : p(a) \vee \neg p(b) \vee p(c)$$

$$C_2 : \neg p(a) \vee \neg p(b) \vee p(d)$$

Then $Filter(C_1, I) = p(c)$, and $Filter(S, I)$ is the set consisting of the unit clause $p(c)$.

3. Inference System

We want to model an SMT solver without any theories except for a quantified FO theory in CNF, represented by clauses with universal variables. Given a set of clauses S, let $g(S)$ be the set of all ground clauses in S, and let $ng(S)$ be the set of all non-ground clauses in S. An SMT solver would build a model from $g(S)$. Call that model $M_{g(S)}$. For most of this paper, it will not be important how that model is built. The SMT solver will use $M_{g(S)}$ to instantiate the clauses of $ng(S)$. We will use inference rules to model the instantiation process.

Let $Trig$ be a function so that for each clause C in $ng(S)$, $Trig(C) \subseteq C$ and $Vars(Trig(C)) = Vars(C)$, which determines which parts of C are used for instantiation. Below we show how to choose triggers in such a way that when we have a model, and no more instantiations can be performed, we can deduce that S is satisfiable. The Instantiation rule is used to instantiate non-ground clauses based on a ground interpretation I.

I-Instantiation:

$$\frac{L_1 \vee \cdots \vee L_n \vee \Gamma}{(L_1 \vee \cdots \vee L_n \vee \Gamma)\theta}$$

where

1. $L_1 \vee \cdots \vee L_n \vee \Gamma \in ng(S)$,
2. $Trig(L_1 \vee \cdots \vee L_n \vee \Gamma) = \{L_1, \cdots, L_n\}$
3. there exists $L'_1 \cdots L'_n$ in I such that $\bar{L}_i\theta = L'_i$ for all $1 \leq i \leq n$

We do not consider equality, so we only require θ to be a matcher, not an E-matcher. SMT solvers allow triggers to be subterms of a literal. But to reduce the number of instantiations, we only use entire literals as triggers. Furthermore, we only need to match a ground literal in the model onto the complement of a non-ground literal. This allows the instantiation rule to be more restrictive than is usually the case for trigger-based instantiation in SMT. Finally, SMT solvers allow for different possible sets of triggers for the same clause. We only require one set of triggers for each clause.

A set of clauses S is *saturated by Instantiation* if either $g(S)$ is unsatisfiable or there exists a model $M_{g(S)}$ of $g(S)$ such that every conclusion of an $M_{g(S)}$-Instantiation inference is in S.

Example 4.

$$C_1 : \neg p(X_1, Y_1) \vee q(f(X_1), Y_1)$$

$$C_2 : \neg q(X_2, Y_2) \vee p(X_2, f(Y_2))$$

$$C_3 : \neg p(f(a), f(b))$$

We define $Trig_1$ so that $Trig_1(C_1) = \{q(f(X_1), Y_1)\}$ and $Trig_1(C_2) = \{p(X_2, f(Y_2))\}$. Given the model $M_1 = \{\neg p(f(a), f(b))\}$ of C_3, we apply the Instantiation rule to create the clause $C_4 = \neg q(f(a), b) \vee p(f(a), f(b))$. Then, we can create a new model $M_2 = \{\neg p(f(a), f(b)), \neg q(f(a), b)\}$ of C_3 and C_4. Instantiation then creates $C_5 = \neg p(a, b) \vee q(f(a), b)$, and we create a new model $M_3 = \{\neg p(f(a), f(b)), \neg q(f(a), b), \neg p(a, b)\}$ of C_3 and C_4. These five clauses are now saturated by Instantiation.

Consider the same set of three clauses with a different trigger function $Trig_2$ defined so that $Trig_2(C_1) = \{q(f(X_1), Y_1)\}$ and $Trig_2(C_2) = \{\neg q(X_2, Y_2)\}$. Let M_1 be the same model as before. There are no instantiations, so the three clauses are saturated by Instantiation. Suppose we also had the clause $C_6 = p(a, b)$. Then the model of the clauses would be $M_4 = \{\neg p(f(a), f(b)), p(a, b)\}$. The set of clauses $\{C_1, C_2, C_3, C_6\}$ is again saturated by Instantiation, although it is unsatisfiable. In other words, this was not a good choice of triggers.

The theory $\{p(X_1), \neg p(X_2)\}$, with no ground clauses, is unsatisfiable, but no instantiations exist. So there cannot be a set of triggers that guarantees completeness for all formulas. To address this problem, we require the non-ground clauses to be saturated under the Factoring and Resolution inference rule defined below. These inference rules depend on a selection function, which selects the literals in each clause that may be used in an inference. A selection function maps a clause to a subset of its literals, just like the trigger function. A selection function Sel is *valid* if, for each clause C and for each $T \subseteq Sel(C)$ with $Vars(T) \neq Vars(C)$, either $Sel(C) - T$ contains all maximal literals in $C - T$, or $Sel(C) - T$ contains a negative literal.

Before we give an intuition of this definition, let us give some properties:

Proposition 1. *For any clause C and valid selection function Sel, $Vars(Sel(C)) = Vars(C)$.*

Proposition 2. *Let Sel be a selection function such that for all C, $Vars(Sel(C)) = Vars(C)$ and either (1) $Sel(C)$ contains only negative literals or (2) $Sel(C)$ is a singleton set containing the maximum literal in C. Then Sel is a valid selection function.*

Selection functions normally select all maximal literals or a negative literal. We additionally require $Sel(C)$ to contain all the variables in C. We also require that if some of the literals from the selected set are removed from the clause, without covering all the variables, the remaining selected set must contain all maximal literals in the remaining clause or a negative literal. In the completeness proof, we will filter our clauses by the ground model, and must ensure that the filtered clauses still have a valid selection function.

In Example 4 with trigger function $Trig_1$, if the selection function is the same as the trigger function, it is easy to construct an ordering where the selected literal is the largest in C_1 and C_2. So this is a valid selection function. If the trigger function is $Trig_2$, the same is true for C_1. For C_2, $Trig_2(C_2)$ contains only negative literals, so the selection function is valid.

Let us look at one more example. Consider the following set of clauses, with an ordering where $r(t_1) > q(t_2) > p(t_3)$ for all terms t_1, t_2, t_3.

Example 5.

$$C_1 : \neg p(X_1) \vee \neg q(X_1)$$

$$C_2 : p(X_2) \vee \neg q(X_2)$$

$$C_3 : \neg p(X_3) \vee q(X_3)$$

$$C_4 : p(X_4) \vee q(X_4) \vee \neg r(Y_4)$$

Let Sel_1 be the selection function such that $Sel_1(C_1) = \{\neg p(X_1)\}$, $Sel_1(C_2) = \{\neg q(X_2)\}$, $Sel_1(C_3) = \{q(X_3)\}$, and $Sel_1(C_4) = \{\neg r(Y_4)\}$. Sel_1 is not valid, because $Sel_1(C_4)$ does not contain all the variables of C_4. So let Sel_2 be a selection function identical to Sel_1 on the first three clauses, but $Sel_2(C_4) = \{p(X_4), \neg r(Y_4)\}$. This selection function is also not valid because in clause C_4 if we let $D = \{\neg r(Y_4)\}$, then $Sel_2(C_4) - D = \{p(X_4)\}$, and $p(X_4)$ is neither maximal nor negative in $p(X_4) \vee q(X_4)$. Finally we define Sel_3 to be the same as Sel_1 on the first three clauses but $Sel_3(C_4) = \{q(X_4), \neg r(Y_4)\}$. This selection function is valid.

Given a valid selection function, our inference system will consist of three inference rules. We defined Instantiation above. Below we define Resolution and Factoring:

Resolution:

$$\frac{A \vee \Gamma \quad \neg B \vee \Delta}{(\Gamma \vee \Delta)\sigma}$$

where (1) $A \vee \Gamma \in ng(S)$, (2) $\neg B \vee \Delta \in ng(S)$, (3) A is selected in $A \vee \Gamma$, (4) $\neg B$ is selected in $\neg B \vee \Delta$, and (5) $\sigma = mgu(A, B)$.

Factoring:

$$\frac{A \vee B \vee \Gamma}{(A \vee \Gamma)\sigma}$$

where (1) $A \vee B \vee \Gamma \in ng(S)$, (2) A is selected in $A \vee B \vee \Gamma$, and (3) $\sigma = mgu(A, B)$.

When applying Resolution and Factoring, it is important to remove redundant clauses. In particular, implementations remove subsumed clauses and tautologies.

Definition 1. *A clause C* subsumes *a clause D if there is a substitution σ such that $C\sigma \subseteq D$. A clause C is a* tautology *if there is an atom A such that $A \in C$ and $\neg A \in C$. A set of clauses S (possibly infinite) is* saturated *by Resolution and Factoring if the conclusion of every Resolution and Factoring inference in S is either a tautology or is subsumed in S. S is* completely saturated *if S is saturated by Instantiation and saturated by Resolution and Factoring.*

In Example 4, with selection and trigger function $Trig_1$, the set $\{C_1, C_2, C_3, C_4, C_5\}$ is completely saturated. If the selection and trigger function are $Trig_2$, the set $\{C_1, C_2, C_3\}$ is saturated by Instantiation but not saturated by Resolution and Factoring. The result of a Resolution between C_1 and C_2 is $C_7 = \neg p(X_7, Y_7) \vee p(f(X_7), f(Y_7))$. We extend the selection and trigger function to C_7. Suppose we extend $Trig_2$ so that $Trig_2(C_7) = \{p(f(X_7), f(Y_7))\}$. Then an instantiation will give us $C_8 = \neg p(a, b) \vee p(f(a), f(b))$. Extending the model to $\{p(a, b), p(f(a), f(b))\}$ allows us to see that $\{C_1, C_2, C_3, C_7, C_8\}$ is completely saturated.

In Example 5, if we add a ground clause $C_5 = r(a)$, then $\{C_1, C_2, C_3, C_4, C_5\}$ is completely saturated under selection function Sel_2, because every Resolution inference yields a tautology, even though the set is unsatisfiable. So if we had only required valid selection functions to select a negative literal or all maximal literals in each clause, we could not prove completeness, even if we additionally required that the selected literals contain all the variables in the clause. Using selection function Sel_3, the set of clauses is not saturated under Resolution and Factoring.

4. Completeness Proof

In this section we will prove the completeness of our inference system. Given a set of clauses S, the first step in our completeness proof is to filter the ground instances of $ng(S)$ with a model $M_{g(S)}$ of $g(S)$. Let $F(S) = Filter(Gr(ng(S)), M_{g(S)})$.

We explain $F(S)$ with an example, where we write $f^n(a)$ to abbreviate f applied n times to a. Note that S is not saturated under Instantiation in this example, although in the proof we only construct $F(S)$ for completely saturated sets.

Example 6. *Let $ng(S) = \{\neg p(X) \vee p(f(X))\}$. Suppose that we have the model $M_{g(S)} = \{\neg p(f(a)), p(f^3(a))\}$ of $g(S)$. Then $Gr(ng(S)) = \{\neg p(f^n(a)) \vee p(f^{n+1}(a)) \mid n \geq 0\}$. So $F(S)$ is the set of clauses $\{\neg p(a), p(f^4(a))\} \cup \{\neg p(f^n(a)) \vee p(f^{n+1}(a)) \mid n \geq 4\}$.*

Instances of subsumed clauses and tautologies in S are also subsumed clauses and tautologies in $F(S)$, if they exist in $F(S)$.

Lemma 1. *Let D be a clause in S. Let θ be a ground substitution. Let $D' = Filter(D\theta, M_{g(S)})$. Then (a) If D is a tautology then either D' is not in $F(S)$ or D' is a tautology. (2) If D is subsumed in S then either D' is not in $F(S)$ or there is a clause C' in $F(S)$ such that $C' \subseteq D'$.*

Subsumption and tautology deletion are instances of the concept of redundancy,[5] A clause C may be redundant in S but $Filter(C)$ not redundant in $F(S)$, so our filtering technique does not cover redundancy in full. However, subsumption and tautology deletion are what is mainly used in practice to control saturation.

Example 7. *Let S be a set of clauses such that $ng(S) = \{p(X) \vee q(X), \neg q(X), \neg r(X) \vee p(X)\}$ with $M_{g(S)} = \{r(a)\}$ and an ordering such that $p(s) < q(t) < r(u)$ for all terms s, t, u. Then $\neg r(X) \vee p(X)$ is implied by smaller clauses $p(X) \vee q(X)$ and $\neg q(X)$. But when we apply filtering, we get clauses $\{p(a) \vee q(a), \neg q(a), p(a)\}$, and $p(a)$ is not implied by smaller clauses.*

To prove completeness, we will let S be a completely saturated set of clauses with $Trig = Sel$. We show that if $\perp$ is not in S and $g(S)$ is satisfiable, then a model of $ng(S)$ can be constructed which is compatible with the model of $g(S)$.

First we need some definitions. For a set of clauses S, let $S_{<C} = \{D \in S \mid D < C\}$ be the set of clauses in S that are smaller than C. We will create an interpretation from a set of positive literals. So, given a set of positive literals T and a set of literals U, we define $Int(T, U) = T \cup \{\neg A \mid A \notin T, (U - T) \cap \{A, \neg A\} \neq \emptyset\}$ In other words, it is the interpretation where all the atoms in T are true, and every atom in U that has not been made true in T is false. For each clause $C \in F(S)$, we will define $P_{<C}$, $M_{<C}$ and P_C co-recursively.

Definition 2. *Let $F(S)$ be the clause set defined above, and C be a clause in $F(S)$.*

1. *Define $P_{<C}$ as the set of positive literals $\bigcup_{D \in F(S)_{<C}} P_D$, where P_D is defined below.*
2. *Define $P^{F(S)} = \bigcup_{C \in F(S)} P_C$, the union of all the P_C defined below.*
3. *Let $M_{<C} = Int(P_{<C}, C \cup \bigcup F(S)_{<C})$, which means that $M_{<C}$ is the interpretation that makes true all the atoms in $P_{<C}$, and makes false all other atoms in clauses of $F(S)$ that are smaller than or equal to C.*
4. *Similarly, let $M^{F(S)} = Int(P^{F(S)}, \bigcup F(S))$.*

Simultaneously we define $P_C = \{A\}$ for atom A if (1) $M_{<C} \not\models C$, (2) A is the largest literal in C, (3) A is selected in C, and (4) A only occurs once in C. Otherwise $P_C = \emptyset$.

If $P_C = \{A\}$, we say that C produces *A.*

The completeness proof is similar to the standard proof of completeness of Resolution, except we deal with filtered clauses, so lifting is more complex. Also, we use Instantiation when the filtering removes all the selected literals. The next lemma follows from the definition of P_C.

[5]A clause is redundant if implied by smaller clauses.

Lemma 2. *Let C be a clause in $F(S)$. Let L be a literal in C. Then (1) If L is not maximum in C then $M^{F(S)} \models L$ if and only if $M_{<C} \models L$. (2) If C produces L then $L \in M^{F(S)}$.*

For the proof below, we will assume that for every $C \in S$ and $L \in C$, where $C' = Filter(C\theta, M_{g(S)})$, then $L\theta$ is selected in $C\theta$ if and only if $L\theta \in C'$ and L is selected in C.

Theorem 1. *Let Sel be a valid selection function with $Trig = Sel$. Suppose that S is completely saturated and $\perp \notin S$ and $g(S)$ is satisfiable. Let $M_{g(S)}$ be a model of $g(S)$ such that S is saturated by $M_{g(S)}$-Instantiation. Then $M^{F(S)} \models F(S)$ where $F(S) = Filter(Gr(ng(S)), M_{g(S)})$.*

Proof Sketch 1. *As usual in model construction proofs, we prove that $M^{F(S)}$ is a model of $F(S)$ by showing that a least counterexample C leads to a contradiction. As usual, if a negative is selected, Resolution leads to a smaller counterexample. If the largest positive is selected, Factoring leads to a smaller counterexample. However, all selected literal may have been filtered out. In which case, Instantiation leads to a contradiction. In all cases, the filtering process allows ground inferences to be lifted.*

We can combine $M^{F(S)}$ with $M_{g(S)}$ to get a model of S.

Corollary 1. *Let Sel be a valid selection function with $Trig = Sel$. Suppose that S is completely saturated and $\perp \notin S$ and $g(S)$ is satisfiable. Let $M_{g(S)}$ be a model of $g(S)$ such that S is saturated by $M_{g(S)}$-Instantiation. Let $F(S) = Filter(Gr(ng(S)), M_{g(S)})$. Then $M^{F(S)}$ is compatible with $M_{g(S)}$ and $M^{F(S)} \cup M_{g(S)} \models S$*

5. Decision Procedure and Complexity Results

Next we assume the clauses are saturated by Resolution and Factoring, and find cases where they can be further finitely saturated by Instantiation. Then SAT Solving plus Instantiation is a decision procedure for the theory of $ng(S)$. For example, if the selection function selects a single maximum literal in each clause.

Theorem 2. *Let Sel be a valid selection function such that a single maximum literal is selected in each clause of $ng(S)$, with $Trig = Sel$. Suppose that S is saturated by Resolution and Factoring. Then S can be saturated by Instantiation in finite time.*

We also get a decision procedure if the ordering used is order isomorphic to ω.

Theorem 3. *Let Sel be a valid selection function such that all maximal literals are selected in each clause, with an ordering that is order isomorphic to ω. Suppose that $Trig = Sel$ and that S is saturated by Resolution and Factoring. Then S can be saturated by Instantiation in finite time.*

If the selection function selects a negative literal, then the Instantiation rule may not halt.

Example 8. *Consider Example 4 where $Trig(C_1) = \{\neg p(X_1, Y_1)\}$ and $Trig(C_2) = \{\neg q(X_2, Y_2)\}$. This is saturated by Resolution and Factoring, but Instantiation with ground clause $p(a, a)$ creates infinitely many clauses.*

Even if all maximal literals are selected in each clause, there may still be infinitely many instantiations, as in the following theory, with an ordering such that $p(s) > q(t)$ for all s and t.

Example 9.

$$C : \neg p(X) \vee \neg q(Y) \vee q(f(Y))$$

Suppose that $Sel(C) = \{\neg p(X), \neg q(Y)\}$. This is a valid selection function, and $p(X)$ is the maximum literal. But suppose we have $p(c)$ and $q(c)$ in the ground model. Instantiation will create $q(f^n(c))$ for all n, all literals smaller than $p(c)$.

We would also like to determine conditions where Instantiation halts in polynomial time, given that S is already saturated by Resolution and Factoring. This requires that only polynomially many instantiations are computed, and that the SAT solver runs in polynomial time.

To get polynomial time results, we must examine the details of an SMT solver. We assume the SMT solver is based on a CDCL SAT solver [12]. We assume the SAT solver can decide the value of a literal, propagate a literal, backjump and learn a literal, but we don't assume the Forget or Restart rule.

For Horn clauses, we assume that the initial decision about an atom is to make it false. It is well-known that satisfiability of ground Horn clauses can be decided in polynomial time, but we are not aware of any results that CDCL SAT solvers solve Horn clauses in polynomial time.

Theorem 4. *Let $<$ be a polynomial ordering. Let Sel be a valid selection function such that all maximal literals are selected in each clause of $ng(S)$, with $Trig = Sel$. If S is saturated by Resolution and Factoring, and S only contains Horn clauses, then the saturation of S by CDCL SAT solving and Instantiation runs in polynomial time if the initial decision for the truth value of ground atoms is false. Therefore SAT solving plus Instantiation is a polynomial time decision procedure for the theory of $ng(S)$, if all ground clauses are Horn clauses.*

Clause learning is crucial for 2SAT. Satisfiability of 2SAT can be decided in polynomial time, but we have not seen any results that CDCL SAT solvers solve 2SAT in polynomial time.

Theorem 5. *Let $<$ be a polynomial ordering. Let Sel be a valid selection function such that all maximal literals are selected in each clause of $ng(S)$, with $Trig = Sel$. If S is saturated by Resolution and Factoring, and S contains no clauses with more than two literals, then the saturation of S by CDCL SAT solving plus Instantiationin runs in polynomial time. Therefore CDCL SAT solving plus Instantiation is a polynomial time decision procedure for the theory of $ng(S)$, if all ground clauses contain at most two literals.*

Example 4 is Horn and 2SAT. Consider selection and trigger function $Trig_1$, with $p(s_1, s_2) \leq p(t_1, t_2)$ if s_1 is a subterm of t_1 and s_2 is a subterm of t_2. We do the same for q. CDCL SAT solving plus Instantiation will solve this theory in polynomial time.

6. Related Work

The paper [13] discusses how a good selection of triggers will give a decision procedure. Their approach is somewhat different from ours. The user needs to supply a correctness and termination proof that the trigger choice will give a decision procedure. Our method is automatic, and inherits the trigger selection function directly from the selection function used in saturation. Good trigger selection is discussed from a practical point of view in [14, 15].

Other papers suggest other approaches to quantifiers instead of triggers. Some successful approaches are Model-Based Quantifier Instantiation [7] for satisfiable problems, and Conflicting Instances[16, 17] for unsatisfiable problems. Several other approaches have been proposed and implemented [18, 19, 20, 21, 22, 23]. Our paper only deals with first-order theories without equality, whereas the above mentioned papers consider other SMT theories.

Other papers have used Saturation under Ordered Resolution [24], as a way to show that a first-order Theory is a Local Theory[25] meaning that the only instantiations necessary are to replace variables with terms smaller than initial ground terms. In this approach, all possible instantiations are made at the beginning. This approach was further extended in [26, 27] to cover other theories. But these extensions still require instantiating all the instances at the beginning. Finally, in [28], an approach was implemented where instantiations are only made when necessary. But that approach is based on the instance generation method of [29, 30], which is not the same as the SMT method. Finally, in [31], the local theory method was implemented in an SMT setting. These ideas don't involve triggers.

Another technique is for an SMT solver to call an FO theorem prover [32, 33, 34]. Our method is different in that we do not need a first-order theorem prover after saturation of the FO clauses.

7. Conclusion

We analyzed the completeness of the trigger selection function for SMT solvers with an FO theory T and no other theories. If T is saturated by Resolution and Factoring with a valid selection function identical to the trigger function, then Saturation by Instantiation gives a model of the ground clauses, that is also a model of those clauses modulo the first-order theory. Saturation by Instantiation is guaranteed to halt if the Selection function selects a single maximum literal in each clause, or if all maximal clauses are selected using an ordering isomorphic to ω. If it is also a polynomial ordering, then Saturation by Instantiation is guaranteed to halt in polynomial time if all clauses are Horn Clauses, or if all clauses contain at most two literals.

SMT solvers often return "unknown" on problems that seem to be easily shown to be satisfiable. We hope implementers of SMT solvers will use our results to return "satisfiable" in more cases. It requires no change to the SMT process. The only change is in the pre-processing, where the SMT solver checks if the FO classes are saturated by a valid selection function, and uses the identical trigger function. At the end, if no contradiction is found, the SAT solver will return "satisfiable", and also return a model modulo extendable to the FO theory.

We have implemented an SMT solver that, given a satisfiable saturated first-order theory, will detect satisfiability and return a ground model. We experimented with our SMT solver using some first-order theories presented in the appendices. Since this is a new SMT solver, we don't expect it to be competitive in speed with existing SMT solvers. However, this paper is not about increasing the speed of an SMT solver. It is about making SMT solvers more precise.

We plan lots of future work on this subject. To make this useful, we need to extend the results to more theories. We are working on extending it to equality with uninterpreted function symbols. Later work will be to extend it to other specialized theories.

Even in the non-equational case, there are many unanswered questions. For example, can this be extended to theories which cannot be saturated under Resolution. These results basically give Herbrand models. There may be ways to use other models to strengthen these results. There are several more detailed results that are not answered in this paper. Does the proof technique work for all cases of redundancy, not just subsumption and tautology deletion? Other decision procedures may be possible by loosening the restrictions on the ordering.

References

[1] C. W. Barrett, R. Sebastiani, S. A. Seshia, C. Tinelli, Satisfiability modulo theories, in: A. Biere, M. Heule, H. van Maaren, T. Walsh (Eds.), Handbook of Satisfiability - Second Edition, volume 336 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2021, pp. 1267–1329. URL: https://doi.org/10.3233/FAIA201017. doi:10.3233/FAIA201017.

[2] D. Detlefs, G. Nelson, J. B. Saxe, Simplify: a theorem prover for program checking, J. ACM 52 (2005) 365–473. URL: https://doi.org/10.1145/1066100.1066102. doi:10.1145/1066100.1066102.

[3] L. M. de Moura, N. S. Bjørner, Efficient e-matching for SMT solvers, in: F. Pfenning (Ed.), Automated Deduction - CADE-21, 21st International Conference on Automated Deduction, Bremen, Germany, July 17-20, 2007, Proceedings, volume 4603 of *Lecture Notes in Computer Science*, Springer, 2007, pp. 183–198. URL: https://doi.org/10.1007/978-3-540-73595-3_13. doi:10.1007/978-3-540-73595-3_13.

[4] L. Bachmair, H. Ganzinger, Resolution theorem proving, in: J. A. Robinson, A. Voronkov (Eds.), Handbook of Automated Reasoning (in 2 volumes), Elsevier and MIT Press, 2001, pp. 19–99. URL: https://doi.org/10.1016/b978-044450813-3/50004-7. doi:10.1016/b978-044450813-3/50004-7.

[5] R. Nieuwenhuis, A. Rubio, Paramodulation-based theorem proving, in: J. A. Robinson, A. Voronkov (Eds.), Handbook of Automated Reasoning (in 2 volumes), Elsevier and MIT Press, 2001, pp. 371–443. URL: https://doi.org/10.1016/b978-044450813-3/50009-6. doi:10.1016/b978-044450813-3/50009-6.

[6] L. M. de Moura, N. S. Bjørner, Z3: an efficient SMT solver, in: C. R. Ramakrishnan, J. Rehof (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings, volume 4963 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 337–340. URL: https://doi.org/10.1007/978-3-540-78800-3_24. doi:10.1007/978-3-540-78800-3_24.

[7] Y. Ge, L. M. de Moura, Complete instantiation for quantified formulas in satisfiabiliby modulo theories, in: A. Bouajjani, O. Maler (Eds.), Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings, volume 5643 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 306–320. URL: https://doi.org/10.1007/978-3-642-02658-4_25. doi:10.1007/978-3-642-02658-4_25.

[8] H. Barbosa, C. W. Barrett, M. Brain, G. Kremer, H. Lachnitt, M. Mann, A. Mohamed, M. Mo-

hamed, A. Niemetz, A. Nötzli, A. Ozdemir, M. Preiner, A. Reynolds, Y. Sheng, C. Tinelli, Y. Zohar, cvc5: A versatile and industrial-strength SMT solver, in: D. Fisman, G. Rosu (Eds.), Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I, volume 13243 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 415–442. URL: https://doi.org/10.1007/978-3-030-99524-9_24. doi:10.1007/978-3-030-99524-9_24.

[9] T. Bouton, D. C. B. D. Oliveira, D. Déharbe, P. Fontaine, verit: An open, trustable and efficient smt-solver, in: R. A. Schmidt (Ed.), Automated Deduction - CADE-22, 22nd International Conference on Automated Deduction, Montreal, Canada, August 2-7, 2009. Proceedings, volume 5663 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 151–156. URL: https://doi.org/10.1007/978-3-642-02959-2_12. doi:10.1007/978-3-642-02959-2_12.

[10] J. Christ, J. Hoenicke, A. Nutz, Smtinterpol: An interpolating SMT solver, in: A. F. Donaldson, D. Parker (Eds.), Model Checking Software - 19th International Workshop, SPIN 2012, Oxford, UK, July 23-24, 2012. Proceedings, volume 7385 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 248–254. URL: https://doi.org/10.1007/978-3-642-31759-0_19. doi:10.1007/978-3-642-31759-0_19.

[11] C. Lynch, S. Miner, Complete trigger selection in satisfiability modulo first-order theories, 2023. arXiv:2306.09436.

[12] J. Marques-Silva, I. Lynce, S. Malik, Conflict-driven clause learning SAT solvers, in: A. Biere, M. Heule, H. van Maaren, T. Walsh (Eds.), Handbook of Satisfiability - Second Edition, volume 336 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2021, pp. 133–182. URL: https://doi.org/10.3233/FAIA200987. doi:10.3233/FAIA200987.

[13] C. Dross, S. Conchon, J. Kanig, A. Paskevich, Adding decision procedures to SMT solvers using axioms with triggers, J. Autom. Reason. 56 (2016) 387–457. URL: https://doi.org/10.1007/s10817-015-9352-2. doi:10.1007/s10817-015-9352-2.

[14] K. R. M. Leino, C. Pit-Claudel, Trigger selection strategies to stabilize program verifiers, in: S. Chaudhuri, A. Farzan (Eds.), Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part I, volume 9779 of *Lecture Notes in Computer Science*, Springer, 2016, pp. 361–381. URL: https://doi.org/10.1007/978-3-319-41528-4_20. doi:10.1007/978-3-319-41528-4_20.

[15] M. Moskal, Programming with triggers, in: Proceedings of the 7th International Workshop on Satisfiability Modulo Theories, SMT '09, Association for Computing Machinery, New York, NY, USA, 2009, p. 20–29. URL: https://doi.org/10.1145/1670412.1670416. doi:10.1145/1670412.1670416.

[16] H. Barbosa, P. Fontaine, A. Reynolds, Congruence closure with free variables, in: A. Legay, T. Margaria (Eds.), Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part II, volume 10206 of *Lecture Notes in Computer Science*, 2017, pp. 214–230. URL: https://doi.org/10.1007/978-3-662-54580-5_13. doi:10.1007/978-3-662-54580-5_13.

[17] A. Reynolds, C. Tinelli, C. W. Barrett, Constraint solving for finite model finding in SMT

solvers, Theory Pract. Log. Program. 17 (2017) 516–558. URL: https://doi.org/10.1017/S1471068417000175. doi:10.1017/S1471068417000175.

[18] P. Rümmer, E-matching with free variables, in: N. S. Bjørner, A. Voronkov (Eds.), Logic for Programming, Artificial Intelligence, and Reasoning - 18th International Conference, LPAR-18, Mérida, Venezuela, March 11-15, 2012. Proceedings, volume 7180 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 359–374. URL: https://doi.org/10.1007/978-3-642-28717-6_28. doi:10.1007/978-3-642-28717-6_28.

[19] A. Reynolds, H. Barbosa, P. Fontaine, Revisiting enumerative instantiation, in: D. Beyer, M. Huisman (Eds.), Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part II, volume 10806 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 112–131. URL: https://doi.org/10.1007/978-3-319-89963-3_7. doi:10.1007/978-3-319-89963-3_7.

[20] P. Fontaine, H. Schurr, Quantifier simplification by unification in SMT, in: B. Konev, G. Reger (Eds.), Frontiers of Combining Systems - 13th International Symposium, FroCoS 2021, Birmingham, UK, September 8-10, 2021, Proceedings, volume 12941 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 232–249. URL: https://doi.org/10.1007/978-3-030-86205-3_13. doi:10.1007/978-3-030-86205-3_13.

[21] A. Reynolds, C. Tinelli, L. M. de Moura, Finding conflicting instances of quantified formulas in SMT, in: Formal Methods in Computer-Aided Design, FMCAD 2014, Lausanne, Switzerland, October 21-24, 2014, IEEE, 2014, pp. 195–202. URL: https://doi.org/10.1109/FMCAD.2014.6987613. doi:10.1109/FMCAD.2014.6987613.

[22] A. Niemetz, M. Preiner, A. Reynolds, C. W. Barrett, C. Tinelli, Syntax-guided quantifier instantiation, in: J. F. Groote, K. G. Larsen (Eds.), Tools and Algorithms for the Construction and Analysis of Systems - 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings, Part II, volume 12652 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 145–163. URL: https://doi.org/10.1007/978-3-030-72013-1_8. doi:10.1007/978-3-030-72013-1_8.

[23] J. Hoenicke, T. Schindler, Incremental search for conflict and unit instances of quantified formulas with e-matching, in: F. Henglein, S. Shoham, Y. Vizel (Eds.), Verification, Model Checking, and Abstract Interpretation - 22nd International Conference, VMCAI 2021, Copenhagen, Denmark, January 17-19, 2021, Proceedings, volume 12597 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 534–555. URL: https://doi.org/10.1007/978-3-030-67067-2_24. doi:10.1007/978-3-030-67067-2_24.

[24] D. A. Basin, H. Ganzinger, Automated complexity analysis based on ordered resolution, J. ACM 48 (2001) 70–109. URL: https://doi.org/10.1145/363647.363681. doi:10.1145/363647.363681.

[25] R. Givan, D. A. McAllester, Polynomial-time computation via local inference relations, ACM Trans. Comput. Log. 3 (2002) 521–541. URL: https://doi.org/10.1145/566385.566387. doi:10.1145/566385.566387.

[26] V. Sofronie-Stokkermans, Hierarchic reasoning in local theory extensions, in: R. Nieuwenhuis (Ed.), Automated Deduction - CADE-20, 20th International Conference on Automated

Deduction, Tallinn, Estonia, July 22-27, 2005, Proceedings, volume 3632 of *Lecture Notes in Computer Science*, Springer, 2005, pp. 219–234. URL: https://doi.org/10.1007/11532231_16. doi:10.1007/11532231_16.

[27] C. Ihlemann, S. Jacobs, V. Sofronie-Stokkermans, On local reasoning in verification, in: C. R. Ramakrishnan, J. Rehof (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings, volume 4963 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 265–281. URL: https://doi.org/10.1007/978-3-540-78800-3_19. doi:10.1007/978-3-540-78800-3_19.

[28] S. Jacobs, Incremental instance generation in local reasoning, in: A. Bouajjani, O. Maler (Eds.), Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings, volume 5643 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 368–382. URL: https://doi.org/10.1007/978-3-642-02658-4_29. doi:10.1007/978-3-642-02658-4_29.

[29] H. Ganzinger, K. Korovin, New directions in instantiation-based theorem proving, in: 18th IEEE Symposium on Logic in Computer Science (LICS 2003), 22-25 June 2003, Ottawa, Canada, Proceedings, IEEE Computer Society, 2003, pp. 55–64. URL: https://doi.org/10.1109/LICS.2003.1210045. doi:10.1109/LICS.2003.1210045.

[30] K. Korovin, iprover - an instantiation-based theorem prover for first-order logic (system description), in: A. Armando, P. Baumgartner, G. Dowek (Eds.), Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings, volume 5195 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 292–298. URL: https://doi.org/10.1007/978-3-540-71070-7_24. doi:10.1007/978-3-540-71070-7_24.

[31] K. Bansal, A. Reynolds, T. King, C. W. Barrett, T. Wies, Deciding local theory extensions via e-matching, in: D. Kroening, C. S. Pasareanu (Eds.), Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part II, volume 9207 of *Lecture Notes in Computer Science*, Springer, 2015, pp. 87–105. URL: https://doi.org/10.1007/978-3-319-21668-3_6. doi:10.1007/978-3-319-21668-3_6.

[32] C. Lynch, Q. Ta, D. Tran, SMELS: satisfiability modulo equality with lazy superposition, J. Autom. Reason. 51 (2013) 325–356. URL: https://doi.org/10.1007/s10817-012-9263-4. doi:10.1007/s10817-012-9263-4.

[33] L. M. de Moura, N. S. Bjørner, Engineering DPLL(T) + saturation, in: A. Armando, P. Baumgartner, G. Dowek (Eds.), Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings, volume 5195 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 475–490. URL: https://doi.org/10.1007/978-3-540-71070-7_40. doi:10.1007/978-3-540-71070-7_40.

[34] A. Voronkov, AVATAR: the architecture for first-order theorem provers, in: A. Biere, R. Bloem (Eds.), Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings, volume 8559 of *Lecture Notes in Computer Science*, Springer, 2014, pp. 696–710. URL: https://doi.org/10.1007/978-3-319-08867-9_46. doi:10.1007/978-3-319-08867-9_46.