Invited paper for FLoC'02. Appears in the Proceedings of RTA'02, LNCS.
©Springer-Verlag

Combining Shostak Theories*

Natarajan Shankar and Harald Rueß

SRI International Computer Science Laboratory
Menlo Park CA 94025 USA
{shankar, ruess}@csl.sri.com
URL: http://www.csl.sri.com/{~shankar, ~ruess}
Phone: +1 (650) 859-5272 Fax: +1 (650) 859-2844

Abstract. Ground decision procedures for combinations of theories are used in many systems for automated deduction. There are two basic paradigms for combining decision procedures. The Nelson–Oppen method combines decision procedures for disjoint theories by exchanging equality information on the shared variables. In Shostak's method, the combination of the theory of pure equality with canonizable and solvable theories is decided through an extension of congruence closure that yields a canonizer for the combined theory. Shostak's original presentation, and others that followed it, contained serious errors which were corrected for the basic procedure by the present authors. Shostak also claimed that it was possible to combine canonizers and solvers for disjoint theories. This claim is easily verifiable for canonizers, but is unsubstantiated for the case of solvers. We show how our earlier procedure can be extended to combine multiple disjoint canonizable, solvable theories within the Shostak framework.

1 Introduction

Consider the sequent

$$2 * car(x) - 3 * cdr(x) = f(cdr(x)) \\ \vdash f(cons(4 * car(x) - 2 * f(cdr(x)), y)) = f(cons(6 * cdr(x), y)).$$

* This work was funded by NSF Grant CCR-0082560, DARPA/AFRL Contract F33615-00-C-3043, and NASA Contract NAS1-00079. During a phone conversation with the first author on 2nd April 2001, Rob Shostak suggested that the problem of combining Shostak solvers could be solved through variable abstraction. His suggestion is the key inspiration for the combination of Shostak theories presented here. We thank Clark Barrett, Sam Owre, and Ashish Tiwari for their meticulous reading of earlier drafts. We also thank Harald Ganzinger for pointing out certain limitations of our original definition of solvability with respect to σ-models. The first author is grateful to the program committees and program chairs of the FME, LICS, and RTA conferences at FLoC 2002 for their kind invitation.

It involves symbols from three different theories. The symbol f is uninterpreted, the operations $*$ and $-$ are from the theory of linear arithmetic, and the pairing and projection operations *cons*, *car*, and *cdr*, are from the theory of lists. There are two basic methods for building combined decision procedures for disjoint theories, i.e., theories that share no function symbols. Nelson and Oppen [NO79] gave a method for combining decision procedures through the use of variable abstraction for replacing subterms with variables, and the exchange of equality information on the shared variables. Thus, with respect to the example above, decision procedures for pure equality, linear arithmetic, and the theory of lists can be composed into a decision procedure for the combined theory. The other combination method, due to Shostak, yields a decision procedure for the combination of canonizable and solvable theories, based on the congruence closure procedure. Shostak's original algorithm and proof were seriously flawed. His algorithm is neither terminating nor complete (even when terminating). These flaws went unnoticed for a long time even though the method was widely used, implemented, and studied [CLS96,BDL96,Bjø99]. In earlier work [RS01], we described a correct algorithm for the *basic* combination of a single canonizable, solvable theory with the theory of equality over uninterpreted terms. That correctness proof has been mechanically verified using PVS [FS02]. The generality of the basic combination rests on Shostak's claim that it is possible to combine solvers and canonizers from disjoint theories into a single canonizer and solver. This claim is easily verifiable for canonizers, but fails for the case of solvers. In this paper, we extend our earlier decision procedure to the combination of uninterpreted equality with multiple canonizable, solvable theories. The decision procedure does not require the combination of solvers. We present proofs for the termination, soundness, and completeness of our procedure.

2 Preliminaries

We introduce some of the basic terminology needed to understand Shostak-style decision procedures. Fixing a countable set of variables X and a set of function symbols F, a term is either a variable x from X or an n-ary function symbol f from F applied to n terms as in $f(a_1, \ldots, a_n)$. Equations between terms are represented as $a = b$. Let $vars(a)$, $vars(a = b)$, and $vars(T)$ represent the sets of variables in a, $a = b$, and the set of equalities T, respectively. We are interested in deciding the validity of sequents of the form $T \vdash c = d$ where c and d are terms, and T is a set of equalities such that $vars(c = d) \subseteq vars(T)$. The condition $vars(c = d) \subseteq vars(T)$ is there for technical reasons. It can always be satisfied by padding T with reflexivity assertions $x = x$ for any variables x in $vars(c = d) - vars(T)$. We write $[\![a]\!]$ for the set of subterms of a, which includes a.

The semantics for a term a, written as $M[\![a]\!]\rho$, is given relative to an interpretation M over a domain D and an assignment ρ. For an n-ary function f, the interpretation $M(f)$ of f in M is a map from D^n to D. For an *uninterpreted*

n-ary function symbol f, the interpretation $M(f)$ may be any map from D^n to D, whereas only restricted interpretations might be suitable for an interpreted function symbol like the arithmetic + operation. An assignment ρ is a map from variables in X to values in D. We define $M[\![a]\!]\rho$ to return a value in D by means of the following equations.

$$M[\![x]\!]\rho = \rho(x)$$
$$M[\![f(a_1, \ldots, a_n)]\!]\rho = M(f)(M[\![a_1]\!]\rho, \ldots, M[\![a_n]\!]\rho)$$

We say that $M, \rho \models a = b$ iff $M[\![a]\!]\rho = M[\![b]\!]\rho$, and $M \models a = b$ iff $M, \rho \models a = b$ for all assignments ρ. We write $M, \rho \models S$ when $\forall a, b : a = b \in S \Rightarrow M, \rho \models a = b$, and $M, \rho \models (T \vdash a = b)$ when $(M, \rho \models T) \Rightarrow (M, \rho \models a = b)$. A sequent $T \vdash c = d$ is valid, written as $\models (T \vdash c = d)$, when $M, \rho \models (T \vdash c = d)$, for all M and ρ.

There is a simple pattern underlying the class of decision procedures studied here. Let ψ be the state of the decision procedure as given by a set of formulas.[1] Let τ be a family of state transformations so that we write $\psi \xrightarrow{\tau} \psi'$ if ψ' is the result of applying a transformation in τ to ψ, where $vars(\psi) \subseteq vars(\psi')$ (variable preservation). An assignment ρ' is said to extend ρ over $vars(\psi') - vars(\psi)$ when it agrees with ρ on all variables except those in $vars(\psi') - vars(\psi)$ for $vars(\psi) \subseteq vars(\psi')$. We say that ψ' *preserves* ψ if $vars(\psi) \subseteq vars(\psi')$ and for all interpretations M and assignments ρ, $M, \rho \models \psi$ holds iff there exists an assignment ρ' extending ρ such that $M, \rho' \models \psi'$.[2] When preservation is restricted to a limited class of interpretations ι, we say that ψ' ι-preserves ψ. Note that the *preserves* relation is transitive. When the operation τ is deterministic, $\tau(\psi)$ represents the result of the transformation, and we call τ a *conservative* operation to indicate that $\tau(\psi)$ preserves ψ for all ψ. Correspondingly, τ is said to be ι-conservative when $\tau(\psi)$ ι-preserves ψ. Let τ^n represent the n-fold iteration of τ, then τ^n is a conservative operation. The composition $\tau_2 \circ \tau_1$ of conservative operations τ_1 and τ_2, is also a conservative operation. The operation $\tau^*(\psi)$ is defined as $\tau^i(\psi)$ for the least i such that $\tau^{i+1}(\psi) = \tau^i(\psi)$. The existence of such a bound i must be demonstrated for the termination of τ^*. If τ is conservative, so is τ^*.

If τ is a conservative operation, it is sound and complete in the sense that for a formula ϕ with $vars(\phi) \subseteq vars(\psi)$, $\models (\psi \vdash \phi)$ iff $\models (\tau(\psi) \vdash \phi)$. This is clear since τ is a conservative operation and $vars(\phi) \subseteq vars(\psi)$.

[1] In our case, the state is actually represented by a list whose elements are sets of equalities. We abuse notation by viewing such a state as the set of equalities corresponding to the union of the sets of equalities contained in it.

[2] In general, one could allow the interpretation M to be extended to M' in the transformation from ψ to ψ' to allow for the introduction of new function symbols, e.g., skolem functions. This abstract design pattern then also covers skolemization in addition to methods like prenexing, clausification, resolution, variable abstraction, and Knuth-Bendix completion.

If $\tau^*(\psi)$ returns a state ψ' such that $\models (\psi' \vdash \bot)$, where $\bot$ is an unsatisfiable formula, then ψ' and ψ are both clearly unsatisfiable. Otherwise, if ψ' is *canonical*, as explained below, $\models (\psi' \vdash \phi)$ can be decided by computing a canonical form $\psi'[\![\phi]\!]$ for ϕ with respect to ψ'.

3 Congruence Closure

In this section, we present a warm-up exercise for deciding equality over terms where all function symbols are uninterpreted, i.e., the interpretation of these operations is unconstrained. This means that a sequent $T \vdash c = d$ is valid, i.e., $\models (T \vdash c = d)$ iff for all interpretations M and assignments ρ, the satisfaction relation $M, \rho \models (T \vdash c = d)$ holds. Whenever we write $f(a_1, \ldots, a_n)$, the function symbol f is uninterpreted, and $f(a_1, \ldots, a_n)$ is then said to be uninterpreted. Later on, we will extend the procedure to allow interpreted function symbols from disjoint Shostak theories such as linear arithmetic and lists. The congruence closure procedure sets up the template for the extended procedure in Section 5.

The congruence closure decision procedure for *pure equality* has been studied by Kozen [Koz77], Shostak [Sho78], Nelson and Oppen [NO80], Downey, Sethi, and Tarjan [DST80], and, more recently, by Kapur [Kap97]. We present the congruence closure algorithm in a Shostak-style, i.e., as an online algorithm for computing and using canonical forms by successively processing the input equations from the set T. For ease of presentation, we make use of variable abstraction in the style of the abstract congruence closure technique due to Bachmair, Tiwari, and Vigneron [BTV02]. Terms of the form $f(a_1, \ldots, a_n)$ are variable-abstracted into the form $f(x_1, \ldots, x_n)$ where the variables $x_1, \ldots, x_n$ abstract the terms $a_1, \ldots, a_n$, respectively. The procedure shown here can be seen as a specific strategy for applying the abstract congruence closure rules. In Section 5, we make essential use of variable abstraction in the Nelson–Oppen style where it is not merely a presentation device.

Let $T = \{a_1 = b_1, \ldots, a_n = b_n\}$ for $n \geq 0$ so that T is empty when $n = 0$. Let x and y be metavariables that range over variables. The state of the algorithm consists of a *solution state* S and the input equalities T. The solution state S will be maintained as the pair $(S_V; S_U)$, where $(l_1; l_2; \ldots; l_n)$ represents a list with n elements and semi-colon is an associative separator for list elements. The set S_U then contains equalities of the form $x = f(x_1, \ldots, x_n)$ for an n-ary uninterpreted function f, and the set S_V contains equalities of the form $x = y$ between variables. We blur the distinction between the equality $a = b$ and the singleton set $\{a = b\}$. Syntactic identity is written as $a \equiv b$ as opposed to semantic equality $a = b$.

A set of equalities R is *functional* if $b \equiv c$ whenever $a = b \in R$ and $a = c \in R$, for any a, b, and c. If R is functional, it can be used as a lookup table for obtaining the right-hand side entry corresponding to a left-hand side expression. Thus $R(a) = b$ if $a = b \in R$, and otherwise, $R(a) = a$. The domain of R, $dom(R)$

is defined as $\{a \mid a = b \in R \text{ for some } b\}$. When R is not necessarily functional, we use $R(\{a\})$ to represent the set $\{b \mid a = b \in R \vee b \equiv a\}$ which is the image of $\{a\}$ with respect to the reflexive closure of R. The inverse of R, written as R^{-1}, is the set $\{b = a \mid a = b \in R\}$. A functional set R of equalities can be applied as in $R[a]$.

$$\begin{aligned} R[x] &= R(x) \\ R[f(a_1, \ldots, a_n)] &= R(f(R[a_1], \ldots, R[a_n])) \\ R[\{a_1 = b_1, \ldots, a_n = b_n\}] &= \{R[a_1] = R[b_1], \ldots, R[a_n] = R[b_n]\} \end{aligned}$$

In typical usage, R will be a *solution set* where the left-hand sides are all variables, so that $R[a]$ is just the result of applying R as a substitution to a.

When S_V is functional, then S given by $(S_V; S_U)$ can also be used to compute the canonical form $S[\![a]\!]$ of a term a with respect to S. Hilbert's epsilon operator is used in the form of the *when* operator: $F(\overline{x})$ when $\overline{x} : P(\overline{x})$ is an abbreviation for $F(\epsilon\overline{x} : \ P(\overline{x}))$, if $\exists\overline{x} : P(\overline{x})$.

$$\begin{aligned} S[\![x]\!] &= S_V(x) \\ S[\![f(a_1, \ldots, a_n)]\!] &= S_V(x), \text{ when } x : x = f(S[\![a_1]\!], \ldots, S[\![a_n]\!]) \in S_U \\ S[\![f(a_1, \ldots, a_n)]\!] &= f(S[\![a_1]\!], \ldots, S[\![a_n]\!]), \text{ otherwise.} \end{aligned}$$

The set S_V of variable equalities will be maintained so that $vars(S_V) \cup vars(S_U) = dom(S_V)$. The set S_V partitions the variables in $dom(S_V)$ into equivalence classes. Two variables x and y are said to be in the same equivalence class with respect to S_V if $S_V(x) \equiv S_V(y)$. If R and R' are solution sets and R' is functional, then $R \triangleright R' = \{a = R'[b] \mid a = b \in R\}$, and $R \circ R' = R' \cup (R \triangleright R')$. The set S_V is maintained in idempotent form so that $S_V \circ S_V = S_V$. Note that S_U need not be functional since it can, for example, simultaneously contain the equations $x = f(y)$, $x = f(z)$, and $x = g(y)$.

We assume a strict total ordering $x \prec y$ on variables. The operation $orient(x = y)$ returns $\{x = y\}$ if $x \prec y$, and returns $\{y = x\}$, otherwise. The solution state S is said to be *congruence-closed* if $S_U(\{x\}) \cap S_U(\{y\}) = \emptyset$ whenever $S_V(x) \not\equiv S_V(y)$. A solution set S is *canonical* if S is congruence-closed, S_V is functional and idempotent, and S_U is normalized, i.e., $S_U \triangleright S_V = S_U$.

In order to determine if $\models (T \vdash c = d)$, we check if $S'[\![c]\!] \equiv S'[\![d]\!]$ for $S' = process(S; T)$, where $S = (S_V; S_U)$, $S_V = id_T$, $id_T = \{x = x \mid x \in vars(T)\}$, and $S_U = \emptyset$. The congruence closure procedure *process* is defined in Figure 1.

Explanation. We explain the congruence closure procedure using the validity of the sequent $f(f(f(x))) = x,\ x = f(f(x)) \vdash f(x) = x$ as an example. Its validity will be verified by constructing a solution state S' equal to $process(S_V; S_U; T)$ for $T = \{f(f(f(x))) = x, x = f(f(x))\}$, $S_V = id_T$, $S_U = \emptyset$, and checking $S'[\![f(x)]\!] \equiv S'[\![x]\!]$. Note that id_T is $\{x = x\}$. In processing $f(f(f(x))) = x$ with respect to S, the canonization step, $S[\![f(f(f(x))) = x]\!]$

$$
\begin{aligned}
process(S;\emptyset) &= S \\
process(S;\{a=b\}\cup T) &= process(S';T),\ \text{where,} \\
&\quad S' = close^*(merge(abstract^*(S;S[\![a=b]\!]))).
\end{aligned}
$$

$$
\begin{aligned}
close(S) &= merge(S;S_V(x)=S_V(y)), \\
&\quad \text{when } x,y : S_V(x)\not\equiv S_V(y), (S_U(\{x\})\cap S_U(\{y\})\neq\emptyset) \\
close(S) &= S,\ \text{otherwise.}
\end{aligned}
$$

$$
\begin{aligned}
merge(S;x=x) &= S \\
merge(S;x=y) &= (S'_V;\ S'_U),\ \text{where } x\not\equiv y, R = orient(x=y), \\
&\quad S'_V = S_V\circ R, S'_U = S_U \triangleright R.
\end{aligned}
$$

$$
\begin{aligned}
abstract(S;x=y) &= (S;\ x=y) \\
abstract(S;a=b) &= (S';\ a'=b'), \text{when } S', a', b', x_1,\ldots,x_n : \\
&\quad f(x_1,\ldots,x_n)\in[\![a=b]\!] \\
&\quad x\notin vars(S;a=b) \\
&\quad R=\{x=f(x_1,\ldots,x_n)\}, \\
&\quad S'=(S_V\cup\{x=x\};\ S_U\cup R), \\
&\quad a'=R^{-1}[a], b'=R^{-1}[b].
\end{aligned}
$$

Fig. 1. Congruence closure

yields $f(f(f(x))) = x$, unchanged. Next, the variable abstraction step computes $abstract^*(f(f(f(x))) = x)$. First $f(x)$ is abstracted to v_1 yielding the state $\{x = x, v_1 = v_1\};\{v_1 = f(x)\};\{f(f(v_1)) = x\}$. Variable abstraction eventually terminates renaming $f(v_1)$ to v_2 and $f(v_2)$ to v_3 so that S is $\{x = x, v_1 = v_1, v_2 = v_2, v_3 = v_3\};\{v_1 = f(x), v_2 = f(v_1), v_3 = f(v_2)\}$. The variable abstracted input equality is then $v_3 = x$. Let $orient(v_3 = x)$ return $v_3 = x$. Next, $merge(S; v_3 = x)$ yields the solution state $\{x = x, v_1 = v_1, v_2 = v_2, v_3 = x\};\{v_1 = f(x), v_2 = f(v_1), v_3 = f(v_2)\}$. The congruence closure step $close^*(S)$ leaves S unchanged since there are no variables that are merged in S_U and not in S_V.

The next input equality $x = f(f(x))$ is canonized as $x = v_2$ which can be oriented as $v_2 = x$ and merged with S to yield the new value $\{x = x, v_1 = v_1, v_2 = x, v_3 = x\};\{v_1 = f(x), v_2 = f(v_1), v_3 = f(x)\}$ for S. The congruence closure step $close^*(S)$ now detects that v_1 and v_3 are merged in S_U but not in S_V and generates the equality $v_1 = v_3$. This equality is merged to yield the new value of S as $\{x = x, v_1 = x, v_2 = x, v_3 = x\};\{v_1 = f(x), v_2 = f(x), v_3 = f(x)\}$, which is congruence-closed.

With respect to this final value of the solution state S, it can be checked that $S[\![f(x)]\!] \equiv x \equiv S[\![x]\!]$.

Invariants. The Shostak-style congruence closure algorithm makes heavy use of canonical forms and this requires some key invariants to be preserved on the solution state S. If $vars(S_V) \cup vars(S_U) \subseteq dom(S_V)$, then $vars(S'_V) \cup vars(S'_U) \subseteq dom(S'_V)$, when S' is either $abstract(S; a = b)$ or $close(S)$. If S is canonical and $a' = S[\![a]\!]$, then $S_V[a'] = a'$. If $S_U \triangleright S_V = S_U$, $S_V[a] = a$, and $S_V[b] = b$, then $S'_U \triangleright S'_V = S'_U$ where $S'; a' = b'$ is $abstract(S; a = b)$. Similarly, if $S_U \triangleright S_V = S_U$, $S_V(x) \equiv x$, $S_V(y) \equiv y$, then $S'_U \circ S'_V = S'_U$ for $S' = merge(S; x = y)$. If S_V is functional and idempotent, then so is S'_V, where S' is either of $abstract(S; a = b)$ or $close(S)$. If $S' = close^*(S)$, then S' is congruence-closed, and if S_V is functional and idempotent, S_U is normalized, then S' is canonical.

Variations. In the *merge* operation, if S'_U is computed as $R[S_U]$ instead of $S_U \triangleright R$, this would preserve the invariant that S_U^{-1} is always functional and $S_V[S_U] = S_U$. If this is the case, the canonizer can be simplified to just return $S_U^{-1}(f(S[\![a_1]\!], \ldots, S[\![a_n]\!]))$.

Termination. The procedure $process(S;\ T)$ terminates after each equality in T has been asserted into S. The operation $abstract^*$ terminates because each recursive call decreases the number of occurrences of function applications in the given equality $a = b$ by at least one. The operation $close^*$ terminates because each invocation of the *merge* operation merges two distinct equivalence classes of variables in S_V. The *process* operation terminates because the number of input equations in T decreases with each recursive call. Therefore the computation of $process(S;\ T)$ terminates returning a canonical solution set S'.

Soundness and Completeness. We need to show that $\models (T \vdash c = d) \iff S'[\![c]\!] \equiv S'[\![d]\!]$ for $S' = process(id_T;\ \emptyset;\ T)$ and $vars(c = d) \subseteq vars(T)$. We do this by showing that S' preserves $(id_T;\ \emptyset;\ T)$, and hence $\models (T \vdash c = d) \iff \models (S' \vdash c = d)$, and $\models (S' \vdash c = d) \iff S'[\![c]\!] \equiv S'[\![d]\!]$. We can easily establish that if $process(S;\ T) = S'$, then S' preserves $(S;\ T)$. If $a' = b'$ is obtained from $a = b$ by applying equality replacements from S, then $(S;\ a' = b')$ preserves $(S;\ a = b)$. In particular, $\models (S \vdash S[\![c]\!] = c)$ holds. The following claims can then be easily verified.

1. $(S;\ S[\![a = b]\!])$ preserves $(S;\ a = b)$.
2. $abstract(S;\ a = b)$ preserves $(S;\ a = b)$.
3. $merge(S;\ a = b)$ preserves $(S;\ a = b)$.
4. $close(S)$ preserves S.

The only remaining step is to show that if S' is canonical, then $\models (S' \vdash c = d) \iff S'[\![c]\!] \equiv S'[\![d]\!]$ for $vars(c = d) \subseteq vars(S)$. Since we know that $\models S' \vdash S'[\![c]\!] = c$ and $\models S' \vdash S'[\![d]\!] = d$, hence $\models (S' \vdash c = d)$ follows from $S'[\![c]\!] \equiv S'[\![d]\!]$. For the *only if* direction, we show that if $S'[\![c]\!] \not\equiv S'[\![d]\!]$, then there is an interpretation $M_{S'}$ and assignment $\rho_{S'}$ such that $M_{S'}, \rho_{S'} \models S$ but $M_{S'}, \rho_{S'} \not\models c = d$. A *canonical* term (in S') is a term a such that $S'[\![a]\!] \equiv a$. The domain $D_{S'}$ is taken to be the set of canonical terms built from the function symbols F and variables from $vars(S')$. We constrain $M_{S'}$ so that $M_{S'}(f)(a_1, \ldots, a_n) = S'_V(x)$

when there is an x such that $x = f(a_1, \ldots, a_n) \in S'_U$, and $f(a_1, \ldots, a_n)$, otherwise. Let $\rho_{S'}$ map x in $vars(S')$ to $S'_V(x)$; the mappings for the variables outside $vars(S')$ are irrelevant. It is easy to see that $M_{S'}[\![c]\!]\rho_{S'} = S'[\![c]\!]$ by induction on the structure of c. In particular, when S' is canonical, $M_{S'}(f)(x_1 \ldots, x_n) = x$ for $x = f(x_1, \ldots, x_n) \in S'_U$, so that one can easily verify that $M_{S'}, \rho_{S'} \models S'$. Hence, if $S'[\![c]\!] \not\equiv S'[\![d]\!]$, then $\not\models (S' \vdash c = d)$.

4 Shostak Theories

A Shostak theory [Sho84] is a theory that is canonizable and solvable. We assume a collection of Shostak theories $\theta_1, \ldots, \theta_N$. In this section, we give a decision procedure for a single Shostak theory θ_i, but with i as a parameter. This background material is adapted from Shankar [Sha01]. Satisfiability $M, \rho \models a = b$ is with respect to i-models M. The equality $a = b$ is i-valid, i.e., $\models_i a = b$, if for all i-models M and assignments ρ, $M[\![a]\!]\rho = M[\![b]\!]\rho$. Similarly, $a = b$ is i-unsatisfiable, i.e., $\models_i a \neq b$, when for all i-models M and assignments ρ, $M[\![a]\!]\rho \neq M[\![b]\!]\rho$. An i-term a is a term whose function symbols all belong to θ_i and $vars(a) \subseteq X \cup X_i$.

A canonizable theory θ_i admits a computable operation σ_i on terms such that $\models_i a = b$ iff $\sigma_i(a) \equiv \sigma_i(b)$, for i-terms a and b. An i-term a is canonical if $\sigma_i(a) \equiv a$. Additionally, $vars(\sigma_i(a)) \subseteq vars(a)$ and every subterm of $\sigma_i(a)$ must be canonical. For example, a canonizer for the theory θ_A of linear arithmetic can be defined to convert expressions into an ordered sum-of-monomials form. Then, $\sigma_A(y + x + x) \equiv 2 * x + y \equiv \sigma_A(x + y + x)$.

A solvable theory admits a procedure $solve_i$ on equalities such that $solve_i(Y)(a = b)$ for a set of variables Y with $vars(a = b) \subseteq Y$, returns a solved form for $a = b$ as explained below. $solve_i(Y)(a = b)$ might contain fresh variables that do not appear in Y. A functional solution set R is in *i-solved form* if it is of the form $\{x_1 = t_1, \ldots, x_n = t_n\}$, where for j, $1 \leq j \leq n$, t_j is a canonical i-term, $\sigma_i(t_j) \equiv t_j$, and $vars(t_j) \cap dom(R) = \emptyset$ unless $t_j \equiv x_j$. The i-solved form $solve_i(Y)(a = b)$ is either $\bot_i$, when $\models_i a \neq b$, or is a solution set of equalities which is the union of sets R_1 and R_2. The set R_1 is the solved form $\{x_1 = t_1, \ldots, x_n = t_n\}$ with $x_j \in vars(a = b)$ for $1 \leq j \leq n$, and for any i-model M and assignment ρ, we have that $M, \rho \models a = b$ iff there is a ρ' extending ρ over $vars(solve_i(Y)(a = b)) - Y$ such that $M, \rho' \models x_j = t_j$, for $1 \leq j \leq n$. The set R_2 is just $\{x = x \mid x \in vars(R_1) - Y\}$ and is included in order to preserve variables. In other words, $solve_i(Y)(a = b)$ i-preserves $a = b$. For example, a solver for linear arithmetic can be constructed to isolate a variable on one side of the equality through scaling and cancellation. We assume that the fresh variables generated by $solve_i$ are from the set X_i. We take $vars(\bot_i)$ to be $X \cup X_i$ so as to maintain variable preservation, and indeed $\bot_i$ could be represented as just $\bot$ were it not for this condition.

We now describe a decision procedure for sequents of the form $T \vdash c = d$ in a single Shostak theory with canonizer σ_i and solver $solve_i$. Here the solution state

S is just a functional solution set of equalities in i-solved form. Given a solution set S, we define $S\langle\!\langle a \rangle\!\rangle_i$ as $\sigma_i(S[a])$. The composition of solutions sets is defined so that $S \circ_i \bot_i = \bot_i \circ_i S = \bot_i$ and $S \circ_i R = R \cup \{a = R\langle\!\langle b \rangle\!\rangle_i \mid a = b \in S\}$. Note that solved forms are idempotent with respect to composition so that $S \circ_i S = S$. The solved form $solveclose_i(id_T;\ T)$ is obtained by processing the equations in T to build up a solution set S. An equation $a = b$ is first canonized with respect to S as $S\langle\!\langle a \rangle\!\rangle_i = S\langle\!\langle b \rangle\!\rangle_i$ and then solved to yield the solution R. If R is $\bot_i$, then T is i-unsatisfiable and we return the solution state with $S_i = \bot_i$ as the result. Otherwise, the composition $S \circ_i R$ is computed and used to similarly process the remaining formulas in T.

$$
\begin{aligned}
solveclose_i(S;\ \emptyset) &= S \\
solveclose_i(\bot_i;\ T) &= \bot_i \\
solveclose_i(S;\ \{a = b\} \cup T) &= solveclose_i(S', T), \\
&\qquad \text{where } S' = S \circ_i solve_i(vars(S))(S\langle\!\langle a \rangle\!\rangle_i = S\langle\!\langle b \rangle\!\rangle_i)
\end{aligned}
$$

To check i-validity, $\models_i (T \vdash c = d)$, it is sufficient to check that either $solveclose_i(id_T;\ T) = \bot$ or $S'\langle\!\langle c \rangle\!\rangle_i \equiv S'\langle\!\langle d \rangle\!\rangle_i$, where $S' = solveclose_i(id_T;\ T)$.

Soundness and Completeness. As with the congruence closure procedure, each step in $solveclose_i$ is i-conservative. Hence $solveclose_i$ is sound and complete: if $S' = solveclose_i(S;\ T)$, then for every i-model M and assignment ρ, $M, \rho \models S \cup T$ iff there is a ρ' extending ρ over the variables in $vars(S') - vars(S)$ such that $M, \rho' \models S'$. If $\sigma_i(S'[a]) \equiv \sigma_i(S'[b])$, then $M, \rho' \models a = S'[a] = \sigma_i(S'[a]) = \sigma_i(S'[b]) = S'[b] = b$, and hence $M, \rho \models a = b$. Otherwise, when $\sigma_i(S'[a]) \not\equiv \sigma_i(S'[b])$, we know by the condition on σ_i that there is an i-model M and an assignment ρ' such that $M[\![S'[a]]\!]\rho' \neq M[\![S'[b]]\!]\rho'$. The solved form S' divides the variables into independent variables x such that $S'(x) = x$, and dependent variables y where $y \neq S'(y)$ and the variables in $vars(S'(y))$ are all independent. We can therefore extend ρ' to an assignment ρ where the dependent variables y are mapped to $M[\![S'(y)]\!]\rho'$. Clearly, $M, \rho \models S'$, $M, \rho \models a = S'[a]$, and $M, \rho \models b = S'[b]$. Since S' i-preserves $(id_T;\ T)$, $M, \rho \models T$ but $M, \rho \not\models a = b$ and hence $T \vdash a = b$ is not i-valid, so the procedure is complete. The correctness argument is thus similar to that of Section 3 but for the case of a single Shostak theory considered here, there is no need to construct a canonical term model since $\models_i a = \sigma_i(a)$, and $\sigma_i(a) \equiv \sigma_i(b)$ iff $\models_i a = b$.

Canonical term model. The situation is different when we wish to combine Shostak theories. It is important to resolve potential semantic incompatibilities between two Shostak theories. With respect to some fixed notion of i-validity for θ_i and j-validity for θ_j with $i \neq j$, a formula A in the union of θ_i and θ_j may be satisfiable in an i-interpretation of only a specific finite cardinality for which there might be no corresponding satisfying j-interpretation for the formula. Such an incompatibility can arise even when a theory θ_i is extended with uninterpreted function symbols. For example, if ϕ is a formula with variables x and y that is satisfiable only in a two-element model M where $\rho(x) \neq \rho(y)$, then

the set of formulas Γ where $\Gamma = \{\phi, f(x) = x, f(u) = y, f(y) = x\}$ additionally requires $\rho(x) \neq \rho(u)$ and $\rho(y) \neq \rho(u)$. Hence, a model for Γ must have at least three elements, so that Γ is unsatisfiable. However, there is no way to detect this kind of unsatisfiability purely through the use of solving and canonization.

We introduce a canonical term model as a way around such semantic incompatibilities. The set of canonical i-terms a such that $\sigma_i(a) \equiv a$ yields a domain for a *term model* M_i where $M_i(f)(a_1, \ldots, a_n) = \sigma_i(f(a_1, \ldots, a_n))$. If M_i is (isomorphic to) an i-model, then we say that the theory θ_i is *composable*. Note that the *solve* operation is conservative with respect to the model M_i as well, since M_i is taken as an i-model.

Given the usual interpretation of disjunction, a notion of validity is said to be *convex* when $\models (T \vdash c_1 = d_1 \vee \ldots \vee c_n = d_n)$ implies $\models (T \vdash c_k = d_k)$ for some k, $1 \leq k \leq n$. If a theory θ_i is composable, then i-validity is convex. Recall that $\models_i (T \vdash c_1 = d_1 \vee \ldots \vee c_n = d_n)$ iff $\models_i (S \vdash c_1 = d_1 \vee \ldots \vee c_n = d_n)$ for $S = \mathit{solveclose}_i(id_T; T)$. If $S = \bot_i$, then $\models_i (T \vdash c_k = d_k)$, for $1 \leq k \leq n$. If $S \neq \bot_i$, then since S i-preserves T, $\models_i (S \vdash c_1 = d_1 \vee \ldots \vee c_n = d_n)$, but (by assumption) $\not\models_i (S \vdash c_k = d_k)$. An assignment ρ_S can be constructed so that for independent (i.e., where $S(x) = x$) variables $x \in \mathit{vars}(S)$, $\rho_S(x) = x$, and for dependent variables $y \in \mathit{vars}(S)$, $\rho_S(y) = M_i[\![S(y)]\!]\rho_S$. If for $S \neq \bot_i$, $\not\models_{\sigma_i} (S \vdash c_k = d_k)$, then $M_i, \rho_S \models S$ and $M_i, \rho_S \not\models c_k = d_k$. Hence $M_i, \rho_S \not\models (S \vdash c_k = d_k)$, for $1 \leq k \leq n$. This yields $M_i, \rho_S \not\models (T \vdash c_1 = d_1 \vee \ldots \vee c_n = d_n)$, contradicting the assumption.

5 Combining Shostak Theories

We now examine the combination of the theory of equality over uninterpreted function symbols with several disjoint Shostak theories. Examples of interpreted operations from Shostak theories include + and − from the theory of linear arithmetic, *select* and *update* from the theory of arrays, and *cons*, *car*, and *cdr* from the theory of lists. The basic Shostak combination algorithm covers the union of equality over uninterpreted function symbols and a single canonizable and solvable equational theory [Sho84,CLS96,RS01]. Shostak [Sho84] had claimed that the basic combination algorithm was sufficient because canonizers and solvers for disjoint theories could be combined into a single canonizer and solver for their union. This claim is incorrect.[3] We present a combined decision procedure for multiple Shostak theories that overcomes the difficulty of combining solvers.

Two theories θ_1 and θ_2 are said to be disjoint if they have no function symbols in common. A typical subgoal in a proof can involve interpreted symbols from several theories. Let σ_i be the canonizer for θ_i. A term $f(a_1, \ldots, a_n)$ is said to be in θ_i if f is in θ_i even though some a_i might contain function symbols outside θ_i. In processing terms from the union of pairwise disjoint theories $\theta_1, \ldots, \theta_N$,

[3] The difficulty with combining Shostak solvers was observed by Jeremy Levitt [Lev99].

it is quite easy to combine the canonizers so that each theory treats terms in the other theory as variables. Since σ_i is only applicable to i-terms, we first have to extend the canonizer σ_i to treat terms in θ_j for $j \neq i$, as variables. Let π_i be a chosen bijective set of equalities between the variables X and the set $\{a|(\exists j : j \neq i \wedge a \in \theta_j)\}$. We treat uninterpreted function symbols as belonging to a special theory θ_0 where $\sigma_0(a) = a$ for $a \in \theta_0$. The extended operation σ_i' is defined below.

$$\sigma_i'(a) = \pi_i[\sigma_i(a')], \text{ when } a' : \begin{array}{l} a' \text{ is an } i\text{-term}, \\ \pi_i[a'] \equiv a. \end{array}$$

Note that the *when* condition in the above definition can always be satisfied. The combined canonizer σ can then be defined as

$$\begin{aligned} \sigma(x) &= x \\ \sigma(f(a_1, \ldots, a_n)) &= \sigma_i'(f(\sigma(a_1), \ldots, \sigma(a_n))), \text{ when } i : f \text{ is in } \theta_i. \end{aligned}$$

This canonizer is, however, not used in the remainder of the paper.

We now discuss the difficulty of combining the solvers $solve_1$ and $solve_2$ for θ_1 and θ_2, respectively, into a single solver. The example uses the theory θ_A of linear arithmetic and the theory θ_L of the pairing and projection operations *cons*, *car*, *cdr*, where, somewhat nonsensically, the projection operations also apply to numerical expressions. Shostak illustrated the combination using the example

$$5 + car(x + 2) = cdr(x + 1) + 3.$$

Since the top-level operation on the left-hand side is $+$, we can treat $car(x + 2)$ and $cdr(x + 1)$ as variables and use $solve_A$. This might yield a partially solved equation of the form $car(x + 2) = cdr(x + 1) - 2$. Now since the top-level operation on the left-hand side is from the theory of lists, we use $solve_L$ to obtain $x + 2 = cons(cdr(x + 1) - 2, u)$ with a fresh variable u. We once again apply $solve_A$ to obtain $x = cons(cdr(x + 1) - 2, u) - 2$. This is, however, not in solved form: the left-hand side variable occurs in an interpreted context in its solution. There is no way to prevent this from happening as long as each solver treats terms from another theory as variables. Therefore the union of Shostak theories is not necessarily a Shostak theory.

The problem of combining disjoint Shostak theories actually has a very simple solution. There is no need to combine solvers. Since the theories are disjoint, the canonizer can tolerate multiple solutions for the same variable as long as there is at most one solution from any individual theory. This can be illustrated on the same example: $5 + car(x + 2) = cdr(x + 1) + 3$. By variable abstraction, we obtain the equation $v_3 = v_6$, where $v_1 = x + 2, v_2 = car(v_1), v_3 = v_2 + 5, v_4 = x + 1, v_5 = cdr(v_4), v_6 = v_5 + 3$. We can separate these equations out into the respective theories so that S is $(S_V;\ S_U;\ S_A;\ S_L)$, where S_V contains the variable equalities in canonical form, S_U is as in congruence closure but is always $\emptyset$ since there are no uninterpreted operations in this example, and S_A and S_L are the

solution sets for θ_A and θ_L, respectively. We then get $S_V = \{x = x, v_1 = v_1, v_2 = v_2, v_3 = v_6, v_4 = v_4, v_5 = v_5, v_6 = v_6\}$, $S_A = \{v_1 = x + 2, v_3 = v_2 + 5, v_4 = x + 1, v_6 = v_5 + 3\}$, and $S_L = \{v_2 = car(v_1), v_5 = cdr(v_4)\}$. Since v_3 and v_6 are merged in S_V, but not in S_A, we solve the equality between $S_A(v_3)$ and $S_A(v_6)$, i.e., $solve_A(v_2 + 5 = v_5 + 3)$ to get $v_2 = v_5 - 2$. This result is composed with S_A to get $\{v_1 = x + 2, v_3 = v_5 + 3, v_4 = x + 1, v_6 = v_5 + 3, v_2 = v_5 - 2\}$ for S_A. There are no new variable equalities to be propagated out of either S_A, S_L, or S_V. Notice that v_2 and v_5 both have different solved forms in S_A and S_L. This is tolerated since the solutions are from disjoint theories and the canonizer can pick a solution that is appropriate to the context. For example, when canonizing a term of the form $f(x)$ for $f \in \theta_i$, it is clear that the only relevant solution for x is the one from S_i.

We can now check whether the resulting solution state verifies the original equation $5 + car(x + 2) = cdr(x + 1) + 3$. In canonizing $f(a_1, \ldots, a_n)$ we return $S_V(y)$ whenever the term $f(S_i(S[\![a_1]\!]), \ldots, S_i(S[\![a_n]\!]))$ being canonized is such that $y = f(S_i(S[\![a_1]\!]), \ldots, S_i(S[\![a_n]\!])) \in S_i$ for $f \in \theta_i$. Thus $x+2$ canonizes to v_1 using S_A, and $car(v_1)$ canonizes to v_2 using S_L. The resulting term $5+v_2$, using the solution for v_2 from S_A, simplifies to $v_5 + 3$, which returns the canonical form v_6 by using S_A. On the right-hand side, $x + 1$ is equivalent to v_4 in S_A, and $car(v_4)$ simplifies to v_5 using S_L The right-hand side therefore simplifies to v_5+3 which is canonized to v_6 using S_A. The canonized left-hand and right-hand sides are identical.

We present a formal description of the procedure used informally in the above example. We show how *process* from Section 3 can be extended to combine the union of disjoint solvable, canonizable, composable theories. We assume that there are N disjoint theories $\theta_1, \ldots, \theta_N$. Each theory θ_i is equipped with a canonizer σ_i and solver $solve_i$ for i-terms. If we let I represent the interval $[1, N]$, then an I-model is a model M that is an i-model for each $i \in I$. We will ensure that each inference step is conservative with respect to I-models, i.e., I-conservative. We represent the uninterpreted part of S as S_0 instead of S_U. The solution state S of the algorithm now consists of a list of sets of equations $(S_V;\ S_0;\ S_1;\ \ldots;\ S_N)$. Here S_V is a set of variable equations of the form $x = y$, and S_0 is the set of equations of the form $x = f(x_1, \ldots, x_n)$ where f is uninterpreted. Each S_i is in i-solved form and is the solution set for θ_i.

Terms now contain a mixture of function symbols that are uninterpreted or are interpreted in one of the theories θ_i. A solution state S is *confluent* if for all $x, y \in dom(S_V)$ and i, $0 \leq i \leq N$: $S_V(x) \equiv S_V(y) \iff S_i(\{x\}) \cap S_i(\{y\}) \neq \emptyset$. A solution state S is canonical if it is confluent; S_V is functional and idempotent, i.e., $S_V \circ S_V = S_V$; the uninterpreted solution set S_0 is normalized, i.e., $S_0 \triangleright S_V = S_0$; each S_i, for $i > 0$, is functional, idempotent, i.e., $S_i \circ_i S_i = S_i$, normalized i.e., $S_i \triangleright S_V = S_i$, and in i-solved form. The canonization of expressions with respect to a canonical solution set S is defined as follows.

$$S[\![x]\!] \stackrel{\triangle}{=} S_V(x)$$

$$
\begin{aligned}
&\mathit{abstract}(S;\ x = y) = (S;\ x = y),\\
&\mathit{abstract}(S;\ a = b) = (S';\ a' = b'),\\
&\qquad\qquad \text{when } S', c, i:\ c \in \mathit{max}(\llbracket a = b \rrbracket_i),\\
&\qquad\qquad\qquad\qquad x \notin \mathit{vars}(S \cup a = b),\\
&\qquad\qquad\qquad\qquad S'_V = S_V \cup \{x = x\},\\
&\qquad\qquad\qquad\qquad S'_i = S_i \cup \{x = c\},\\
&\qquad\qquad\qquad\qquad S'_j = S_j,\ \text{for }, i \neq j\\
&\qquad\qquad\qquad\qquad a' = S'\llbracket a \rrbracket,\\
&\qquad\qquad\qquad\qquad b' = S'\llbracket b \rrbracket.
\end{aligned}
$$

Fig. 2. Variable abstraction step for multiple Shostak theories

$$
\begin{aligned}
S\llbracket f(a_1, \ldots, a_n) \rrbracket &\triangleq S_V(x),\ \text{when } i, x :\\
&\qquad i > 0, f \in \theta_i, x = S\{\{f(a_1, \ldots, a_n)\}\} \in S_i\\
S\llbracket f(a_1, \ldots, a_n) \rrbracket &\triangleq S\{\{f(a_1, \ldots, a_n)\}\},\ \text{otherwise.}
\end{aligned}
$$

$$
\begin{aligned}
S\{\{f(a_1, \ldots, a_n)\}\} &\triangleq \sigma'_i(f(S_i(S\llbracket a_1 \rrbracket), \ldots, S_i(S\llbracket a_n \rrbracket))),\\
&\qquad \text{if } f \in \theta_i, i > 0\\
S\{\{f(a_1, \ldots, a_n)\}\} &\triangleq f(S\llbracket a_1 \rrbracket, \ldots, S\llbracket a_n \rrbracket),\ \text{if } f \in \theta_0.
\end{aligned}
$$

Since variables are used to communicate between the different theories, the canonical variable x in S_V is returned when the term being canonized is known to be equivalent to an expression a such that $y = a$ in S_i, where $x \equiv S_V(y)$. The definition of the above global canonizer is one of the key contributions of this paper. This definition can be applied to the example above of computing $S\llbracket 5 + \mathit{car}(x + 2) \rrbracket$.

Variable Abstraction. The variable abstraction procedure $\mathit{abstract}(S; a = b)$ is shown in Figure 2. If a is an i-term such that $a \notin X$, then a is said to be a pure i-term. Let $\llbracket a = b \rrbracket_i$ represent the set of subterms of $a = b$ that are pure i-terms. A maximal 0-term is one of the form $f(x_1, \ldots, x_n)$ for $f \in \theta_0$. For $i > 0$, the set $\mathit{max}(M)$ of maximal terms in M is defined to be $\{a \in M | a \equiv b \vee a \notin \llbracket b \rrbracket,\ \text{for any } b \in M\}$. In a single variable abstraction step, $\mathit{abstract}(S;\ a = b)$ picks a maximal pure i-subterm c from the canonized input equality $a = b$, and replaces it with a fresh variable x from X while adding $x = c$ to S_i. By abstracting a maximal pure i-term, we ensure that S_i remains in i-solved form.

Explanation. The procedure in Figure 3 is similar to that of Figure 1. Equations from the input set T are processed into the solution state S of the form $S_V; S_0; \ldots, S_N$. Initially, S must be canonical. In processing the input equation $a = b$ into S, we take steps to systematically restore the canonicity of S. The first step is to compute the canonical form $S\llbracket a = b \rrbracket$ of $a = b$ with respect to S. It is easy to see that $(S; S\llbracket a = b \rrbracket)$ I-preserves $(S; a = b)$.

$$
\begin{aligned}
process(S;\ \emptyset) &= S \\
process(S;\ T) &= S, \text{ when } i : S_i = \bot_i \\
process(S;\ \{a = b\} \cup T) &= process(S';\ T), \text{ where} \\
&\qquad S' = close^*(merge_V(abstract^*(S;\ S[\![a = b]\!]))).
\end{aligned}
$$

$$
\begin{aligned}
close(S) &= S, \text{ when } i : S_i = \bot_i \\
close(S) &= S', \text{ when } S', i, x, y : \\
&\qquad x, y \in dom(S_V), \\
&\qquad (i > 0, S_V(x) \equiv S_V(y), S_i(x) \not\equiv S_i(y), \text{ and} \\
&\qquad\quad S' = merge_i(S;\ x = y)) \\
&\qquad \text{or} \\
&\qquad (i \geq 0, S_V(x) \not\equiv S_V(y), S_i(\{x\}) \cap S_i(\{y\}) \neq \emptyset, \text{ and} \\
&\qquad\quad S' = merge_V(S;\ S_V(x) = S_V(y))) \\
close(S) &= normalize(S), \text{ otherwise.}
\end{aligned}
$$

$$
normalize(S) = (S_V;\ S_0;\ S_1 \triangleright S_V;\ \ldots;\ S_N \triangleright S_V).
$$

$$
\begin{aligned}
merge_i(S; x = y) &= S', \text{ where } i > 0, \\
&\qquad S'_i = S_i \circ_i solve_i(vars(S_i))(S_i(x) = S_i(y)), \\
&\qquad S'_j = S_j, \text{ for } i \neq j, \\
&\qquad S'_V = S_V. \\
merge_V(S; x = x) &= S \\
merge_V(S; x = y) &= (S_V \circ R;\ S_0 \triangleright R;\ S_1;\ \ldots;\ S_N), \text{ where } R = orient(x = y).
\end{aligned}
$$

Fig. 3. Combining Multiple Shostak Theories

The result of the canonization step $a' = b'$ is then variable abstracted as $abstract^*(a' = b')$ (shown in Figure 2) so that in each step, a maximal, pure i-subterm c of $a' = b'$ is replaced by a fresh variable x, and the equality $x = c$ is added to S_i. This is also easily seen to be an I-conservative step. The equality $x = y$ resulting from the variable abstraction of $a' = b'$ is then merged into S_V and S_0. This can destroy confluence since there may be variables w and z such that w and z are merged in S_V (i.e., $S_V(w) \equiv S_V(z)$) that are unmerged in some S_i (i.e., $S_i(\{w\}) \cap S_i(\{z\}) = \emptyset$), or vice-versa.[4] The number of variables in $dom(S_V)$ remains fixed during the computation of $close^*(S)$. Confluence is restored by $close^*(S)$ which finds a pair of variables that are merged in some S_i but not in S_V, and merging them in S_V, or that are merged in S_V and not in some S_i and merging them in S_i. Each such merge step is also I-conservative. When this process terminates, S is once again canonical. The solution sets S_i are normalized with respect to S_V in order to ensure that the entries are in the normalized form for lookup during canonization.

[4] For $i > 0$, S_i is maintained in i-solved form and hence, $S_i(\{x\}) = \{x, S_i(x)\}$.

Invariants. As with congruence closure, several key invariants are needed to ensure that the solution state S is maintained in canonical form whenever it is given as the argument to *process*. If S is canonical and a and b are canonical with respect to S, then for $(S'; a' = b') = \mathit{abstract}(S;\ a = b)$, S' is canonical, and a' and b' are canonical with respect to S'. The state $\mathit{abstract}(S;\ a = b)$ I-preserves $(S;\ a = b)$. A solution state is said to be well-formed if S_V is functional and idempotent, S_0 is normalized, and each S_i is functional, idempotent, and in solved form. Note that if S is well-formed, confluent, and each S_i is normalized, then it is canonical. When S is well-formed, and $S' = \mathit{merge}_V(S; x = y)$ or $S' = \mathit{merge}_i(S; x = y)$, then S' is well-formed and I-preserves $(S; x = y)$. If S is well-formed and congruence-closed, and $S' = \mathit{normalize}(S)$, then S' is well-formed and each S'_i is normalized. If $S' = \mathit{normalize}(S)$, then each S'_i is in solved form because if x replaces y on the right-hand side of a solution set S_i, then $S_i(y) \equiv y$ since S_i is in i-solved form. By congruence closure, we already have that $S_i(x) \equiv S_i(y) \equiv y$. Therefore, the uniform replacement of y by x ensures that $S'_i(x) \equiv x$, thus leaving S in solved form. If $S' = \mathit{close}^*(S)$, where S is well-formed, then S' is canonical.

Variations. As with congruence closure, once S is confluent, it is safe to strengthen the normalization step to replace each S_i by $S_V[S_i]$. This renders S_0^{-1} functional, but S_i^{-1} may still be non-functional for $i > 0$, since it might contain left-hand side variables that are local. However, if $\hat{S}_i$ is taken to be S_i restricted to $\mathit{dom}(S_V)$, then $\hat{S}_i^{-1}$ with the strengthened normalization is functional and can be used in canonization. The solutions for local variables can be safely discarded in an actual implementation. The canonization and variable abstraction steps can be combined within a single recursion.

Termination. The operations $S[\![a = b]\!]$ and $\mathit{abstract}^*(S; a = b)$ are easily seen to be terminating. The operation $\mathit{close}^*(S)$ also terminates because the sum of the number of equivalence classes of variables in $\mathit{dom}(S_V)$ with respect to each of the solution sets $S_V, S_0, S_1, \ldots, S_N$, decreases with each *merge* operation.

Soundness and Completeness. We have already seen that each of the steps: canonization, variable abstraction, composition, merging, and normalization, is I-conservative. It therefore follows that if $S' = \mathit{process}(S; T)$, then S' I-preserves S. Hence, if $S'[\![c]\!] \equiv S'[\![d]\!]$, then clearly $\models_I (S' \vdash c = d)$, and hence $\models_I (S; T \vdash c = d)$.

The completeness argument requires the demonstration that if $S'[\![c]\!] \not\equiv S'[\![d]\!]$, then $\not\models_I (S' \vdash c = d)$ when S' is canonical. This is done by means of a construction of $M_{S'}$ and $\rho_{S'}$ such that $M_{S'}, \rho_{S'} \models S'$ but $M_{S'}, \rho_{S'} \not\models c = d$. The domain D consists of canonical terms e such that $S'[\![e]\!] = e$. As with congruence closure, $M_{S'}$ is defined so that $M_{S'}(f)(e_1, \ldots, e_n) = S'[\![f(e_1, \ldots, e_n)]\!]$. The assignment $\rho_{S'}$ is defined so that $\rho_{S'}(x) = S_V(x)$. By induction on c, we have that $M_{S'}[\![c]\!]\rho_{S'} = S'[\![c]\!]$. We can also easily check that $M_{S'}, \rho_{S'} \models S'$.

It is also the case that $M_{S'}$ is an I-model since $M_{S'}$ is isomorphic to M_i for each i, $1 \leq i \leq N$. This can be demonstrated by constructing a bijective

map μ_i between D and the domain D_i corresponding to M_i so that $\mu_i(x) = a'$, where $\pi_i[a'] = S_I(x)$, $\mu_i(f(a_1, \ldots, a_n)) = f(\mu_i(a_1), \ldots, \mu_i(a_n))$ if $f \in \theta_i$, and $\pi_i^{-1}(f(a_1, \ldots, a_n)$, otherwise. It can then be verified that for any $f \in \theta_i$ and terms $a_1, \ldots, a_n$ in D, $\mu_i(M_{S'}(f)(a_1, \ldots, a_n)) = M_i(f)(\mu_i(a_1), \ldots, \mu_i(a_n))$. This concludes the proof of completeness.

Convexity revisited. As in Section 4, the term model construction of $M_{S'}$ once again establishes that I-validity is convex. In other words, a sequent $\models_I (T \vdash c_1 = d_1 \vee \ldots \vee c_n = d_n)$ iff $\models_I (T \vdash c_k = d_k)$ for some k, $1 \leq k \leq n$.

6 Conclusions

Ground decision procedures for equality are crucial for discharging the myriad proof obligations that arise in numerous applications of automated reasoning. These goals typically contain operations from a combination of theories, including uninterpreted symbols. Shostak's basic method deals only with the combination of a single canonizable, solvable theory with equality over uninterpreted function symbols. Indeed, in all previous work based on Shostak's method, only the basic combination is considered. Though Shostak asserted that the basic combination was adequate to cover the more general case of multiple Shostak theories, this claim has turned out to be unsubstantiated. We have given here the first Shostak-style combination method for the general case of multiple Shostak theories. The algorithm is quite simple and is supported by straightforward arguments for termination, soundness, and completeness.

Shostak's combination method, as we have described it, is clearly an instance of a Nelson–Oppen combination [NO79] since it involves the exchange of equalities between variables through the solution set S_V. The added advantage of a Shostak combination is that it combines the canonizers of the individual theories into a global canonizer. The definition of such a canonizer for multiple Shostak theories is the key contribution of this paper. The technique of achieving confluence across the different solution sets is unique to our method. Confluence is needed for obtaining useful canonical forms, and is therefore not essential in a general Nelson–Oppen combination. The global canonizer $S[\![a]\!]$ can be applied to input formulas to discharge queries and simplify input formulas. The reduction to canonical form with respect to the given equalities helps keep the size of the term universe small, and makes the algorithm more efficient than a black box Nelson–Oppen combination. The decision algorithm for a Shostak theory given in Section 4 fits the requirements for a black box procedure that can be used within a Nelson–Oppen combination. The Nelson–Oppen combination of Shostak theories with other decision procedures has been studied by Tiwari [Tiw00], Barrett, Dill, and Stump [BDS02], and Ganzinger [Gan02], but none of these methods includes a general canonization procedure as is required for a Shostak combination.

Variable abstraction is also used in the combination unification procedure of Baader and Schulz [BS96], which addresses a similar problem to that of combining Shostak solvers. In our case, there is no need to ensure that solutions are compatible across distinct theories. Furthermore, variable dependencies can be cyclic across theories so that it is possible to have $y \in vars(S_i(x))$ and $x \in vars(S_j(y))$ for $i \neq j$. Our algorithm can be easily and usefully adapted for combining unification and matching algorithms with constraint solving in Shostak theories.

Insights derived from the Nelson–Oppen combination method have been crucial in the design of our algorithm and its proof. Our presentation here is different from that of our previous algorithm for the basic Shostak combination [RS01] in the use of variable abstraction and the theory-wise separation of solution sets. Our proof of the basic algorithm additionally demonstrated the existence of proof objects in a sound and complete proof system. This can easily be replicated for the general algorithm studied here. The soundness and completeness proofs given here are for composable theories and avoid the use of σ-models.

Our Shostak-style algorithm fits modularly within the Nelson–Oppen framework. It can be employed within a Nelson–Oppen combination (as suggested by Rushby [CLS96]) in which there are other decision procedures that generate equalities between variables. It is also possible to combine it with decision procedures that are not disjoint, as for example with linear arithmetic inequalities. Here, the existence of a canonizer with respect to equality is useful for representing inequality information in a canonical form. A variant of the procedure described here is implemented in ICS [FORS01] in exactly such a combination.

References

[BDL96] Clark Barrett, David Dill, and Jeremy Levitt. Validity checking for combinations of theories with equality. In Mandayam Srivas and Albert Camilleri, editors, *Formal Methods in Computer-Aided Design (FMCAD '96)*, volume 1166 of *Lecture Notes in Computer Science*, pages 187–201, Palo Alto, CA, November 1996. Springer-Verlag.

[BDS02] Clark W. Barrett, David L. Dill, and Aaron Stump. A generalization of Shostak's method for combining decision procedures. In A. Armando, editor, *Frontiers of Combining Systems, 4th International Workshop, FroCos 2002*, number 2309 in Lecture Notes in Artificial Intelligence, pages 132–146, Berlin, Germany, April 2002. Springer-Verlag.

[Bjø99] Nikolaj Bjørner. *Integrating Decision Procedures for Temporal Verification.* PhD thesis, Stanford University, 1999.

[BS96] F. Baader and K. Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. *J. Symbolic Computation*, 21:211–243, 1996.

[BTV02] Leo Bachmair, Ashish Tiwari, and Laurent Vigneron. Abstract congruence closure. *Journal of Automated Reasoning*, 2002. To appear.

[CLS96] David Cyrluk, Patrick Lincoln, and N. Shankar. On Shostak's decision procedure for combinations of theories. In M. A. McRobbie and J. K. Slaney, edi-

tors, *Automated Deduction—CADE-13*, volume 1104 of *Lecture Notes in Artificial Intelligence*, pages 463–477, New Brunswick, NJ, July/August 1996. Springer-Verlag.

[DST80] P.J. Downey, R. Sethi, and R.E. Tarjan. Variations on the common subexpressions problem. *Journal of the ACM*, 27(4):758–771, 1980.

[FORS01] J.-C. Filliâtre, S. Owre, H. Rueß, and N. Shankar. ICS: Integrated Canonization and Solving. In G. Berry, H. Comon, and A. Finkel, editors, *Computer-Aided Verification, CAV '2001*, volume 2102 of *Lecture Notes in Computer Science*, pages 246–249, Paris, France, July 2001. Springer-Verlag.

[FS02] Jonathan Ford and Natarajan Shankar. Formal verification of a combination decision procedure. In A. Voronkov, editor, *Proceedings of CADE-19*, Berlin, Germany, 2002. Springer-Verlag.

[Gan02] Harald Ganzinger. Shostak light. In A. Voronkov, editor, *Proceedings of CADE-19*, Berlin, Germany, 2002. Springer-Verlag.

[Kap97] Deepak Kapur. Shostak's congruence closure as completion. In H. Comon, editor, *International Conference on Rewriting Techniques and Applications, RTA '97*, number 1232 in Lecture Notes in Computer Science, pages 23–37, Berlin, 1997. Springer-Verlag.

[Koz77] Dexter Kozen. Complexity of finitely presented algebras. In *Conference Record of the Ninth Annual ACM Symposium on Theory of Computing*, pages 164–177, Boulder, Colorado, 2–4 May 1977.

[Lev99] Jeremy R. Levitt. *Formal Verification Techniques for Digital Systems.* PhD thesis, Stanford University, 1999.

[NO79] G. Nelson and D. C. Oppen. Simplification by cooperating decision procedures. *ACM Transactions on Programming Languages and Systems*, 1(2):245–257, 1979.

[NO80] G. Nelson and D. C. Oppen. Fast decision procedures based on congruence closure. *Journal of the ACM*, 27(2):356–364, 1980.

[RS01] Harald Rueß and Natarajan Shankar. Deconstructing Shostak. In *16th Annual IEEE Symposium on Logic in Computer Science*, pages 19–28, Boston, MA, July 2001. IEEE Computer Society.

[Sha01] Natarajan Shankar. Using decision procedures with a higher-order logic. In *Theorem Proving in Higher Order Logics: 14th International Conference, TPHOLs 2001*, volume 2152 of *Lecture Notes in Computer Science*, pages 5–26, Edinburgh, Scotland, September 2001. Springer-Verlag. Available at ftp://ftp.csl.sri.com/pub/users/shankar/tphols2001.ps.gz.

[Sho78] R. Shostak. An algorithm for reasoning about equality. *Comm. ACM*, 21:583–585, July 1978.

[Sho84] Robert E. Shostak. Deciding combinations of theories. *Journal of the ACM*, 31(1):1–12, January 1984.

[Tiw00] Ashish Tiwari. *Decision Procedures in Automated Deduction.* PhD thesis, State University of New York at Stony Brook, 2000.