Strategies for Combining Decision Procedures*

Sylvain Conchon[1] and Sava Krstić[2]

[1] École des Mines de Nantes
[2] OGI School of Science & Engineering at Oregon Health & Sciences University

Abstract. Implementing efficient algorithms for combining decision procedures has been a challenge and their correctness precarious. In this paper we describe an inference system that has the classical Nelson-Oppen procedure at its core and includes several optimizations: variable abstraction with sharing, canonization of terms at the theory level, and Shostak's streamlined generation of new equalities for theories with solvers. The transitions of our system are fine-grained enough to model most of the mechanisms currently used in designing combination procedures. In particular, with a simple language of regular expressions we are able to describe several combination algorithms as strategies for our inference system, from the basic Nelson-Oppen to the very highly optimized one recently given by Shankar and Rueß. Presenting the basic system at a high level of generality and nondeterminism allows transparent correctness proofs that can be extended in a modular fashion whenever a new feature is introduced in the system. Similarly, the correctness proof of any new strategy requires only minimal additional proof effort.

1 Introduction

Efficient decision procedures exist for many first-order theories commonly occurring in modeling practice. Linear arithmetic, the pure theory of equality, and theories associated with algebraic datatypes are some examples. Since the interesting properties are often expressed by formulas involving symbols from more than one theory, what one really needs is the integration of these "little engines of proof" into a single efficient tool [12]. Several such systems have been designed [5,15] and used in a variety of applications: general purpose theorem provers, static analysis, extended type checking, hardware verification, etc.

The promise of combination provers is great, but their actual use is still limited and their design is in the state of active research and experimentation. The basic design principles have been set down in the landmark papers of Nelson and Oppen [8] and Shostak [14]. Nelson and Oppen described and proved a general combination algorithm, and Shostak offered an apparently more efficient algorithm, but of restricted scope. What exactly the scope of Shostak's method is has remained unclear for a long time, and it took twenty years to obtain the first correct versions of his algorithm [13,3,6].

* The research reported in this paper was supported by the NSF Grant CCR-9703218. It was performed while S. Conchon was with OGI School of Science & Engineering.

On the other hand, correctness of the Nelson and Oppen framework has not been a concern; a pleasing high-level proof is given by Tinelli and Harandi [16]. Correctness becomes a concern, however, as soon as we attempt to describe this framework at a lower level that explicates important implementation features, or to incorporate Shostak's algorithm into it.

Our goal in this paper is to describe the Nelson-Oppen framework at a level that is high enough to enjoy a simple correctness proof (based on the theorem of Tinelli and Harandi), and low enough to incorporate crucial optimizations, like variable abstraction with sharing, theory state normalization, and deduction by lookup.

Our system is described in Section 3 by a set of transformation rules which can be applied in arbitrary order. The generality and nondeterminism expose only the essential parts of the system and allow for simple correctness proofs. They also give us great flexibility to restrict the system further without needing to reprove most of the necessary correctness facts. We demonstrate this in Section 4, by expressing several interesting strategies with a simple language of regular expressions and proving their correctness with only a little extra effort.

We believe we have described the essence of Shostak's method by the rules we present in Section 5. The rules capture the inference pattern that is possible for the so-called *Shostak theories* and that allows these theories to "cooperate" in the Nelson-Oppen framework more efficiently than by using a generic search-and-backtrack mechanism. With these rules added to our inference system, it becomes possible to express complex algorithms, and we show in Section 6 a regular expression that (as a strategy) quite accurately describes the recent algorithm of Shankar and Rueß. The algorithm combines decision procedures of several Shostak theories and is the most detailed algorithm of this kind whose correctness has been proved [13].

2 Notations and Conventions

This section contains the notation and conventions used throughout the paper.

Given a first-order signature Σ and a fixed countable set X of variables, we will denote by $T_\Sigma(X)$ the set of terms constructed over Σ and X. We will use the symbols a, b to denote terms and x, y, z to denote variables. Viewing terms as trees, subterms within a given term a are identified by their positions. Given a position π, a_π denotes the subterm of a at position π, and $a[\pi \mapsto b]$ the term obtained by the *replacement* of a_π by the term b.

For simplicity we will consider only signatures without predicate symbols. *Literals* are thus equations $a \approx b$ between terms over Σ, and disequations $\neg(a \approx b)$ that will be written as $a \not\approx b$. *Formulas* over Σ are built from literals using the standard logical connectives. Sets of formulas are viewed as conjunctions of their elements.

We will write $a \bowtie b$ for a general literal (equation or disequation). If a and b are variables, we say that this literal is *simple*. Sets of simple equations are called *queries* and sets of disjunctions of simple equations are *answers*.

As usual, we say that a formula Φ over Σ is *satisfiable* (resp. *valid*) if it holds for some (resp. all) Σ-models and variable assignments. A *theory* is a satisfiable set of closed formulas over some signature Σ. If $\mathcal{T}$ and Φ are, respectively, a theory and a formula over Σ, we say that Φ is $\mathcal{T}$*-satisfiable* if $\mathcal{T} \cup \{\Phi\}$ is satisfiable. The entailment notation $\mathcal{T}, \Gamma \models \Phi$ means that the implication $\Gamma \longrightarrow \Phi$ holds in all models of $\mathcal{T}$ and for all variable assignments. A *decision procedure* for a theory $\mathcal{T}$ is an algorithm that decides for a given quantifier-free formula Φ whether $\mathcal{T} \models \Phi$ or not. As is well known, having a decision procedure for a theory amounts to having an algorithm that checks satisfiability of sets of literals.

A theory $\mathcal{T}$ is *stably-infinite* if every quantifier-free formula satisfiable in some model of $\mathcal{T}$ is also satisfiable in an infinite model of $\mathcal{T}$. All theories in this paper will be stably-infinite by assumption.

Two theories $\mathcal{T}_1$ and $\mathcal{T}_2$ are *disjoint* if they are defined over two disjoint signatures Σ_1 and Σ_2. We will use the notation $\mathcal{T}_1 + \mathcal{T}_2$ for the union of disjoint theories. Terms over $\Sigma_1 + \Sigma_2$ are usually called *mixed*; a mixed term is a *pure* i-term if its symbols are all in Σ_i.

3 The Equality Propagation Procedure

We present in this section an abstract version of the equality propagation procedure of Nelson and Oppen [8]. It combines decision procedures of disjoint stably-infinite theories into a single decision procedure for the union theory.

3.1 Abstract Combination Procedure

Let $\mathcal{T}_0, \ldots, \mathcal{T}_n$ be disjoint stably-infinite theories and $\mathcal{T} = \mathcal{T}_0 + \cdots + \mathcal{T}_n$ the combined theory. In the following, we will use the term *satisfiable* to mean $\mathcal{T}$-satisfiable.

We define the operation of our abstract procedure by a set of inference rules, shown in Figure 1. The rules describe the evolution of the state of the procedure, represented as a *configuration* $\langle V \mathbin{[\!]} \Delta \mathbin{[\!]} \Gamma \mathbin{[\!]} \Phi_0, \ldots, \Phi_n \rangle$, where Γ is a set of literals over $\mathcal{T}$, Δ is a set of disjunctions of simple literals, each Φ_i is a set of equations of the form $x \approx a$ where a is an i-term, and V is a set of variables containing those occurring in Γ and Δ. (The set V is redundant, but convenient for bookkeeping purposes.) We also use the symbol $\perp$ as a configuration, and call a configuration *proper* if it is not $\perp$. The aim of our inference system is to determine satisfiability of configurations, formally defined as follows.

Definition 1 (Satisfiability). *A configuration* $\langle V \mathbin{[\!]} \Delta \mathbin{[\!]} \Gamma \mathbin{[\!]} \Phi_0, \ldots, \Phi_n \rangle$ *is* satisfiable *if the formula* $\Gamma \wedge \Phi_0 \wedge \cdots \wedge \Phi_n \wedge \Delta$ *is satisfiable. The configuration* $\perp$ *is not satisfiable.*

We say that a configuration $\mathcal{C}$ *reduces* to a configuration $\mathcal{C}'$, written $\mathcal{C} \Rightarrow \mathcal{C}'$, if $\mathcal{C}$ can be transformed into $\mathcal{C}'$ by applying one of the inference rules. Configurations that allow no reductions will be called *irreducible.*

Satisfiability of any set Γ of literals over $\mathcal{T}$ is clearly equivalent to the satisfiability of the corresponding *initial configuration* $\mathcal{C}_\Gamma = \langle V \mathrel{[\!]} \emptyset \mathrel{[\!]} \Gamma \mathrel{[\!]} \emptyset \rangle$, where V is the set of variables in Γ. With this interpretation of Γ as a configuration, and in view of the following theorem, our inference system is indeed a nondeterministic decision procedure for $\mathcal{T}$.

$$(\textbf{Ab})\textbf{stract}_i \quad \frac{\langle V \mathrel{[\!]} \Delta \mathrel{[\!]} \Gamma \uplus \{a \bowtie b\} \mathrel{[\!]} \dots, \Phi_i, \dots\rangle}{\langle V \cup \{z\} \mathrel{[\!]} \Delta \mathrel{[\!]} \Gamma \cup \{a[\pi \mapsto z] \bowtie b\} \mathrel{[\!]} \dots, \Phi_i \cup \{z \approx a_\pi\}, \dots\rangle}$$

if $a_\pi \in T_{\Sigma_i}(X)$; $a_\pi \notin X$; $z \notin V$

$$(\textbf{Ar})\textbf{range} \quad \frac{\langle V \mathrel{[\!]} \Delta \mathrel{[\!]} \Gamma \uplus \{x \bowtie y\} \mathrel{[\!]} \Phi_0, \dots, \Phi_n\rangle}{\langle V \mathrel{[\!]} \Delta \cup \{x \bowtie y\} \mathrel{[\!]} \Gamma \mathrel{[\!]} \Phi_0, \dots, \Phi_n\rangle}$$

$$(\textbf{De})\textbf{duct}_i \quad \frac{\langle V \mathrel{[\!]} \Delta \mathrel{[\!]} \Gamma \mathrel{[\!]} \Phi_0, \dots, \Phi_n\rangle}{\langle V \mathrel{[\!]} \Delta \cup \delta \mathrel{[\!]} \Gamma \mathrel{[\!]} \Phi_0, \dots, \Phi_n\rangle}$$

if $\mathcal{T}_i, \Phi_i \models \Lambda \longrightarrow \delta$; $\Lambda \subseteq \Delta$ is a query; δ is an answer; $\Delta \not\models \delta$

$$(\textbf{Co})\textbf{ntradict}_i \quad \frac{\langle V \mathrel{[\!]} \Delta \mathrel{[\!]} \Gamma \mathrel{[\!]} \Phi_0, \dots, \Phi_n\rangle}{\bot}$$

if $\Phi_i \wedge \Delta$ is not satisfiable

$$(\textbf{Br})\textbf{anch} \quad \frac{\langle V \mathrel{[\!]} \Delta \uplus \{x_1 \approx y_1 \vee \dots \vee x_k \approx y_k\} \mathrel{[\!]} \Gamma \mathrel{[\!]} \Phi_1, \dots, \Phi_n\rangle}{\langle V \mathrel{[\!]} \Delta \cup \{x_i \approx y_i\} \mathrel{[\!]} \Gamma \mathrel{[\!]} \Phi_1, \dots, \Phi_n\rangle}$$

if $\Delta \not\models x_i \approx y_i$; $1 \le i \le k$

Fig. 1. Inference system for combining decision procedures

Theorem 1 (Correctness). *A set of formulas Γ is satisfiable if and only if there exists a proper irreducible configuration $\mathcal{C}$ such that $\mathcal{C}_\Gamma \Rightarrow^* \mathcal{C}$.*

We will turn to the proof of Theorem 1 after a brief discussion of the rules. For convenience we treat literals as syntactically symmetric in these rules, so that $a \bowtie b$ also matches $b \bowtie a$. The rules $\textbf{Abstract}_i$ $(0 \le i \le n)$ are used to *purify* the literals of Γ. If a_π is a pure i-subterm of a, then $\textbf{Abstract}_i$ replaces a_π in a with a new variable z, at the same time adding the equation $z \approx a_\pi$ to the set Φ_i. The rule **Arrange** just transfers simple literals from Γ to Δ. The rules $\textbf{Contradict}_i$, $\textbf{Deduct}_i$ and **Branch** perform *equality propagation* by moving to Δ new (disjunctions of) simple equations that are valid in some theory $\mathcal{T}_i$.

Given a query part Λ of Δ and an answer set δ entailed by Λ and Φ_i, the rule **Deduct**$_i$ adds δ to Δ if δ is not already entailed by Δ. The rule **Contradict**$_i$ produces the configuration $\bot$ as soon as the state Φ_i becomes incompatible with Δ. Finally, the rule **Branch** performs a case split by choosing an equation from a disjunction of simple equations contained in Δ.

Example 1. The following table shows the reduction of an unsatisfiable initial configuration to $\bot$. It also uses the rule **Share**$_i$ defined later in this section. The theory $\mathcal{T}_1$ is the theory of linear arithmetic and $\mathcal{T}_0$ is the theory of one uninterpreted unary symbol f.

V	Δ	Γ	Φ_0	Φ_1	Rule
x	$\emptyset$	$f(x) \approx x$, $f(2x - f(x)) \not\approx x$	$\emptyset$	$\emptyset$	
x, y	$\emptyset$	$y \approx x$, $f(2x - f(x)) \not\approx x$	$y \approx f(x)$	$\emptyset$	$\mathbf{Ab}_0$
x, y	$y \approx x$	$f(2x - f(x)) \not\approx x$	$y \approx f(x)$	$\emptyset$	**Ar**
x, y	$y \approx x$	$f(2x - y) \not\approx x$	$y \approx f(x)$	$\emptyset$	$\mathbf{Sh}_0$
x, y, z	$y \approx x$	$f(z) \not\approx x$	$y \approx f(x)$	$z \approx 2x - y$	$\mathbf{Ab}_1$
x, y, z, u	$y \approx x$	$u \not\approx x$	$y \approx f(x), u \approx f(z)$	$z \approx 2x - y$	$\mathbf{Ab}_0$
x, y, z, u	$y \approx x, u \not\approx x$	$\emptyset$	$y \approx f(x), u \approx f(z)$	$z \approx 2x - y$	**Ar**
x, y, z, u	$y \approx x$, $u \not\approx x, z \approx x$	$\emptyset$	$y \approx f(x), u \approx f(z)$	$z \approx 2x - y$	$\mathbf{De}_1$
$\bot$					$\mathbf{Co}_0$

Remark 1. The inference system in Figure 1 leads naturally to a modularly designed combined prover of Nelson-Oppen style depicted in Figure 2. The prover consists of a core module and a set of theory modules. The behavior of the core module is specified using the rules in Figure 1. The rules suggest a natural set of interface functions for theory modules. Correctness of the prover follows from the fact that its behavior can be simulated by the inference system.

3.2 Proof of Theorem 1

The theorem follows from the following four lemmas. We give the proof only of the most important one. Complete proofs are given in the technical report [4].

Lemma 1 (Termination). *The relation $\Rightarrow$ is terminating.*

Lemma 2. *Every proper irreducible configuration is satisfiable.*

Proof. Let $\langle V \parallel \Delta \parallel \Gamma \parallel \Phi_0, \dots, \Phi_n \rangle$ be a proper irreducible configuration. Since the rules **Abstract**$_i$ and **Arrange** cannot be applied, Γ must be empty. Since **Contradict**$_i$ does not apply, $\Phi_i \wedge \Delta$ is $\mathcal{T}_i$-satisfiable for every i. If Δ is an

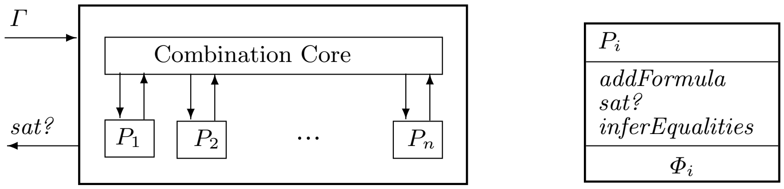

Fig. 2. Rudimentary architecture of a Nelson-Oppen prover based on the inference system in Figure 1. The interface function *addFormula* is needed to implement the rule $\mathbf{Abstract}_i$; it adds a new pure formula to the state Φ_i of the theory module P_i. Implementation of the rule $\mathbf{Contradict}_i$ requires the function *sat?* that reports whether P_i's state is inconsistent. Finally, for $\mathbf{Deduct}_i$ we need the function *inferEqualities* that computes a new disjunction of equalities that can be inferred from Δ and Φ_i.

arrangement[1], then the theorem of Tinelli and Harandi [16] implies that $\Phi_0 \wedge \cdots \wedge \Phi_n \wedge \Delta$ is satisfiable, finishing the proof. If Δ is not an arrangement, we will show that that exists an arrangement Δ' such that $\Delta' \models \Delta$ and such that $\Phi_i \wedge \Delta'$ is $\mathcal{T}_i$-satisfiable. The proof will again follow from the theorem of Tinelli and Harandi.

Take Δ' to be a maximal satisfiable extension $\Delta \cup \{x_1 \not\approx y_1, \ldots, x_k \not\approx y_k\}$ of Δ with disequations that are not entailed by Δ. If for some $x, y \in V$, neither $x \approx y$ nor $x \not\approx y$ is entailed by Δ', then $\Delta' \cup \{x \not\approx y\}$ is a satisfiable extension of Δ', contradicting the maximality assumption about Δ'. Thus, Δ' is an arrangement.

It remains to prove satisfiability of $\Phi_i \wedge \Delta'$. Assuming the contrary, we have that $\Phi_i \wedge \Delta \wedge x_1 \not\approx y_1 \wedge \cdots \wedge x_k \not\approx y_k$ is not $\mathcal{T}_i$-satisfiable. In other words, we have $\mathcal{T}_i, \Phi_i \models \Delta \longrightarrow \delta$ where δ is the answer formula $x_1 \approx y_1 \vee \cdots \vee x_k \approx y_k$. Since the **Branch** rule cannot be applied, Δ must be a set of equations and disequations. Thus, Δ is equivalent to a formula of the form $\Lambda \wedge \neg\delta'$, where Λ is a query and δ' is an answer or false. Thus, we have $\mathcal{T}_i, \Phi_i \models \Lambda \longrightarrow \delta \vee \delta'$. Since the rule $\mathbf{Deduct}_i$ cannot be applied, we conclude that $\Delta \models \delta \vee \delta'$ and then (since Δ implies $\neg\delta'$) that $\Delta \models \delta$. This contradicts the assumed satisfiability of Δ'.

Lemma 3 (Equisatisfiability). *If $\mathcal{C} \Rightarrow \mathcal{C}'$ is a non-branching reduction, then $\mathcal{C}$ and $\mathcal{C}'$ are equisatisfiable.*

Lemma 4 (Branching). *Suppose $\mathcal{C} \Rightarrow \mathcal{C}'$ is a branching reduction. Then:*

(a) if $\mathcal{C}'$ is satisfiable, then $\mathcal{C}$ is satisfiable;
(b) if $\mathcal{C}$ is satisfiable, then there exists a branching reduction $\mathcal{C} \Rightarrow \mathcal{C}''$ such that $\mathcal{C}''$ is satisfiable.

Proof of Theorem 1. It suffices to prove that a configuration $\mathcal{C}$ is satisfiable if and only if there exists a proper irreducible $\mathcal{C}'$ such that $\mathcal{C} \Rightarrow^* \mathcal{C}'$. If $\mathcal{C}$ is irreducible,

[1] Δ is an *arrangement* if for every $x, y \in V$ either $x \approx y$ or $x \not\approx y$ is implied by Δ.

the claim is true by Lemma 2. For non-irreducible $\mathcal{C}$, we have by Lemmas 3 and 4 that $\mathcal{C}$ is satisfiable if and only if there exists a satisfiable $\mathcal{C}'$ such that $\mathcal{C} \Rightarrow \mathcal{C}'$. The proof follows by wellfounded induction over the terminating relation $\Rightarrow$.

3.3 Optimized Variable Abstraction

The rules **Share**$_i$ describe a space-efficient variable abstraction mechanism which allows us to replace a subterm a_π of a term a by an existing variable z which is known by one of the theories to be equal to a_π.

$$(\mathbf{Sh})\mathbf{are}_i \qquad \frac{\langle V \;[\!]\; \Delta \;[\!]\; \Gamma \uplus \{a \bowtie b\} \;[\!]\; \Phi_0, \ldots, \Phi_n \rangle}{\langle V \;[\!]\; \Delta \;[\!]\; \Gamma \cup \{a[\pi \mapsto z] \bowtie b\} \;[\!]\; \Phi_0, \ldots, \Phi_n \rangle}$$

$$\text{if } \; a_\pi \in T_{\Sigma_i}(X); \;\; a_\pi \notin X; \;\; \mathcal{T}_i, \Phi_i \models \Lambda \longrightarrow z \approx a_\pi; \;\; \Lambda \subseteq \Delta \; \text{ is a query}$$

It is not difficult to show that Theorem 1 and the four lemmas needed for its proof all remain valid when the system in Figure 1 is extended by adding the rules **Share**$_i$.

3.4 Deduction in the Case of Convex Theories

A theory $\mathcal{T}$ is called *convex* if for every set Λ of literals the truth of a judgment of the form $\mathcal{T} \models \Lambda \longrightarrow a_1 \approx b_1 \vee \cdots \vee a_k \approx b_k$ implies $\mathcal{T} \models \Lambda \longrightarrow a_i \approx b_i$ for some i. This property allows us to simplify the system of Figure 1 by strengthening the side condition of **Deduct**$_i$ with an additional requirement that the answer formula δ be a single equation. Let us call this modified rule **DeductConvex**$_i$. The following theorem states that the system will remain correct after this change; the proof of Theorem 1 applies almost verbatim and only Lemma 2 requires a (straightforward) modification.

Theorem 2. *The correctness result expressed in Theorem 1 remains valid if for every convex theory $\mathcal{T}_i$ we replace the rule* **Deduct**$_i$ *in the inference system in Figure 1 with the rule* **DeductConvex**$_i$.

Corollary 1. *If all theories $\mathcal{T}_0, \ldots, \mathcal{T}_n$ are convex, then Theorem 1 remains valid when all the rules* **Deduct**$_i$ *are replaced with* **DeductConvex**$_i$ *and the rule* **Branch** *is excluded from the system.*

4 Strategies

Strategies introduce determinism in our inference system by constraining the shape of reduction chains. A variety of strategies can be described by using the simple language given in Figure 3. It is the language of regular expressions over the set of basic actions (rules of our inference system), extended with the operator $\oplus$. The figure also gives the semantics of the language: the concatenation

($\cdot$), and choice ($+$) operators have their standard meaning, the star ($*$) is for exhaustive application, and $\oplus$ denotes a left-associative choice that gives preference to its left argument. Clearly, every strategy e is sound in the sense that $\mathcal{C} \Rightarrow_e \mathcal{C}'$ implies $\mathcal{C} \Rightarrow^* \mathcal{C}'$.

$$a ::= \mathbf{Ab}_i \mid \mathbf{Ar} \mid \mathbf{Sh}_i \mid \mathbf{De}_i \mid \mathbf{Co} \mid \mathbf{Br}$$
$$e ::= a \mid e + e \mid e^* \mid e \cdot e \mid e \oplus e$$

$$\frac{\mathcal{C} \Rightarrow \mathcal{C}' \quad \text{by applying the rule } a}{\mathcal{C} \Rightarrow_a \mathcal{C}'}$$

$$\frac{\mathcal{C} \Rightarrow_e \mathcal{C}' \qquad \mathcal{C}' \Rightarrow_{e'} \mathcal{C}''}{\mathcal{C} \Rightarrow_{e \cdot e'} \mathcal{C}''} \qquad \frac{\mathcal{C}_0 \Rightarrow_e \cdots \Rightarrow_e \mathcal{C}_n \not\Rightarrow_e \qquad 0 \leq n}{\mathcal{C}_0 \Rightarrow_{e^*} \mathcal{C}_n}$$

$$\frac{\mathcal{C} \Rightarrow_e \mathcal{C}'}{\mathcal{C} \Rightarrow_{e+e'} \mathcal{C}'} \qquad \frac{\mathcal{C} \Rightarrow_{e'} \mathcal{C}'}{\mathcal{C} \Rightarrow_{e+e'} \mathcal{C}'} \qquad \frac{\mathcal{C} \Rightarrow_e \mathcal{C}'}{\mathcal{C} \Rightarrow_{e \oplus e'} \mathcal{C}'} \qquad \frac{\mathcal{C} \not\Rightarrow_e \qquad \mathcal{C} \Rightarrow_{e'} \mathcal{C}'}{\mathcal{C} \Rightarrow_{e \oplus e'} \mathcal{C}'}$$

Fig. 3. Syntax and semantics of a simple language for strategies.

For most of this section we will assume that all theories $\mathcal{T}_i$ are convex. Then, if a strategy e satisfies the condition

(S-1) *For every $\mathcal{C}$, there exists $\mathcal{C}'$ such that $\mathcal{C} \Rightarrow_e \mathcal{C}'$, and all such $\mathcal{C}'$ are irreducible.*

then e implements a decision procedure for the union theory $\mathcal{T}$. Indeed, for a given input Γ, we just need to find $\mathcal{C}'$ such that $\mathcal{C}_\Gamma \Rightarrow_e \mathcal{C}'$ and check whether $\mathcal{C}' = \bot$.

We will show several examples of strategies satisfying the property (S-1). Then we will see how to incorporate branching in the case when there are non-convex theories in the system.

4.1 The Basic Strategy

The following expression describes the original Nelson-Oppen algorithm for the disjoint union of convex theories.

$$\mathbf{Ab}^* \cdot \mathbf{Ar}^* \cdot (\mathbf{Co} \oplus \mathbf{De})^* \tag{1}$$

The action $\mathbf{Ab}$ is an abbreviation for $\mathbf{Ab}_0 + \cdots + \mathbf{Ab}_n$ and similarly $\mathbf{De}$ is the sum of all $\mathbf{De}_i$ (which are now $\mathbf{DeductConvex}_i$). The effect of $\mathbf{Ab}^*$ is "purification" of Γ; it reduces Γ to a set of simple literals. The action $\mathbf{Ar}^*$ then moves all these literals to Δ. Thus, $\mathbf{Ab}^* \cdot \mathbf{Ar}^*$ describes a strategy for the variable abstraction part of the algorithm.

The remaining expression $(\mathbf{Co} \oplus \mathbf{De})^*$ describes the equality propagation mechanism of the algorithm: repeated application of the rules $\mathbf{Contradict}_i$ or $\mathbf{DeductConvex}_i$ until the $\bot$ configuration is reached, or no more equations between variables can be deduced.

When applied to an arbitrary configuration $\mathcal{C}$, the strategy $\mathbf{Ab}^* \cdot \mathbf{Ar}^*$ produces configurations with empty Δ-part that are all equisatisfiable with $\mathcal{C}$. If $\mathcal{C}'$ is any of these configurations, and if it can be reduced in the original system, then every step in any reduction chain of $\mathcal{C}'$ must be by one of the rules $\mathbf{Contradict}_i$ or $\mathbf{DeductConvex}_i$. Thus, the strategy $(\mathbf{Co} \oplus \mathbf{De})^*$ when applied to $\mathcal{C}'$ produces irreducible configurations. This proves that the strategy (1) satisfies the property (S-1).

4.2 An Incremental Strategy

The following expression describes an incremental version of the strategy (1) which processes one literal of Γ at a time.

$$\big((\mathbf{Va}^1 + \cdots + \mathbf{Va}^m) \cdot (\mathbf{Co} \oplus \mathbf{De})^*\big)^* \tag{2}$$

Here we use $\mathbf{Va}^j$ as an abbreviation for the strategy $\mathbf{Ab}^* \cdot \mathbf{Ar}$ applied only to the j^{th} literal of Γ. (A precise definition would require primitive actions $\mathbf{Ab}_i^j$ and $\mathbf{Ar}^j$.) The main idea of the strategy is that processing a new literal begins only after it has been checked that the contradiction cannot be reached from the literals that have already been processed.

When applied to a configuration $\mathcal{C} = \langle V \mathbin{[\!]} \Delta \mathbin{[\!]} \Gamma \mathbin{[\!]} \Phi_0, \dots, \Phi_n \rangle$, the strategy $\mathbf{Va}^1 + \cdots + \mathbf{Va}^m$ fails only if Γ is empty; otherwise, it produces configurations of the form $\langle V' \mathbin{[\!]} \Delta' \mathbin{[\!]} \Gamma' \mathbin{[\!]} \Phi_0', \dots, \Phi_n' \rangle$, where Γ' is obtained by removing one literal from Γ. The outer closure operator in (2) guarantees that when the strategy (2) is applied to $\mathcal{C}$, the result will be a configuration equisatisfiable to $\mathcal{C}$ that is either $\bot$ or of the form $\mathcal{C}' = \langle V \mathbin{[\!]} \Delta \mathbin{[\!]} \emptyset \mathbin{[\!]} \Phi_0, \dots, \Phi_n \rangle$. Similarly as in the case of the strategy (1), we can see that $\mathcal{C}'$ is actually irreducible, proving that (2) satisfies (S-1).

4.3 Strategies with Sharing

The variable abstraction part of the previous strategies can be optimized against proliferation of new variables by an aggressive use of the rules $\mathbf{Share}_i$. Introducing sharing into the basic strategy gives

$$(\mathbf{Sh} \oplus \mathbf{Ab})^* \cdot \mathbf{Ar}^* \cdot (\mathbf{Co} \oplus \mathbf{De})^* \tag{3}$$

Similarly, the incremental strategy (2) can be optimized by replacing the action $\mathbf{Va}^j$ in it with the appropriate form of $(\mathbf{Sh} \oplus \mathbf{Ab})^* \cdot \mathbf{Ar}$. Checking the property (S-1) for these strategies proceeds as in the case of strategies (1) and (2), with minimal changes.

4.4 Branching Strategies

If some of the component theories $\mathcal{T}_i$ are not convex, then the corresponding rules $\mathbf{Deduct}_i$ must be used in place of the simpler $\mathbf{DeductConvex}_i$. The answer sets δ can now contain disjunctions and case splitting may be necessary to check the satisfiability of a configuration. A strategy that implements a decision procedure now must satisfy the following additional condition.

(S-2) *If $\mathcal{C}$ is satisfiable, then there exists a satisfiable $\mathcal{C}'$ such that $\mathcal{C} \Rightarrow_e \mathcal{C}'$.*

Since branching is expensive, the obvious approach is to use it only when everything else fails. This gives us strategies

$$(\mathbf{NO} \oplus \mathbf{Br})^* \tag{4}$$

where $\mathbf{NO}$ denotes any of the above strategies (1), (2) and (3) with $\mathbf{De}_i$ denoting $\mathbf{DeductConvex}_i$ or $\mathbf{Deduct}_i$, depending on whether $\mathcal{T}_i$ is convex or not. We know that $\mathbf{NO}$ will reduce any configuration into one to which no rule applies, except possibly **Branch**. It follows that the strategy (4) produces only irreducible configurations. It is easy to check, using Lemma 4 that this strategy also satisfies (S-2).

5 Shostak Optimization

A modular design of a decision procedure for the combined theory $\mathcal{T} = \mathcal{T}_0 + \cdots + \mathcal{T}_n$ can be derived from the inference system given in Section 3. In Remark 1 and Figure 2 we sketched such a design. Note that the strategies of the previous section are possible ways of programming the control core module. The requirements for the theory modules can be seen from Figure 1: the rule $\mathbf{Abstract}_i$ needs support for addition of a new formula to the state Φ_i of the theory module; $\mathbf{Contradict}_i$ needs a decision procedure for $\mathcal{T}_i$; and $\mathbf{Deduct}_i$ needs generation of answers from input queries. In principle, a theory module can implement this last task on top of its decision procedure: with a given input query Λ, it can search for an answer δ such that $\Lambda \wedge \neg\delta$ is unsatisfiable.

Now, for some theories there exist more efficient algorithms for computing answers to given queries. A prime example is the *free theory* over a signature consisting of uninterpreted functions, where the *congruence closure algorithm* [9,1] can process the input query and change its state appropriately so that new equations between variables can be directly seen from it. Shostak made an important discovery that a similar inference pattern is possible for many other theories [14]. Roughly speaking, the theory module maintains a union-find data structure on a set of terms so that the answer equation $x \approx y$ is deduced by checking that $\mathsf{find}(x) = \mathsf{find}(y)$ is true. To make such "trivial deduction" possible, the theory module must have some powerful mechanism for processing input queries. We describe it abstractly below by the concept of "state normalization" which essentially means bringing a set of equations (the original state together with the query equations) to some kind of normal form from which the maximum information about equalities between variables can be directly drawn.

To formalize the pattern, we need to make several assumptions. The first is that $\mathcal{T}_i$ is a convex theory with a *canonizer.* A canonizer is a function that for every term a returns a unique representative $\mathsf{canon}_i(a)$ in the equivalence class of the relation $\mathcal{T}_i \models a \approx b$.[2] A $\mathcal{T}_i$-term a is in *canonical form* when $\mathsf{canon}_i(a) = a$.

We will also assume that there is a function that picks a representative from each class of the equivalence relation on V defined by $\Delta \models x \approx y$. The representative of x will simply be denoted $\Delta(x)$. Extending this notation to terms, we will also write $\Delta(a)$ for the term in which every variable x is replaced by its representative $\Delta(x)$.

The following rule $\mathbf{TDeduct}_i$ is a trivial special case of $\mathbf{Deduct}_i$, where the answer $x \approx y$ is found by a simple lookup into the state. Similarly, $\mathbf{TShare}_i$ is a special case of $\mathbf{Share}_i$ that finds the required shared variable by inspecting the state.

(**TDe**)**duct**$_i$
$$\frac{\langle V \parallel \Delta \parallel \Gamma \parallel \dots, \Phi_i \cup \{x \approx a, y \approx a\}, \dots \rangle}{\langle V \parallel \Delta \cup \{x \approx y\} \parallel \Gamma \parallel \dots, \Phi_i \cup \{x \approx a, y \approx a\}, \dots \rangle}$$
if $\Delta(x) \neq \Delta(y)$

(**TSh**)**are**$_i$
$$\frac{\langle V \parallel \Delta \parallel \Gamma \uplus \{a \bowtie b\} \parallel \dots, \Phi_i \cup \{z \approx c\}, \dots \rangle}{\langle V \parallel \Delta \parallel \Gamma \cup \{a[\pi \mapsto z] \bowtie b\} \parallel \dots, \Phi_i \cup \{z \approx c\}, \dots \rangle}$$
if $a_\pi \in T_{\Sigma_i}(X)$; $a_\pi \notin X$; $\mathsf{canon}_i(\Delta(a_\pi)) = c$

The concept of state normalization requires a *normalization function* $\mathcal{N}_i$. If Φ_i' is the state obtained by adding equations of Δ to Φ_i, the idea is that $\mathcal{N}_i(\Phi_i, \Delta)$ denotes the first intermediate result in the (possibly multi-step) normalization process from Φ_i' to its normal form.

(**Nor**)**m**$_i$
$$\frac{\langle V \parallel \Delta \parallel \Gamma \parallel \dots, \Phi_i, \dots \rangle}{\langle V \parallel \Delta \parallel \Gamma \parallel \dots, \mathcal{N}_i(\Phi_i, \Delta), \dots \rangle}$$
if $\mathcal{N}_i(\Phi_i, \Delta) \neq \Phi_i$

In order to make the Shostak inference pattern possible, the normalization function has to satisfy the following conditions.

Termination: There exists k such that $\mathcal{N}_i^k(\Phi_i, \Delta) = \mathcal{N}_i^{k+1}(\Phi_i, \Delta)$;

Equisatisfiability: $\mathcal{T}_i \models \Phi_i \wedge \Delta \longleftrightarrow \mathcal{N}_i(\Phi_i, \Delta) \wedge \Delta$;

Completeness: If $\mathcal{T}_i, \Phi_i, \Delta \models x \approx y$ and $\Delta(x) \neq \Delta(y)$, then there exist k and a such that $\mathcal{N}_i^k(\Phi_i, \Delta)$ contains equations $x \approx a$ and $y \approx a$.

[2] Some proofs require that canonizers satisfy additional conditions. It is safe to assume that: (1) $\mathsf{canon}_i(a)$ contains only variables that occur in a; (2) all subterms of a term in canonical form are canonical too; cf. [13].

Lemma 5. *If the above three conditions are satisfied, then Theorem 2 remains valid when the rule* $\mathbf{DeductConvex}_i$ *is replaced by* $\mathbf{Norm}_i$ *and* $\mathbf{TDeduct}_i$*.*

It can also be proved that $\mathbf{Norm}_i$ and $\mathbf{TShare}_i$ together have equal optimizing effect as $\mathbf{Share}_i$. A necessary condition for this is that the normalization produces equations in which the right-hand side is in canonical form and contains only representative variables.

Presently, concrete examples of normalization are known only for the free theories and for Shostak theories. Before describing them, we give two rules that bring canonization of terms and substitution of variables with their representatives into our system. These rules simplify the state Φ_i at the term level and are the reasonable first step for any state normalization function.

$$(\mathbf{Su})\mathbf{bst}_i \qquad \frac{\langle V \,[\!]\, \Delta \,[\!]\, \Gamma \,[\!]\, \dots, \Phi_i \uplus \{x \approx a\}, \dots \rangle}{\langle V \,[\!]\, \Delta \,[\!]\, \Gamma \,[\!]\, \dots, \Phi_i \cup \{x \approx \Delta(a)\}, \dots \rangle}$$

if $a \neq \Delta(a)$ for some i

$$(\mathbf{Ca})\mathbf{nonize}_i \qquad \frac{\langle V \,[\!]\, \Delta \,[\!]\, \Gamma \,[\!]\, \dots, \Phi_i \uplus \{x \approx a\}, \dots \rangle}{\langle V \,[\!]\, \Delta \,[\!]\, \Gamma \,[\!]\, \dots, \Phi_i \cup \{x \approx \mathsf{canon}_i(a)\}, \dots \rangle}$$

if $a \neq \mathsf{canon}_i(a)$

5.1 Free Theories

To define the state normalization function for a free theory $\mathcal{T}_i$, we need to assume that every variable in V occurs as the left-hand side in at least one equation of Φ_i, and that all equations of Φ_i are of the form $x \approx y$ or $x \approx f(y_1, \dots, y_k)$, where x and y_i are variables in V. (That is, the right-hand sides can contain at most one occurrence of functional symbols.) The normalization function $\mathcal{N}_i$ just picks one of the equations and replaces the variables on its right-hand side with their Δ-representatives. In other words, in this case we have $\mathbf{Norm}_i = \mathbf{Su}_i$.

Proving that the assumptions of Lemma 5 hold for this normalization function amounts to proving correctness of the congruence closure algorithm.

5.2 Shostak Theories

Some theories admit solutions to equations. A *solver* for a theory $\mathcal{T}$ is an algorithm solve that takes a $\mathcal{T}$-equation $u \approx v$ as input, and if this equation is $\mathcal{T}$-satisfiable, solve returns its general solution in the form of an equisatisfiable set of equations

$$x_1 \approx t_1, \dots, x_k \approx t_k,$$

where the variables $x_1, \dots x_k$ are those occurring in $u \approx v$ and none of them occurs in the terms t_i. (For more details about solvers, see [13,3,6].)

By definition, a *Shostak theory* is a convex theory with a canonizer and a solver. If $\mathcal{T}_i$ is a Shostak theory, we assume that every variable occurs at most once as a left-hand side in the equations of Φ_i, and if it does have such an occurrence, then it does not occur in any right-hand side. That is, viewed as a substitution, Φ_i is idempotent.

The normalization for a Shostak theory can now be defined by

$$\mathbf{Norm}_i = \mathbf{Ca}_i \oplus \mathbf{So}_i \oplus \mathbf{Su}_i$$

where the crucial new rule $\mathbf{Solve}_i$ is as follows.

$$\boxed{\begin{array}{ll} (\mathbf{So})\mathbf{lve}_i & \dfrac{\langle V \mathbin{[\!]} \Delta \mathbin{[\!]} \Gamma \mathbin{[\!]} \ldots, \Phi_i \cup \{x \approx a, y \approx b\}, \ldots\rangle}{\langle V \mathbin{[\!]} \Delta \mathbin{[\!]} \Gamma \mathbin{[\!]} \ldots, (\Phi_i \cup \{x \approx a, y \approx b\} \cup \mathsf{solve}(a = b))^2, \ldots\rangle} \\ & \\ \text{if } \Delta(x) = \Delta(y); \quad a \neq b; \quad a \approx b \text{ is } \mathcal{T}_i\text{-satisfiable} & \end{array}}$$

To explain the rule, we note first that the variables on the left-hand sides in the set $\mathsf{solve}(a = b)$ are those of a and b, and so no variable occurs twice as a left-hand side in $\Psi = \Phi_i \cup \{x \approx a, y \approx b\} \cup \mathsf{solve}(a = b)$. Thus, Ψ defines a substitution. It is not idempotent since the variables of a and b occur also in right-hand sides of Ψ. However, the composition $\Psi^2 = \Psi \circ \Psi$ is easily seen to be idempotent, and regarded as a set of equations, it is equisatisfiable with Ψ. Thus, Ψ^2 has all the properties required for the state.

Proving that the state normalization of Shostak theories satisfies the conditions of Lemma 5 requires an effort commensurable with proving correctness of the "single theory Shostak algorithm" (Algorithm S1 of [3]). As a consequence we obtain that for a Shostak theory $\mathcal{T}_i$ the set of rules $\mathbf{Subst}_i$, $\mathbf{Canonize}_i$, $\mathbf{Solve}_i$, $\mathbf{TDeduct}_i$ and $\mathbf{TShare}_i$ can replace $\mathbf{DeductConvex}_i$ and $\mathbf{Share}_i$ in our system.

6 The Shankar-Rueß Algorithm

A highly efficient algorithm to combine decision procedures of a free theory and several Shostak theories has recently been given and proved correct by Shankar and Rueß [13]. We show now that their algorithm can be with reasonable precision described as a strategy in the language of Section 4 extended with actions corresponding to the rules introduced in Section 5. As in [13], we assume that the free theory is $\mathcal{T}_0$, and that $\mathcal{T}_1, \ldots, \mathcal{T}_n$ are Shostak theories. The strategy is given by the expression

$$\Big(\mathbf{abstraction} \cdot \big(\mathbf{Co} \oplus \mathbf{merge} \oplus \mathbf{infer} \oplus \mathbf{normalize}\big)^*\Big)^* \qquad (5)$$

where

$$\begin{aligned} \mathbf{abstraction} &= (\mathbf{Va}^1 \oplus \cdots \oplus \mathbf{Va}^m) \cdot \mathbf{Su}_0^* \\ \mathbf{merge} &= (\mathbf{So}_1 \cdot \mathbf{Ca}_1^*) + \cdots + (\mathbf{So}_n \cdot \mathbf{Ca}_n^*) \\ \mathbf{infer} &= (\mathbf{TDe}_0 + \cdots + \mathbf{TDe}_n) \cdot \mathbf{Su}_0^* \\ \mathbf{normalize} &= (\mathbf{Su}_1 + \cdots + \mathbf{Su}_n) \cdot (\mathbf{Su}_1^* \cdots \mathbf{Su}_n^*) \end{aligned}$$

Here **Va** denotes the strategy $(\mathbf{TSh} \oplus \mathbf{ASC})^* \cdot \mathbf{Ar}$, where **TSh** is the sum of all $\mathbf{TSh}_i$ and **ASC** is the sum of all $\mathbf{Ab}_i \cdot \mathbf{Su}_i^* \cdot \mathbf{Ca}_i^*$. As before, superscirpts indicate application to a particular literal of Γ.

The algorithm starts by executing an efficient *incremental* variable abstraction; hence the superscripts in **abstraction** and the outer star operator in (5). **abstraction** generates new equations only when the rules $\mathbf{TShare}_i$ fail to find shared variables. It also maintains the sets Φ_i in normal form by applying $\mathbf{Subst}_i$ and $\mathbf{Canonize}_i$ exhaustively. After this step comes the equality propagation mechanism. It immediately examines all theory states attempting to find a contradiction in one. If this fails, every Φ_i is satisfiable, and then the state normalization is initiated by **merge** which solves one equation $x \approx y$ in some Shostak theory state Φ_i when the variables x and y are equal in Δ but not yet in Φ_i. **merge** finishes by restoring the normal form of Φ_i with exhaustive application of $\mathbf{Canonize}_i$. ($\mathbf{Subst}_i$ is unnecessary here, since the right-hand side variables of Φ_i are all representatives.) When the state is in normal form and if x and y are equal in some Φ_i but not in Δ, **infer** propagates the new equality $x \approx y$ to Δ and normalizes the set Φ_0 by applying $\mathbf{Subst}_0$ exhaustively. Finally, **normalize** substitutes the variables in the Shostak theory states Φ_i by their new representatives which may have been added to Δ by **infer**.

7 Conclusion and Related Work

We have presented results of our initial study of design of correct algorithms for combining decision procedures. Having in mind a modular implementation with theory modules as black boxes and a programmable control core module, we formalized the entire system as an inference system that is convenient to reason about and to refine. Our system is of Nelson-Oppen type, but we have shown that the congruence closure algorithm and the Shostak algorithm can be incorporated into it with additional rules so that overall correctness is preserved. We have given a simple strategy language capable of expressing complex combination algorithms. Proving correctness of a concrete algorithm written as a strategy is reduced to proving one or two simple properties of the strategy; the rest follows from the correctness of the whole system.

The Nelson-Oppen method has been widely adopted as the basis for combination algorithms [12]. Its bare bones versions are described and proved correct by Ringeissen [10] and by Tinelli and Harandi [16]. We work at the level of abstraction that is close to these works, but our system is extended with implementation-related details.

A series of recent papers is devoted to proofs of correctness of various versions of the Shostak algorithm. Rueß and Shankar [11] and Ganzinger [6] consider the algorithm for combining a free theory with one Shostak theory. In Barrett, Dill and Stump [3], the algorithm is for the combination of a Shostak theory with any convex theory. Finally, Shankar and Rueß [13] settle the case of a free theory combined with an arbitrary number of Shostak theories. (The same case is considered in the preliminary draft [7].) We have borrowed from all these sources.

In particular, the idea to model the whole system by state-transformation rules is already in [6] and in [1,17], which also uses regular expressions to express various strategies for the same system. Our system allows arbitrary combinations of stably-infinite theories and so is significantly more general. Moreover, this generality does not come at the price of ignoring important details, as demonstrated by modeling the Shankar-Rueß algorithm as a strategy for our system.

We leave for future work a description of a modular implementation of our system, with precise interfaces for theory modules. The intention is to establish correctness of such an implementation by simulating it in our abstract system. A similar project has been carried out very recently by Barrett [2]. He verified a combination procedure described as a modular system with an impressive list of implementation features; his system includes non-convex theories, but allows only one Shostak theory. We believe our approach will lead to shorter and more general proofs.

We also believe our work will contribute to the understanding of the scope of the Shostak algorithm. We hypothesize that in a modular implementation there is no advantage in allowing the core module to have access to Shostak module primitives (canonizer and solver); the same efficiency can be achieved with a plain Nelson-Oppen core that communicates with Shostak theory modules through generic theory module interfaces, while canonizer and solver are used only to implement those interfaces. If this is correct, the Shostak algorithm would largely be a single theory affair; cf. [3]. We expect to gain some insights by comparing the complexity (number of reductions needed for a given initial configuration) of the Rueß-Shankar strategy against our best strategy that uses Shostak theories in a generic way.

Acknowledgments. We thank John Matthews and Andrew Tolmach for valuable discussions, comments, and corrections.

References

1. L. Bachmair, A. Tiwari, and L. Vigneron. Abstract congruence closure. *Journal of Automated Reasoning*, 2002. To appear.
2. C. Barrett. *Checking Validity of Quantifier-free formulas in Combinations of First-Order Theories.* PhD thesis, Stanford University, 2002.
3. C. W. Barrett, D. L. Dill, and A. Stump. A generalization of Shostak's method for combining decision procedures. In *Frontiers of Combining Systems (FROCOS)*, volume 2309 of *Lecture Notes in Artificial Intelligence*, pages 132–147. Springer-Verlag, 2002.
4. S. Conchon and S. Krstic. Strategies for combining decision procedures. Technical Report CSE-03-001, OHSU, 2003.
5. J.-C. Filliâtre, S. Owre, H. Rueß, and N. Shankar. ICS: Integrated Canonization and Solving (Tool presentation). In G. Berry, H. Comon, and A. Finkel, editors, *Proceedings of CAV'2001*, volume 2102 of *Lecture Notes in Computer Science*, pages 246–249. Springer-Verlag, 2001.

6. H. Ganzinger. Shostak light. In A. Voronkov, editor, *Automated Deduction – CADE-18*, volume 2392 of *Lecture Notes in Artificial Intelligence*, pages 332–347. Springer-Verlag, 2002.
7. D. Kapur. A rewrite rule based framework for combining decision procedures. In *Frontiers of Combining Systems (FROCOS)*, volume 2309 of *Lecture Notes in Artificial Intelligence*, pages 87–103. Springer-Verlag, 2002.
8. G. Nelson and D. C. Oppen. Simplification by cooperating decision procedures. *ACM Transactions on Programming Languages and Systems*, 1(2):245–257, 1979.
9. G. Nelson and D. C. Oppen. Fast decision procedures based on congruence closure. *JACM*, 27(2):356–364, 1980.
10. Ch. Ringeissen. Cooperation of Decision Procedures for the Satisfiability Problem. In F. Baader and K. U. Schulz, editors, *Frontiers of Combining Systems: Proceedings of the 1st International Workshop*, Applied Logic, pages 121–140. Kluwer Academic Publishers, 1996.
11. H. Rueß and N. Shankar. Deconstructing Shostak. In *Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science (LICS-01)*, pages 19–28. IEEE Computer Society, 2001.
12. N. Shankar. Little engines of proof. In L.-H. Eriksson and P. Lindsay, editors, *FME 2002: Formal Methods – Getting IT Right*, pages 1–20, Copenhagen, 2002. Springer-Verlag.
13. N. Shankar and H. Rueß. Combining Shostak theories. In S. Tison, editor, *Rewriting Techniques and Applications (RTA)*, volume 2378 of *Lecture Notes in Computer Science*, pages 1–19. Springer-Verlag, 2002.
14. R. E. Shostak. Deciding combinations of theories. *Journal of the ACM*, 31(1):1–12, 1984.
15. A. Stump, C. Barrett, and D. Dill. CVC: a Cooperating Validity Checker. In *14th International Conference on Computer-Aided Verification*, 2002.
16. C. Tinelli and M. T. Harandi. A new correctness proof of the Nelson–Oppen combination procedure. In F. Baader and K. U. Schulz, editors, *Frontiers of Combining Systems: Proceedings of the 1st International Workshop*, Applied Logic, pages 103–120. Kluwer Academic Publishers, 1996.
17. A. Tiwari. *Decision Procedures in Automated Deduction.* PhD thesis, University of Stony Brook, 2000.