A Generalization of Shostak's Method for Combining Decision Procedures

Clark W. Barrett, David L. Dill, and Aaron Stump

Stanford University, Stanford, CA 94305, USA,
http://verify.stanford.edu

Abstract. Consider the problem of determining whether a quantifier-free formula ϕ is satisfiable in some first-order theory $\mathcal{T}$. Shostak's algorithm decides this problem for a certain class of theories with both interpreted and uninterpreted function symbols. We present two new algorithms based on Shostak's method. The first is a simple subset of Shostak's algorithm for the same class of theories but without uninterpreted function symbols. This simplified algorithm is easy to understand and prove correct, providing insight into how and why Shostak's algorithm works. The simplified algorithm is then used as the foundation for a generalization of Shostak's method based on a variation of the Nelson-Oppen method for combining theories.

1 Introduction

In 1984, Shostak introduced a clever and subtle algorithm which decides the satisfiability of quantifier-free formulas in a combined theory which includes a first-order theory (or combination of first-order theories) with certain properties and the theory of equality with uninterpreted function symbols [12]. But despite the fact that Shostak's method is less general than its predecessor, the Nelson-Oppen method [8, 9], it has generated considerable interest and is the basis for decision procedures found in several tools, including PVS [10], STeP [4, 6], and SVC [1, 2, 7].

There are several good reasons for this. First of all, it is easier to implement: the Nelson-Oppen method provides a framework for combining decision procedures, but gives no help on how to construct the individual decision procedures. But as we show in the next section, at the core of Shostak's procedure is a simple method for generating decision procedures for a large class of theories. A second reason for the success of Shostak's method is that despite requiring more restrictive conditions in order to accommodate a theory, a wide variety of useful theories have been shown to satisfy these conditions [4, 12]. Finally, empirical studies have shown that Shostak's method is an order of magnitude more efficient than the Nelson-Oppen method [5].

Unfortunately, the original paper is difficult to follow, due in part to the fact that it contains several errors, and despite an ongoing effort to understand and clarify the method [5, 11, 14], it remains difficult to understand.

In this paper, we take a new approach to explaining Shostak's algorithm. We first present a subset of the original algorithm, in particular, the subset which decides formulas *without* uninterpreted function symbols. This algorithm is surprisingly simple and straightforward, and gives considerable insight into how Shostak's algorithm works.

This algorithm then forms the basis for a more general algorithm that lies at an abstraction level somewhere between the general Nelson-Oppen framework and the highly-specialized Shostak procedure. The purpose is to describe an algorithm which is abstract enough that it can be understood and proved correct, but specific enough that it is not hard to see how to specialize it further to recover Shostak's original algorithm. The correctness proof of this algorithm relies on a new variation of the Nelson-Oppen procedure and new theorem which relates *convexity* (a requirement for Shostak) and *stable-infiniteness* (a requirement for Nelson-Oppen).

It is our hope that this exercise will not only shed light on how Shostak's method can be seen as an efficient refinement of the Nelson-Oppen method, but also provide a generalization which can be used to achieve other efficient refinements. Indeed, one such possible refinement is described in the first author's dissertation [3].

In Section 2, below, some preliminary definitions and notation are given. The simple algorithm without uninterpreted function symbols is presented in Section 3. Section 4 reviews the Nelson-Oppen method in preparation for the generalized algorithm which is presented in Section 5. Finally, Section 6 compares our approach to other work on Shostak's algorithm and describes the refinements necessary to recover Shostak's original algorithm.

2 Preliminary Concepts

2.1 Some Notions from Logic

A *theory* is a set of closed formulas. For the purposes of this paper, all theories are assumed to be first-order and to include the axioms of equality. The *signature* of a theory is the set of function, predicate (other than equality), and constant symbols appearing in those sentences. A *literal* is an atomic formula or its negation. To avoid confusion with the logical equality symbol $=$, we use the symbol $\equiv$ to indicate that two logical expressions are syntactically identical.

For a given model, M, a *variable assignment* ρ is a function which assigns to each variable an element of the domain of M. We write $M \models_\rho \phi$ if ϕ is true in the model M with variable assignment ρ. If Φ is a set of formulas, then $M \models_\rho \Phi$ indicates that $M \models_\rho \phi$ for each $\phi \in \Phi$. In general, whenever sets of formulas are used as logical formulas, the intended meaning is the conjunction of the formulas in the set. A formula ϕ is satisfiable if there exists some model M and variable assignment ρ such that $M \models_\rho \phi$. If Γ is a set of formulas and ϕ is a formula, then $\Gamma \models \phi$ means that whenever a model and variable assignment satisfy Γ, they also satisfy ϕ. A set S of literals is *convex* in a theory $\mathcal{T}$ if $\mathcal{T} \cup S$ does not

entail any disjunction of equalities between variables without entailing one of the equalities itself. A theory $\mathcal{T}$ is convex if every set of literals in the language of the theory is convex in $\mathcal{T}$.

2.2 Equations in Solved Form

Definition 1. *A set $\mathcal{S}$ of equations is said to be in* solved form *iff the left-hand side of each equation in $\mathcal{S}$ is a variable which appears only once in $\mathcal{S}$. We refer to the variables which appear only on the left-hand sides as* solitary *variables.*

A set $\mathcal{S}$ of equations in solved form defines an idempotent substitution: the one which replaces each solitary variable with its corresponding right-hand side. If S is an expression or set of expressions, we denote the result of applying this substitution to S by $\mathcal{S}(S)$. Another interesting property of equations in solved form is that the question of whether such a set $\mathcal{S}$ entails some formula ϕ in a theory $\mathcal{T}$ can be answered simply by determining the validity of $\mathcal{S}(\phi)$ in $\mathcal{T}$:

Proposition 1. *If $\mathcal{T}$ is a theory with signature Σ and $\mathcal{S}$ is a set of Σ-equations in solved form, then $\mathcal{T} \cup \mathcal{S} \models \phi$ iff $\mathcal{T} \models \mathcal{S}(\phi)$.*

Proof. Clearly, $\mathcal{T} \cup \mathcal{S} \models \phi$ iff $\mathcal{T} \cup \mathcal{S} \models \mathcal{S}(\phi)$. Thus we need only show that $\mathcal{T} \cup \mathcal{S} \models \mathcal{S}(\phi)$ iff $\mathcal{T} \models \mathcal{S}(\phi)$. The "if" direction is trivial. To show the other direction, assume that $\mathcal{T} \cup \mathcal{S} \models \mathcal{S}(\phi)$. Any model of $\mathcal{T}$ can be made to satisfy $\mathcal{T} \cup \mathcal{S}$ by assigning any value to the non-solitary variables of $\mathcal{S}$, and then choosing the value of each solitary variable to match the value of its corresponding right-hand side. Since none of the solitary variables occur anywhere else in $\mathcal{S}$, this assignment is well-defined and satisfies $\mathcal{S}$. By assumption then, this model and assignment also satisfy $\mathcal{S}(\phi)$, but none of the solitary variables appear in $\mathcal{S}(\phi)$, so the initial arbitrary assignment to non-solitary variables must be sufficient to satisfy $\mathcal{S}(\phi)$. Thus it must be the case that every model of $\mathcal{T}$ satisfies $\mathcal{S}(\phi)$ with every variable assignment. □

Corollary 1. *If $\mathcal{T}$ is a satisfiable theory with signature Σ and $\mathcal{S}$ is a set of Σ-equations in solved form, then $\mathcal{T} \cup \mathcal{S}$ is satisfiable.*

3 Algorithm S1

In this section we present an algorithm, based on a subset of Shostak's algorithm, for deciding satisfiability of quantifier-free formulas in a theory $\mathcal{T}$ which meets certain conditions. We call such a theory a *Shostak* theory.

Definition 2. *A satisfiable theory $\mathcal{T}$ with signature Σ is a* Shostak *theory if the following conditions hold.*

1. *Σ does not contain any predicate symbols.*
2. *$\mathcal{T}$ is convex.*

3. *There exists a* canonizer *canon, a computable function from Σ-terms to Σ-terms, with the property that $\mathcal{T} \models a = b$ iff $canon(a) \equiv canon(b)$.*
4. *There exists a* solver *solve, a computable function from Σ-equations to sets of formulas defined as follows:*
   (a) *If $\mathcal{T} \models a \neq b$, then $solve(a = b) = \{false\}$.*
   (b) *Otherwise, $solve(a = b)$ returns a set $\mathcal{S}$ of equations in solved form such that $\mathcal{T} \models [(a = b) \leftrightarrow \exists \overline{x}.\mathcal{S}]$, where $\overline{x}$ is the set of variables which appear in $\mathcal{S}$ but not in a or b. Each of these variables must be fresh.*

These requirements are slightly different from those given by Shostak and others. These differences are discussed in Section 6 below. In the rest of this section, $\mathcal{T}$ is assumed to be a Shostak theory with signature Σ, canonizer *canon*, and solver *solve*. As we will show, the solver can be used to convert an arbitrary set of equations into a set of equations in solved form. The canonizer is used to determine whether a specific equality is entailed by a set of equations in solved form, as shown by the following proposition.

Proposition 2. *If $\mathcal{S}$ is a set of Σ-equations in solved form, then $\mathcal{T} \cup \mathcal{S} \models a = b$ iff $canon(\mathcal{S}(a)) \equiv canon(\mathcal{S}(b))$.*

Proof. By Proposition 1, $\mathcal{T} \cup \mathcal{S} \models a = b$ iff $\mathcal{T} \models \mathcal{S}(a) = \mathcal{S}(b)$. But $\mathcal{T} \models \mathcal{S}(a) = \mathcal{S}(b)$ iff $canon(\mathcal{S}(a)) \equiv canon(\mathcal{S}(b))$ by the definition of *canon*. □

```
S1(Γ, Δ, canon, solve)
1.  S := ∅;
2.  WHILE Γ ≠ ∅ DO BEGIN
3.    Remove some equality a = b from Γ;
4.    a* := S(a); b* := S(b);
5.    S* := solve(a* = b*);
6.    IF S* = {false} THEN RETURN FALSE;
7.    S := S*(S) ∪ S*;
8.  END
9.  IF canon(S(a)) ≡ canon(S(b)) for some a ≠ b ∈ Δ THEN RETURN FALSE;
10. RETURN TRUE;
```

Fig. 1. Algorithm S1: based on a simple subset of Shostak's algorithm

Algorithm *S1* (shown in Fig. 1) makes use of the properties of a Shostak theory to check the joint satisfiability of an arbitrary set of equalities, Γ, and an arbitrary set of disequalities, Δ, in a Shostak theory with canonizer *canon* and solver *solve*. Since the satisfiability of any quantifier-free formula can be determined by first converting it to disjunctive normal form, it suffices to have a satisfiability procedure for a conjunction of literals. Since Σ contains no predicate symbols, all Σ-literals are either equalities or disequalities. Thus, Algorithm *S1* is sufficient for deciding the satisfiability of quantifier-free Σ-formulas. Termination of the algorithm is trivial since each step terminates and each time line 3 is executed the size of Γ is reduced. The following lemmas are needed before proving correctness.

Lemma 1. *If $\mathcal{T}'$ is a theory, Γ and Θ are sets of formulas, and $\mathcal{S}$ is a set of equations in solved form, then for any formula ϕ, $\mathcal{T}' \cup \Gamma \cup \Theta \cup \mathcal{S} \models \phi$ iff $\mathcal{T}' \cup \Gamma \cup \mathcal{S}(\Theta) \cup \mathcal{S} \models \phi$.*

Proof. Follows trivially from the fact that $\Theta \cup \mathcal{S}$ and $\mathcal{S}(\Theta) \cup \mathcal{S}$ are satisfied by exactly the same models and variable assignments. □

Lemma 2. *If Γ is any set of formulas, then for any formula ϕ, and Σ-terms a and b,*

$$\mathcal{T} \cup \Gamma \cup \{a = b\} \models \phi \text{ iff } \mathcal{T} \cup \Gamma \cup \mathit{solve}(a = b) \models \phi.$$

Proof.
$\Rightarrow$: Given that $\mathcal{T} \cup \Gamma \cup \{a = b\} \models \phi$, suppose that $M \models_\rho \mathcal{T} \cup \Gamma \cup \mathit{solve}(a = b)$. It is easy to see from the definition of *solve* that $M \models_\rho a = b$ and hence by the hypothesis, $M \models_\rho \phi$.
$\Leftarrow$: Given that $\mathcal{T} \cup \Gamma \cup \mathit{solve}(a = b) \models \phi$, suppose that $M \models_\rho \mathcal{T} \cup \Gamma \cup \{a = b\}$. Then, since $\mathcal{T} \models (a = b) \leftrightarrow \exists \overline{x}.\mathit{solve}(a = b)$, there exists a modified assignment ρ^* which assigns values to all the variables in $\overline{x}$ and satisfies $\mathit{solve}(a = b)$ but is otherwise equivalent to ρ. Then, by the hypothesis, $M \models_{\rho^*} \phi$. But the variables in $\overline{x}$ are *new* variables, so they do not appear in ϕ, meaning that changing their values cannot affect whether ϕ is true. Thus, $M \models_\rho \phi$. □

Lemma 3. *If Γ, $\{a = b\}$, and $\mathcal{S}$ are sets of Σ-formulas, with $\mathcal{S}$ in solved form, and if $\mathcal{S}^* = \mathit{solve}(\mathcal{S}(a = b))$ then if $\mathcal{S}^* \neq \{\mathit{false}\}$, then for every formula ϕ, $\mathcal{T} \cup \Gamma \cup \{a = b\} \cup \mathcal{S} \models \phi$ iff $\mathcal{T} \cup \Gamma \cup \mathcal{S}^* \cup \mathcal{S}^*(\mathcal{S}) \models \phi$.*

Proof.

$\mathcal{T} \cup \Gamma \cup \{a = b\} \cup \mathcal{S} \models \phi$	$\Leftrightarrow$	$\mathcal{T} \cup \Gamma \cup \{\mathcal{S}(a = b)\} \cup \mathcal{S} \models \phi$	Lemma 1
	$\Leftrightarrow$	$\mathcal{T} \cup \Gamma \cup \mathcal{S}^* \cup \mathcal{S} \models \phi$	Lemma 2
	$\Leftrightarrow$	$\mathcal{T} \cup \Gamma \cup \mathcal{S}^* \cup \mathcal{S}^*(\mathcal{S}) \models \phi$	Lemma 1

□

Lemma 4. *During the execution of Algorithm S1, $\mathcal{S}$ is always in solved form.*

Proof. Clearly, $\mathcal{S}$ is in solved form initially. Consider one iteration. By construction, a^* and b^* do not contain any of the solitary variables of $\mathcal{S}$, and thus by the definition of *solve*, $\mathcal{S}^*$ doesn't either. Furthermore, if $\mathcal{S}^* = \{\text{false}\}$ then the algorithm terminates at line 6. Thus, at line 7, $\mathcal{S}^*$ must be in solved form. Applying $\mathcal{S}^*$ to $\mathcal{S}$ guarantees that none of the solitary variables of $\mathcal{S}^*$ appear in $\mathcal{S}$, so the new value of $\mathcal{S}$ is also in solved form. □

Lemma 5. *Let Γ_n and $\mathcal{S}_n$ be the values of Γ and $\mathcal{S}$ after the while loop in Algorithm S1 has been executed n times. Then for each n, and any formula ϕ, the following invariant holds: $\mathcal{T} \cup \Gamma_0 \models \phi$ iff $\mathcal{T} \cup \Gamma_n \cup \mathcal{S}_n \models \phi$.*

Proof. The proof is by induction on n. For $n = 0$, the invariant holds trivially. Now suppose the invariant holds for some $k \geq 0$. Consider the next iteration.

$$
\begin{array}{lcll}
\mathcal{T} \cup \Gamma_0 \models \phi & \Leftrightarrow & \mathcal{T} \cup \Gamma_k \cup \mathcal{S}_k \models \phi & \text{Induction Hypothesis} \\
& \Leftrightarrow & \mathcal{T} \cup \Gamma_{k+1} \cup \{a = b\} \cup \mathcal{S}_k \models \phi & \text{Line 3} \\
& \Leftrightarrow & \mathcal{T} \cup \Gamma_{k+1} \cup \mathcal{S}^* \cup \mathcal{S}^*(\mathcal{S}_k) \models \phi & \text{Lemmas 3 and 4} \\
& \Leftrightarrow & \mathcal{T} \cup \Gamma_{k+1} \cup \mathcal{S}_{k+1} \models \phi & \text{Line 7}
\end{array}
$$

□

Now we can show the correctness of Algorithm *S1*.

Theorem 1. *Suppose $\mathcal{T}$ is a Shostak theory with signature Σ, canonizer canon, and solver solve. If Γ is a set of Σ-equalities and Δ is a set of Σ-disequalities, then $\mathcal{T} \cup \Gamma \cup \Delta$ is satisfiable iff* $\texttt{S1}(\Gamma, \Delta, \mathit{canon}, \mathit{solve}) = \texttt{TRUE}$.

Proof. Suppose $\texttt{S1}(\Gamma, \Delta, \mathit{canon}, \mathit{solve}) = \texttt{FALSE}$. If the algorithm terminates at line 9, then, $\mathit{canon}(\mathcal{S}(a)) \equiv \mathit{canon}(\mathcal{S}(b))$ for some $a \neq b \in \Delta$. It follows from Proposition 2 and Lemma 5 that $\mathcal{T} \cup \Gamma \models a = b$, so clearly $\mathcal{T} \cup \Gamma \cup \Delta$ is not satisfiable. The other possibility is that the algorithm terminates at line 6. Suppose the loop has been executed n times and that Γ_n and $\mathcal{S}_n$ are the values of Γ and $\mathcal{S}$ at the end of the last loop. It must be the case that $\mathcal{T} \models a^* \neq b^*$, so $\mathcal{T} \cup \{a^* = b^*\}$ is unsatisfiable. Clearly then, $\mathcal{T} \cup \{a^* = b^*\} \cup \mathcal{S}_n$ is unsatisfiable, so by Lemma 1, $\mathcal{T} \cup \{a = b\} \cup \mathcal{S}_n$ is unsatisfiable. But $\{a = b\}$ is a subset of Γ_n, so $\mathcal{T} \cup \Gamma_n \cup \mathcal{S}_n$ must be unsatisfiable, and thus by Lemma 5, $\mathcal{T} \cup \Gamma$ is unsatisfiable.

Suppose on the other hand that $\texttt{S1}(\Gamma, \Delta, \mathit{canon}, \mathit{solve}) = \texttt{TRUE}$. Then the algorithm terminates at line 10. By Lemma 4, $\mathcal{S}$ is in solved form. Let $\overline{\Delta}$ be the disjunction of equalities equivalent to $\neg(\Delta)$. Since the algorithm does not terminate at line 9, $\mathcal{T} \cup \mathcal{S}$ does not entail any equality in $\overline{\Delta}$. Because $\mathcal{T}$ is convex, it follows that $\mathcal{T} \cup \mathcal{S} \not\models \overline{\Delta}$. Now, since $\mathcal{T} \cup \mathcal{S}$ is satisfiable by Corollary 1, it follows that $\mathcal{T} \cup \mathcal{S} \cup \Delta$ is satisfiable. But by Lemma 5, $\mathcal{T} \cup \Gamma \models \phi$ iff $\mathcal{T} \cup \mathcal{S} \models \phi$, so in particular $\mathcal{T} \cup \mathcal{S} \models \Gamma$. Thus $\mathcal{T} \cup \mathcal{S} \cup \Delta \cup \Gamma$ is satisfiable, and hence $\mathcal{T} \cup \Gamma \cup \Delta$ is satisfiable. □

3.1 An Example

Perhaps the most obvious example of a Shostak theory is the theory of linear rational arithmetic. A simple canonizer for this theory can be obtained by imposing an order on all variables (lexicographic or otherwise), and combining like terms. For example, $\mathit{canon}(z + 3y - x - 5z) \equiv -x + 3y + (-4z)$. Similarly, a solver can be obtained simply by solving for one of the variables in an equation. Consider the following system of equations:

$$
\begin{aligned}
x + 3y - 2z &= 1 \\
x - y - 6z &= 1 \\
2x + 8y - 2z &= 3
\end{aligned}
$$

The following table shows values for Γ, $\mathcal{S}$, $\mathcal{S}(a = b)$, and $\mathcal{S}^*$ at each iteration of Algorithm *S1* starting with $\Gamma = \{x+3y-2z = 1, x-y-6z = 1, 2x+8y-2z = 3\}$:

Γ	$\mathcal{S}$	$\mathcal{S}(a=b)$	$\mathcal{S}^*$
$x+3y-2z=1$ $x-y-6z=1$ $2x+8y-2z=3$	$\emptyset$	$x+3y-2z=1$	$x=1-3y+2z$
$x-y-6z=1$ $2x+8y-2z=3$	$x=1-3y+2z$	$1-3y+2z-y-6z=1$	$y=-z$
$2x+8y-2z=3$	$x=1+5z$ $y=-z$	$2(1+5z)+8(-z)-2z=3$	false

The solver detects an inconsistency when it tries to solve the equation obtained after applying the substitution from $\mathcal{S}$. The solver indicates this by returning {false}, which results in the algorithm returning `FALSE`.

3.2 Combining Shostak Theories

In [12], Shostak claims that two Shostak theories can always be combined to form a new Shostak theory. A canonizer for the combined theory is obtained simply by composing the canonizers from each individual theory. A solver for the combined theory is ostensibly obtained by repeatedly applying the solver for each theory (treating terms in other theories as variables) until a true variable is on the left-hand side of each equation in the solved form. This does in fact work for many theories, providing a simple and efficient method for combining Shostak theories. However, as pointed out in [7] and [11], the construction of the solver as described is not always possible. We do not address this issue here, but mention it as a question which warrants further investigation.

4 The Nelson-Oppen Combination Method

Nelson and Oppen [8, 9] described a method for combining decision procedures for theories which are stably-infinite and have disjoint signatures. A theory $\mathcal{T}$ is stably-infinite if any quantifier-free formula is satisfiable in some model of $\mathcal{T}$ iff it is satisfiable in an infinite model of $\mathcal{T}$. In this section, we assume $\mathcal{T}_1$ and $\mathcal{T}_2$ are two such theories with signatures Σ_1 and Σ_2 respectively (the generalization to more than two theories is straightforward). Furthermore, we let $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$ and $\Sigma = \Sigma_1 \cup \Sigma_2$. The Nelson-Oppen procedure decides the satisfiability in $\mathcal{T}$ of a set Φ of Σ-literals.

4.1 Tinelli and Harandi's Approach

There have been many detailed presentations of the Nelson-Oppen method. Tinelli and Harandi's approach is particularly appealing because it is rigorous and conceptually simple [13]. Here we give a brief review of the method based on their approach. First, a few more definitions are required.

Members of Σ_i, for $i = 1,2$ are called i-symbols. In order to associate all terms with some theory, each variable is also arbitrarily associated with either $\mathcal{T}_1$ or $\mathcal{T}_2$.

A variable is called an i-variable if it is associated with $\mathcal{T}_i$ (note that an i-variable is *not* an i-symbol, as it is not a member of Σ_i). A Σ-term t is an i-term if it is an i-variable, a constant i-symbol, or an application of a functional i-symbol. An i-predicate is an application of a predicate i-symbol. An atomic i-formula is an an i-predicate or an equality whose left term is an i-term. An i-literal is an atomic i-formula or the negation of an atomic i-formula. An occurrence of a j-term t in either a term or a literal is i-alien if $i \neq j$ and all super-terms (if any) of t are i-terms. An i-term or i-literal is *pure* if it contains only i-symbols (i.e. its i-alien sub-terms are all variables).

Given an equivalence relation $\sim$, let $dom_\sim$ be the domain of the relation. We define the following sets of formulas induced by $\sim$:

$$\begin{aligned} E_\sim &= \{x = y \mid x, y \in dom_\sim \text{ and } x \sim y\} \\ D_\sim &= \{x \neq y \mid x, y \in dom_\sim \text{ and } x \not\sim y\} \\ A_\sim &= E_\sim \cup D_\sim. \end{aligned}$$

Let Ar be a set of equalities and disequalities. If $Ar = A_\sim$ for some equivalence relation $\sim$ with domain $\mathcal{V}$, we call Ar an *arrangement* of $\mathcal{V}$.

The first step in determining the satisfiability of Φ is to transform Φ into an equisatisfiable formula $\Phi_1 \wedge \Phi_2$ where Φ_i consists only of pure i-literals as follows. Let ψ be some i-literal in Φ containing a non-variable i-alien j-term t. Replace all occurrences of t in ψ with a new j-variable z and add the equation $z = t$ to Φ. Repeat until every literal in Φ is pure. The literals can then easily be partitioned into Φ_1 and Φ_2. It is easy to see that Φ is satisfiable if and only if $\Phi_1 \wedge \Phi_2$ is satisfiable.

Now, let $\mathcal{V}$ be the set of all variables which appear in both Φ_1 and Φ_2. A simple version of the Nelson-Oppen procedure simply guesses an equivalence relation $\sim$ on $\mathcal{V}$ nondeterministically, and then checks whether $\mathcal{T}_i \cup \Phi_i \cup A_\sim$ is satisfiable. The correctness of the procedure is based on the following theorem from [13].

Theorem 2. *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two stably-infinite, signature-disjoint theories and let Φ_i be a set of pure i-literals for $i = 1, 2$. Let $\mathcal{V}$ be the set of variables which appear in both Φ_1 and Φ_2. Then $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \Phi_1 \cup \Phi_2$ is satisfiable iff there exists an arrangement Ar of $\mathcal{V}$ such that $\mathcal{T}_i \cup \Phi_i \cup Ar$ is satisfiable for $i = 1, 2$.*

4.2 A Variation of the Nelson-Oppen Procedure

The first step in the version of the Nelson-Oppen procedure described above changes the structure and number of literals in Φ. However, it is possible to give a version of the procedure which does not change the literals in Φ by instead treating alien terms as variables. This simplifies the algorithm by eliminating the need for the purification step. But more importantly, this variation is required for the combination of Shostak and Nelson-Oppen described next.

First, we introduce a *purifying* operator which formalizes the notion of treating alien terms as variables. Let v be a mapping from Σ-terms to variables such that for $i = 1, 2$, each i-term t is mapped to a *fresh* i-variable $v(t)$. Then, for

some Σ-formula or Σ-term α, define $\gamma_i(\alpha)$ to be the result of replacing all i-alien occurrences of terms t by $v(t)$. It is easy to see that as a result, $\gamma_i(\alpha)$ is i-pure. Since γ_i simply replaces terms with unique place-holders, it is injective. We will denote its inverse by γ_i^{-1}. We will also denote by $\gamma_0(\alpha)$ the result of replacing each maximal term (i.e. terms without any super-terms) t in α by $v(t)$. Thus, the only terms in $\gamma_0(\alpha)$ are variables.

Our variation on the Nelson-Oppen procedure works as follows. Given a set of literals, Φ, first partition Φ into two sets Φ_1 and Φ_2, where Φ_i is exactly the set of i-literals in Φ. Let $\mathcal{V}$ be the set of all terms which are i-alien (for some i) in some literal in Φ or in some sub-term of some literal in Φ. $\mathcal{V}$ consists of exactly those terms that would end up being replaced by variables in the original Nelson-Oppen method. $\mathcal{V}$ will also be referred to as the set of *shared* terms. As before, an equivalence relation $\sim$ on $\mathcal{V}$ is guessed. If $\mathcal{T}_i \cup \gamma_i(\Phi_i \cup A_\sim)$ is satisfiable for each i, then $\mathcal{T} \cup \Phi$ is satisfiable, as shown by the following theorem.

Theorem 3. *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two stably-infinite, signature-disjoint theories and let Φ be a set of literals in the combined signature Σ. If Φ_i is the set of all i-literals in Φ and $\mathcal{V}$ is the set of shared terms in Φ, then $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \Phi$ is satisfiable iff there exists an equivalence relation $\sim$ on $\mathcal{V}$ such that for $i = 1, 2$, $\mathcal{T}_i \cup \gamma_i(\Phi_i \cup A_\sim)$ is satisfiable.*

Proof.
$\Rightarrow$: Suppose $M \models_\rho \mathcal{T} \cup \Phi$. Let $a \sim b$ iff $a, b \in \mathcal{V}$ and $M \models_\rho a = b$. Then clearly for $i = 1, 2$, $M \models_\rho \mathcal{T}_i \cup \Phi_i \cup A_\sim$. It is then easy to see that $\mathcal{T}_i \cup \gamma_i(\Phi_i \cup A_\sim)$ is satisfiable by choosing a variable assignment which assigns to each variable $v(t)$ the corresponding value of the term t which it replaces.
$\Leftarrow$: Suppose that for each i, $T_i \cup \gamma_i(\Phi_i \cup A_\sim)$ is satisfiable. Consider $i = 1$. Let Θ_1 be the set of all equations $v(t) = t$, where $t \in \mathcal{V}$ is a 1-term. Consider $\gamma_1(\Theta_1)$. Since γ_1 never replaces 1-terms and each $v(t)$ is a new variable, it follows that $\gamma_1(\Theta_1)$ is in solved form, and its solitary variables are exactly the variables which are used to replace 1-terms. Thus, by Corollary 1, $\mathcal{T}_1 \cup \gamma_1(\Theta_1)$ is satisfiable. Furthermore, since none of the solitary variables of $\gamma_1(\Theta_1)$ appear in $\gamma_1(\Phi_1 \cup A_\sim)$, a satisfiable assignment for $\mathcal{T}_1 \cup \gamma_1(\Theta_1)$ can be constructed from the satisfying assignment for $\mathcal{T}_1 \cup \gamma_1(\Phi_1 \cup A_\sim)$ (which exists by hypothesis) so that the resulting assignment satisfies $\mathcal{T}_1 \cup \gamma_1(\Phi_1 \cup A_\sim \cup \Theta_1)$. Now, each term in $\gamma_1(A_\sim)$ is the right-hand side of some equation in $\gamma_1(\Theta_1)$, so by repeatedly applying equations from $\gamma_1(\Theta_1)$ as substitutions, $\gamma_1(A_\sim)$ can be transformed into $\gamma_0(A_\sim)$, and thus $\mathcal{T}_1 \cup \gamma_1(\Phi_1 \cup \Theta_1) \cup \gamma_0(A_\sim)$ must also be satisfiable. Applying the same argument with i=2, we conclude that $\mathcal{T}_2 \cup \gamma_2(\Phi_2 \cup \Theta_2) \cup \gamma_0(A_\sim)$ is satisfiable. But for each i, $\gamma_i(\Phi_i \cup \Theta_i)$ is a set of i-literals. Furthermore, $\gamma_0(A_\sim)$ is an arrangement of the variables shared by these two sets, so Theorem 2 can be applied to conclude that $\mathcal{T} \cup \Phi \cup \Theta_1 \cup \Theta_2$, and thus $\mathcal{T} \cup \Phi$, is satisfiable. □

5 Combining the methods

Let $\mathcal{T}_1, \mathcal{T}_2, \Sigma_1, \Sigma_2, \mathcal{T}$, and Σ be defined as in the previous section, with the additional assumptions that $\mathcal{T}_1$ is a Shostak theory and that neither $\mathcal{T}_1$ nor $\mathcal{T}_2$

admits trivial models (typically, theories of interest do not admit trivial models, or can be easily modified so that this is the case). The following theorem shows that both theories are also stably-infinite.

Theorem 4. *Every convex first-order theory with no trivial models is stably-infinite.*

Proof. Suppose $\mathcal{U}$ is a first-order theory which is not stably-infinite. Then there exists some quantifier-free set of literals Φ which is satisfiable in a finite model of $\mathcal{U}$, but not in an infinite model of $\mathcal{U}$. Let $\exists\overline{x}\Phi$ be the existential closure of Φ. Then $\exists\overline{x}.\Phi$ is true in some finite model, but not in any infinite model, of $\mathcal{U}$. It follows that $\mathcal{U} \cup \{\exists\overline{x}.\Phi\}$ is a theory with no infinite models. By first-order compactness, there must be some finite cardinality n such that there is a model of $\mathcal{U} \cup \{\exists\overline{x}.\Phi\}$ of cardinality n, but none of cardinality larger than n. Clearly, $\mathcal{U} \cup \Phi$ is satisfiable in some model of size n, but not in any models larger than n. It follows by the pigeonhole principle that if $y_i, 0 \leq i \leq n$ are fresh variables, then $\mathcal{U} \cup \Phi \models \bigvee_{i \neq j} y_i = y_j$, but because $\mathcal{U}$ has no trivial models (i.e. models of size 1), $\mathcal{U} \cup \Phi \not\models y_i = y_j$ for any i, j with $i \neq j$. Thus, $\mathcal{U}$ is not convex. □

5.1 The Combined Algorithm

Suppose Φ is a set of Σ-literals. As in Section 4.2, divide Φ into Φ_1 and Φ_2 where Φ_i contains exactly the i-literals of Φ. Let $\mathcal{V}$ be the set of shared terms. By Theorem 3, $\mathcal{T} \cup \Phi_1 \cup \Phi_2$ is satisfiable iff there exists an equivalence relation $\sim$ such that for $i = 1, 2$, $\mathcal{T}_i \cup \gamma_i(\Phi_i \cup A_\sim)$ is satisfiable.

In order for the approach in Algorithm *S1* to function in a multiple-theory environment, it is necessary to generalize the definition of equations in solved form to accommodate the notion of treating alien terms as variables. A set $\mathcal{S}$ of equations is said to be in *i-solved form* if $\gamma_i(\mathcal{S})$ is in solved form. If $\mathcal{S}$ is a set of equations in i-solved form and $\mathcal{V}$ is an expression or set of expressions in a mixed language including Σ_i, then we define $\mathcal{S}(\mathcal{V})$ to be the result of replacing each left-hand side in $\mathcal{S}$ which occurs as an i-alien in $\mathcal{V}$ with the corresponding right-hand side. Formally, $\mathcal{S}(\mathcal{V})$ is redefined to be $\gamma_i^{-1}(\gamma_i(\mathcal{S})(\gamma_i(\mathcal{V})))$, i.e. the application of $\mathcal{S}$ to $\mathcal{V}$ should be equivalent to first replacing all i-alien terms with variables in both $\mathcal{S}$ and $\mathcal{V}$, then doing the substitution, and then finally restoring the i-alien terms to their places. We similarly need to extend the definitions of *canon* and *solve*. Let $\mathit{canon}(\alpha)$ denote $\gamma_1^{-1}(\mathit{canon}(\gamma_1(\alpha)))$ and $\mathit{solve}(\beta)$ denote $\gamma_1^{-1}(\mathit{solve}(\gamma_1(\beta)))$.

Now, let Γ be the set of equalities in Φ_1 and Δ the set of disequalities in Φ_1. Furthermore, let Sat_2 be a decision procedure for satisfiability of literals in $\mathcal{T}_2$:

$$Sat_2(\Phi) = \texttt{TRUE} \;\; \textit{iff} \;\; \mathcal{T}_2 \cup \gamma_2(\Phi) \not\models \text{false}.$$

Algorithm *S2* is a modification of Algorithm *S1* which accommodates the additional theory $\mathcal{T}_2$. Essentially, the algorithm is identical except for the addition of lines 3 through 5 which check whether Φ_2 is consistent in theory $\mathcal{T}_2$ with an arrangement $A_\sim$. The equivalence relation $\sim$ on $\mathcal{V}$ is derived from $\mathcal{S}$ as follows:

```
S2(Γ, Δ, canon, solve, Φ2, Sat2)
1.  S := ∅;
2.  WHILE Γ = ∅ OR ¬Sat2(Φ2 ∪ A∼) DO BEGIN
3.    IF ¬Sat2(Φ2 ∪ A∼) THEN BEGIN
4.      IF ¬Sat2(Φ2 ∪ E∼) THEN RETURN FALSE;
5.      ELSE Choose a ≠ b ∈ D∼ such that ¬Sat2(Φ2 ∪ E∼ ∪ {a ≠ b});
6.    END ELSE Remove some equality a = b from Γ;
7.    a* := S(a); b* := S(b);
8.    S* := solve(a* = b*);
9.    IF S* = {false} THEN RETURN FALSE;
10.   S := S*(S) ∪ S*;
11. END
12. IF a∼b for some a ≠ b ∈ Δ THEN RETURN FALSE;
13. RETURN TRUE;
```

Fig. 2. Algorithm S2: a generalization of Shostak's algorithm

$$a \sim b \quad \textit{iff} \quad a, b \in \mathcal{V} \text{ and } \textit{canon}(\mathcal{S}(a)) \equiv \textit{canon}(\mathcal{S}(b))$$

In each iteration of the while loop, an equation is processed and integrated with $\mathcal{S}$. This equation is either the result of the current arrangement being inconsistent in $\mathcal{T}_2$ (lines 3 through 5) or simply an equation from Γ (line 6). As shown below, the definition of $\sim$ ensures that $\mathcal{S}$ is consistent with $A_\sim$. Similarly, equations are added to $\mathcal{S}$ until $A_\sim$ is also consistent with Φ_2. Thus, when the algorithm returns TRUE, both Φ_1 and Φ_2 are known to be consistent with the arrangement $A_\sim$. Line 5 requires a little explanation. If the algorithm reaches line 5, it means that $\Phi_2 \cup E_\sim \cup D_\sim$ is not satisfiable in $\mathcal{T}_2$, but $\Phi_2 \cup E_\sim$ is. It follows from convexity of $\mathcal{T}_2$ that there must be a disequality $a \neq b$ in $D_\sim$ such that $\Phi_2 \cup E_\sim \cup \{a \neq b\}$ is not satisfiable in $\mathcal{T}_2$.

Algorithm *S2* terminates because each step terminates and in each iteration either the size of Γ is reduced by one or two equivalence classes in $\sim$ are merged. As before, the correctness proof requires a couple of preparatory lemmas.

Lemma 6. *Suppose $\mathcal{S}$ is a set of Σ-formulas in 1-solved form, $\mathcal{V}$ is a set of Σ-terms, and $\sim$ is defined as above. If $\approx$ is an equivalence relation on $\mathcal{V}$ such that $\mathcal{T}_1 \cup \gamma_1(A_\approx \cup \mathcal{S})$ is satisfiable, then $E_\sim \subseteq A_\approx$. In other words, every arrangement of $\mathcal{V}$ consistent with $\mathcal{S}$ must include $E_\sim$.*

Proof. Consider an arbitrary equation $a = b$ between terms in $\mathcal{V}$. $a = b \in E_\sim$ iff $\textit{canon}(\mathcal{S}(a)) \equiv \textit{canon}(\mathcal{S}(b))$ iff (by Proposition 2) $\mathcal{T}_1 \cup \gamma_1(\mathcal{S}) \models \gamma_1(a = b)$. So $\gamma_1(a = b)$ must be true in every model and assignment satisfying $\mathcal{T}_1 \cup \gamma_1(\mathcal{S})$. In particular, if $\mathcal{T}_1 \cup \gamma_1(A_\approx \cup \mathcal{S})$ is satisfiable, the corresponding model and assignment must also satisfy $\gamma_1(a = b)$. Since either the equation $a = b$ or the disequation $a \neq b$ must be in $A_\approx$, it must be the case that $a = b \in A_\approx$. Thus, $E_\sim \subseteq A_\approx$. □

Lemma 7. *Let Γ_n and $\mathcal{S}_n$ be the values of Γ and $\mathcal{S}$ after the loop in Algorithm S2 has been executed n times. Then for each n, the following invariant holds: $\mathcal{T} \cup \Phi$ is satisfiable iff there exists an equivalence relation $\approx$ on $\mathcal{V}$ such that*

(1) $\mathcal{T}_1 \cup \gamma_1(\Gamma_n \cup \Delta \cup A_\approx \cup \mathcal{S}_n)$ is satisfiable, and
(2) $\mathcal{T}_2 \cup \gamma_2(\Phi_2 \cup A_\approx)$ is satisfiable.

Proof. The proof is by induction on n. For the base case, notice that by Theorem 3, $\mathcal{T} \cup \Phi$ is satisfiable iff there exists an equivalence relation $\approx$ such that (1) and (2) hold with $n = 0$.

Before doing the induction case, we first show that for some fixed equivalence relation $\approx$, (1) and (2) hold when $n = k$ iff (1) and (2) hold when $n = k + 1$. Notice that (2) is independent of n, so it is only necessary to consider (1). There are two cases to consider.

First, suppose that the condition of line 3 is true and line 5 is executed. We first show that (1) holds when $n = k$ iff the following holds:

(3) $\mathcal{T}_1 \cup \gamma_1(\Gamma_{k+1} \cup \Delta \cup A_\approx \cup \{a = b\} \cup \mathcal{S}_k)$ is satisfiable.

Since line 6 is not executed, $\Gamma_{k+1} = \Gamma_k$. The if direction is then trivial since the formula in (1) is a subset of the formula in (3). To show the only if direction, first note that it follows from line 5 that $\mathcal{T}_2 \cup \gamma_2(\Phi_2 \cup E_\sim) \models \gamma_2(a = b)$. But by Lemma 6, $E_\sim \subseteq A_\approx$, so it follows that $\mathcal{T}_2 \cup \gamma_2(\Phi_2 \cup A_\approx) \models \gamma_2(a = b)$. Since either $a = b \in A_\approx$ or $a \neq b \in A_\approx$, it must be the case that $a = b \in A_\approx$ and thus (3) follows trivially from (1). Now, by Lemma 3 (where ϕ is false), if line 10 is reached, then (3) holds iff

(4) $\mathcal{T}_1 \cup \gamma_1(\Gamma_{k+1} \cup \Delta \cup A_\approx \cup \mathcal{S}^*(\mathcal{S}_k) \cup \mathcal{S}^*)$ is satisfiable,

where $\mathcal{S}^* = \mathit{solve}(\mathcal{S}(a = b))$. But $\mathcal{S}_{k+1} = \mathcal{S}^*(\mathcal{S}_k) \cup \mathcal{S}^*$, so (4) is equivalent to (1) with $n = k + 1$.

In the other case, line 6 is executed (so that $\Gamma_{k+1} = \Gamma_k - \{a = b\}$). Thus, (1) holds with $n = k$ iff $\mathcal{T}_1 \cup \gamma_1(\Gamma_{k+1} \cup \Delta \cup \{a = b\} \cup A_\approx \cup \mathcal{S}_k)$ is satisfiable, which is equivalent to (3). As in the previous case, it then follows from Lemma 3 that (1) holds at k iff (1) holds at $k + 1$.

Thus, given an equivalence relation, (1) and (2) hold at $k + 1$ exactly when they hold at k. It follows easily that if an equivalence relation exists which satisfies (1) and (2) at k, then there exists an equivalence relation satisfying (1) and (2) at $k + 1$ and vice-versa. Finally, the induction case assumes that that $\mathcal{T} \cup \Phi$ is satisfiable iff there exists an equivalence relation $\approx$ such that (1) and (2) hold at k. It follows from the above argument that $\mathcal{T} \cup \Phi$ is satisfiable iff there exists an equivalence relation $\approx$ such that (1) and (2) hold at $k + 1$. □

Theorem 5. *Suppose that $\mathcal{T}_1$ is a Shostak theory with signature Σ_1, canonizer canon, and solver solve, and that $\mathcal{T}_2$ is a convex theory with signature Σ_2 disjoint from Σ_1 and satisfiability procedure Sat_2. Suppose also that neither $\mathcal{T}_1$ nor $\mathcal{T}_2$ admit trivial models, and let $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$ and $\Sigma = \Sigma_1 \cup \Sigma_2$. Suppose Φ is a set of*

Σ-literals. Let Γ be the subset of Φ which consists of 1*-equalities, Δ the subset of Φ which consists of* 1*-disequalities, and Φ_2 the remainder of the literals in Φ. $\mathcal{T} \cup \Phi$ is satisfiable iff* $\texttt{S2}(\Gamma, \Delta, canon, solve, \Phi_2, Sat_2) = \texttt{TRUE}$.

Proof. First note that by the same argument used in Lemma 4, $\mathcal{S}$ is always in 1-solved form.

Suppose $\texttt{S2}(\Gamma, \Delta, canon, solve, \Phi_2, Sat_2) = \texttt{FALSE}$. If the algorithm terminates at line 9 or 12, then the proof that Φ is unsatisfiable is the same as that for Algorithm *S1* above. If it stops at line 4, then suppose there is an equivalence relation $\approx$ satisfying condition (1) of Lemma 7. It follows from Lemma 6 that $E_\sim \subseteq A_\approx$. But since the algorithm terminates at line 4, $\mathcal{T}_2 \cup \gamma_2(\Phi_2 \cup A_\approx)$ must be unsatisfiable. Thus condition (2) of Lemma 7 cannot hold. Thus, by Lemma 7, $\mathcal{T} \cup \Phi$ is unsatisfiable.

Suppose on the other hand that $\texttt{S2}(\Gamma, \Delta, canon, solve, \Phi_2, Sat_2) = \texttt{TRUE}$. By the definition of $\sim$ and Proposition 2, $a = b \in A_\sim$ iff $\mathcal{T}_1 \cup \gamma_1(\mathcal{S}) \models \gamma_1(a = b)$. It follows from the convexity of $\mathcal{T}_1$ and Corollary 1 that $\mathcal{T}_1 \cup \gamma_1(\mathcal{S} \cup A_\sim)$ is satisfiable. It then follows from the fact that *S2* does not terminate at line 12 (as well as convexity again) that $\mathcal{T}_1 \cup \gamma_1(\mathcal{S} \cup \Delta \cup A_\sim)$ is satisfiable. This is condition (1) of Lemma 7. Condition (2) must hold because the while loop terminates. Thus, by Lemma 7, $\mathcal{T} \cup \Phi$ is satisfiable. □

6 A Comparison with Shostak's Original Method

There are two main ways in which this work differs from Shostak's original method, which is best represented by Ruess and Shankar in [11]. The first is in the set of requirements a theory must fulfill. The second is in the level of abstraction at which the algorithm is presented.

6.1 Requirements on the Theory

Of the four requirements given in our definition of a Shostak theory, the first two are clarifications which are either assumed or not addressed in other work, and the last two are similar to, but slightly less restrictive, than the requirements listed by others. The first requirement is simply that the theory contain no predicate symbols. This is a minor point that is included simply to be explicit about an assumption which is implicit in other work. Shostak's method does not give any guidance on what to do if a theory includes predicate symbols. One possible approach is to encode predicates as functions, but this only works if the resulting encoding admits a canonizer and solver.

The second requirement is that the theory be convex. This may seem overly restrictive since Shostak claims that non-convex theories can be handled [12]. Consider, however, the following simple non-convex theory with signature $\{a, b\}$: $\{a \neq b, \forall x.(x = a \vee x = b)\}$. It is easy to see that this theory admits a (trivial) canonizer and a solver. However, for the unsatisfiable set of formulas $\{x \neq y, y \neq z, x \neq z\}$, any version of Shostak's algorithm will fail to detect the inconsistency.

Ruess and Shankar avoid this difficulty by restricting their attention to the problem of whether $\mathcal{T} \cup \Gamma \models a = b$ for some set of equalities Γ. However, the ability to solve this problem does not lead to a self-contained decision procedure unless the theory is convex.

The third requirement on the theory is that a canonizer exist. Shostak gave several additional properties that must be satisfied by the canonizer. These are not needed at the level of abstraction of our algorithms, though some efficient implementations may require the additional properties.

A similar situation arises with the requirements on the solver: only a subset of the original requirements are needed. Note that although we require the set of equalities returned by the solver to be equisatisfiable with the input set in *every* model of $\mathcal{T}$, whereas Ruess and Shankar require only that it be equisatisfiable with the input set in every σ-model[1], it is not difficult to show that their requirements on the canonizer imply that every model of $\mathcal{T}$ must be a σ-model.

6.2 Level of Abstraction

Algorithm *S2* looks very different from Shostak's original published algorithm as well as most other published versions, though these are, in fact, closely related. An algorithm equivalent to that found in [11] can be obtained by making a number of refinements. We do not have the space to describe these in detail, but we outline them briefly below. We also describe some general principles they exemplify which could be used in other refinements.

The most obvious refinement is to replace $\mathcal{T}_2$ by the theory of equality with uninterpreted function symbols. The data structure for $\mathcal{S}$ can be expanded to include all equations (not just the 1-equations), obviating the need to track Φ_2 separately. The check for satisfiability in $\mathcal{T}_2$ is replaced by a simple check for congruence closure over the terms in $\mathcal{S}$. The general principle here is that if $\mathcal{S}$ can be expanded to track the equalities in another theory, then equality information only needs to be maintained in one place, which is more efficient.

Another refinement is that a more sophisticated substitution can be applied at line 7 of Algorithm *S2*. The more sophisticated substitution considers each sub-term t, and if it is known to be equivalent to a term u already appearing in $\mathcal{S}$, then all instances of t are replaced with u. For terms in the Shostak theory, this is essentially accomplished by applying the canonizer. For uninterpreted function terms, it is a bit more subtle. For example, if $x = y \in \mathcal{S}$ and $f(x)$ appears in $\mathcal{S}$, then if $f(y)$ is encountered, it can be replaced by $f(x)$. As a result, fewer total terms are generated and thus fewer terms need to be considered when updating $\mathcal{S}$ or when performing congruence closure. The general principle is that simplifications and substitutions which reduce the total number of terms can improve efficiency. This is especially important in a natural generalization of Algorithm *S2* to accommodate non-convex theories in which the search for an appropriate arrangement of the shared terms can take time which is more than exponential in the number of shared terms [9].

[1] In the notation of Ruess and Shankar, the canonizer is denoted by σ, and a *σ-model* M is one where $M \models a = \sigma(a)$ for any term a.

Acknowledgments

We are especially grateful to Natarajan Shankar for many helpful and productive conversations regarding Shostak's method. We would also like to thank Cesare Tinelli and the anonymous referees who provided important corrections and valuable feedback. This work was partially supported by the National Science Foundation Grant CCR-9806889, and the DARPA PCES program (DARPA/AirForce contract number F33615-00-C-1693).

References

1. C. Barrett, D. Dill, and J. Levitt. Validity Checking for Combinations of Theories with Equality. In M. Srivas and A. Camilleri, editors, *Formal Methods in Computer-Aided Design*, volume 1166 of *Lecture Notes in Computer Science*, pages 187–201. Springer-Verlag, 1996.
2. C. Barrett, D. Dill, and A. Stump. A Framework for Cooperating Decision Procedures. In *17th International Conference on Automated Deduction*, Lecture Notes in Computer Science. Springer-Verlag, 2000.
3. Clark W. Barrett. *Checking Validity of Quantifier-Free Formulas in Combinations of First-Order Theories.* PhD thesis, Stanford University, 2002.
4. Nikolaj S. Bjørner. *Integrating Decision Procedures for Temporal Verification.* PhD thesis, Stanford University, 1999.
5. D. Cyrluk, P. Lincoln, and N. Shankar. On Shostak's Decision Procedure for Combinations of Theories. In M. McRobbie and J. Slaney, editors, *13th International Conference on Computer Aided Deduction*, volume 1104 of *Lecture Notes in Computer Science*, pages 463–477. Springer-Verlag, 1996.
6. Z. Manna et al. STeP: Deductive-Algorithmic Verification of Reactive and Real-time Systems. In *8th International Conference on Computer-Aided Verification*, volume 1102 of *Lecture Notes in Computer Science*, pages 415–418. Springer-Verlag, 1996.
7. Jeremy R. Levitt. *Formal Verification Techniques for Digital Systems.* PhD thesis, Stanford University, 1999.
8. G. Nelson and D. Oppen. Simplification by Cooperating Decision Procedures. *ACM Transactions on Programming Languages and Systems*, 1(2):245–57, 1979.
9. Derek C. Oppen. Complexity, Convexity and Combinations of Theories. *Theoretical Computer Science*, 12:291–302, 1980.
10. S. Owre, J. Rushby, and N. Shankar. PVS: A Prototype Verification System. In D. Kapur, editor, *11th International Conference on Automated Deduction*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752. Springer-Verlag, 1992.
11. H. Ruess and N. Shankar. Deconstructing Shostak. In *16th Annual IEEE Symposium on Logic in Computer Science*, pages 19–28, June 2001.
12. Robert E. Shostak. Deciding Combinations of Theories. *Journal of the Association for Computing Machinery*, 31(1):1–12, 1984.
13. C. Tinelli and M. Harandi. A New Correctness Proof of the Nelson-Oppen Combination Procedure. In F. Baader and K. Schulz, editors, *1st International Workshop on Frontiers of Combining Systems (FroCoS'96)*, volume 3 of *Applied Logic Series*. Kluwer Academic Publishers, 1996.
14. Ashish Tiwari. *Decision Procedures in Automated Deduction.* PhD thesis, State University of New York at Stony Brook, 2000.