Uniform Derivation of Decision Procedures by Superposition*

Alessandro Armando[1], Silvio Ranise[1,2], and Michaël Rusinowitch[2]

[1] DIST–Università degli Studi di Genova, via all'Opera Pia 13, 16145, Genova, Italy
{armando,silvio}@dist.unige.it
Phone: +39.010.353-2216 and Fax: +39.010.353-2948

[2] LORIA-INRIA-Lorraine, 615, rue du Jardin Botanique, BP 101,
54602 Villers les Nancy Cedex, France
{ranise,rusi}@loria.fr
Phone: (33) 03.83.59.30.20 and Fax: (33) 03.83.27.83.19

Abstract. We show how a well-known superposition-based inference system for first-order equational logic can be used almost directly as a decision procedure for various theories including lists, arrays, extensional arrays and combinations of them. We also give a superposition-based decision procedure for homomorphism.

Keywords: Automated Deduction, Equational Logic, Term Rewriting, Superposition, Decision Procedures, Lists, Arrays with Extensionality, Homomorphism

1 Introduction

In verification with proof assistants (such as PVS, COQ, HOL, and Nqthm), decision procedures are typically used for eliminating trivial subgoals represented for instance as sequents modulo a background theory. These theories axiomatize standard data-types such as arrays, lists, bit-vectors and have proved to be quite useful for, e.g., hardware verification. Elimination of trivial sequents often reduces to the **problem of proving the unsatisfiability of conjunctions of literals modulo a background theory** T, which is the problem we shall consider here.

The rewriting approach permits us the uniform design of decision procedures for eliminating these subgoals and also offers an efficient alternative to congruence closure techniques. This approach was inspired by Greg Nelson's thesis [Nel81] where it is suggested to apply Knuth-Bendix completion to derive decision procedures. Here, instead of the Knuth-Bendix completion procedure, we apply a standard complete superposition-based inference system for clausal equational logic (given for instance in [NR01]). This allows us not only to handle pure equality but also several interesting axiomatic theories that were not handled previously that way such as lists, arrays, and extensional arrays. The proof

* The authors would like to thank C. Ringeissen and L. Vigneron for their comments on a draft of this paper and the anonymous referees for helpful criticisms.

that the decision procedures are correct is straightforward w.r.t. other correctness proofs given in the literature (compare for instance our decision procedure for arrays with extensionality of Section 6 with [SDBL01]). In our approach, combining theories is also immediate. As an illustration, we show how to decide a combination of lists and arrays.

A second contribution of the paper is in the same spirit of applying Knuth-Bendix completion to derive a decision procedure for the theory of homomorphism. This is the first decision procedure, to our knowledge, for this theory.

Related work. For lack of space we only discuss results that are closely related to ours. In previous work, the rewriting approach was mainly used for pure equality theories. For instance, [BT00] focus on abstracting the control of congruence closure algorithms, in order to give a uniform presentation of several known algorithms. A recent extension to deal with equality modulo AC is presented in [BRTV00].

In [NO80], Nelson and Oppen describe a decision procedure for the "quantifier-free theory of the LISP list structure". The procedure is obtained as an extension of a congruence closure algorithm with a mechanism which augments the graph by selected instances of the axioms of the theory. The proof of correctness is model theoretic and seems difficult to generalize. A discussion of the difficulties of deriving a general method to obtain decision procedures by extending congruence closure algorithms as well as a decision procedure for the theory of arrays (without extensionality) can be found in [Nel81]. This discussion has motivated our work.

In [SDBL01], the first decision procedure for an extensional theory of arrays is presented. The key ingredient is a modified congruence closure algorithm which is capable of handling (so called) partial equations. The correctness proof is rather complex and it takes the main part of the paper; it is model-theoretic and rather *ad-hoc*. In Section 6, we give a decision procedure for the same theory considered in [SDBL01]. Our procedure is simpler to understand since it amounts to applying (almost directly) standard equality reasoning in contrast to handling partial equalities and our proof of correctness relies on basic properties of skolemization. As a consequence, the decision procedure (as well as its correctness proof) for the theory of arrays with extensionality can be adapted to similar presentations for sets and multisets.

Finally, we notice that we can easily derive decision procedures for combinations of theories in a manner closely resembling the combination schema described in [NO78]. This is exemplified for a combination of the theory of lists and arrays in Section 7. Furthermore, the decision procedures derived in our framework can be extended so to provide the interface functionalities needed for them to be plugged into the Nelson and Oppen combination schema [NO78].

2 Preliminaries

We assume the usual (first-order) syntactic notions of *signature*, (*ground*) *term*, *position*, *substitution*, *replacement*, *rewrite relation* $\rightarrow$, as defined, e.g., in [DJ90].

If Σ is a signature and X is a set of variables, then $T(\Sigma, X)$ denotes the set of terms built out of the symbols in Σ and the variables in X. $T(\Sigma)$ abbreviates $T(\Sigma, \emptyset)$. 0-ary function symbols are called *individual constants*. Let l and r be elements of $T(\Sigma, X)$, then $l = r$ is a $T(\Sigma, X)$-*equality* and $\neg(l = r)$ (also written as $l \neq r$) is a $T(\Sigma, X)$-*disequality*. A $T(\Sigma, X)$-*literal* is either a $T(\Sigma, X)$-equality or a $T(\Sigma, X)$-disequality, i.e. an expression of the form $s \bowtie t$ where $\bowtie$ is either $=$ or $\neq$. A $T(\Sigma, X)$-*clause* is a disjunction of literals, i.e. an expression of the form $\neg A_1 \vee \cdots \vee \neg A_n \vee B_1 \vee \cdots \vee B_m$ (abbreviated with $A_1, \ldots, A_n \Rightarrow B_1, \ldots, B_m$) where $A_1, \ldots, A_n$, $B_1, \ldots, B_m$ are $T(\Sigma, X)$-equalities ($n \geq 0$ and $m \geq 0$). We simply use the terms equality, disequality, literals, and clauses when $T(\Sigma, X)$ is clear from the context. A *flat equality* is an equality of the form $f(t_1, \ldots, t_n) = t_0$ or $t_0 = f(t_1, \ldots, t_n)$ where f is an n-ary function symbol and t_i is either a variable or an individual constant for $i = 0, 1, \ldots, n$ with $n \geq 0$. A *distinction* is a disequality $t_1 \neq t_2$, where t_i is either a variable or an individual constant for $i = 1, 2$. A *flat literal* is either a flat equality or a distinction. A *flat clause* is a disjunction of flat literals.

We assume the usual (first-order) notions of interpretation, satisfiability, validity, logical consequence (in symbols, $\models$), and theory (see, e.g., [End72]). Let S be a set of ground literals, then we say that S is T*-satisfiable* (T*-unsatisfiable*) iff $T \cup S$ is satisfiable (unsatisfiable, resp.). All the theories we shall consider in this paper contain the quantifier-free theory of equality $\mathcal{E}$.

Example 1. Assume that the axiom of T is $h(f(x, y)) = f(h(x), h(y))$ (where x and y are implicitly universally quantified variables). We can show the T-unsatisfiability of $\{h(c) = c',\ h(c') = c,\ f(c, c') = h(h(a)),\ f(c', c) = a,\ h(h(h(a))) \neq a\}$.

The *satisfiability problem for a theory* T amounts to establishing whether any given finite set of ground literals is T-satisfiable or not. A *decision procedure for* T is any algorithm that solves the satisfiability problem for T.

3 Our Approach

In this paper, we propose a uniform approach based on superposition inference rules to build decision procedures for a variety of decidable theories. For all theories T, **the first step is to flatten all the input literals**. The soundness of this preprocessing step is ensured by the following fact.

Lemma 1. *Let T be a $T(\Sigma, X)$-theory and S be a finite set of $T(\Sigma)$-literals. Then there exists a finite set of flat $T(\Sigma')$-literals S' (where Σ' is obtained from Σ by adding a finite number of individual constants) such that S' is T-satisfiable iff S is.*

Notice that flattening augments the size of the input set S of literals to $O(n)$, where n is the number of subterms in S.

Example 2. The following set of flat literals can be derived from the previous example: $\{h(c) = c',\ h(c') = c,\ f(c, c') = c_2,\ f(c', c) = a, h(a) = c_1, h(c_1) = c_2,\ h(c_2) = c_3, c_3 \neq a\}$.

Table 1. Inference rules of $\mathcal{SP}$

Name	**Rule**	**Applicability Conditions**
Superposition	$\dfrac{\Gamma \Rightarrow \Delta, l[u'] = r \quad \Pi \Rightarrow \Sigma, u = v}{\sigma(\Gamma, \Pi \Rightarrow \Delta, \Sigma, l[v] = r)}$	$\sigma(u) \not\preceq \sigma(v)$, $\sigma(u = v) \not\preceq \sigma(\Pi \cup \Sigma)$, $\sigma(l[u']) \not\preceq \sigma(r)$, $\sigma(l[u'] = r) \not\preceq \sigma(\Gamma \cup \Delta)$
Paramodulation	$\dfrac{\Gamma, l[u'] = r \Rightarrow \Delta \quad \Pi \Rightarrow \Sigma, u = v}{\sigma(l[v] = r, \Gamma, \Pi \Rightarrow \Delta, \Sigma)}$	$\sigma(u) \not\preceq \sigma(v)$, $\sigma(u = v) \not\preceq \sigma(\Pi \cup \Sigma)$, $\sigma(l[u']) \not\preceq \sigma(r)$, $\sigma(l[u'] = r) \not\prec \sigma(\Gamma \cup \Delta)$
Reflection	$\dfrac{\Gamma, u' = u \Rightarrow \Delta}{\sigma(\Gamma \Rightarrow \Delta)}$	$\sigma(u' = u) \not\prec \sigma(\Gamma \cup \Delta)$
Factoring	$\dfrac{\Gamma \Rightarrow \Delta, u = t, u' = t'}{\sigma(\Gamma, t = t' \Rightarrow \Delta, u = t')}$	$\sigma(u) \not\preceq \sigma(t)$,$\sigma(u) \not\preceq \sigma(\Gamma)$, $\sigma(u = t) \not\prec \sigma(\{u' = t'\} \cup \Delta)$

Table 2. Simplification rules of $\mathcal{SP}$

Name	**Rule**	**Applicability Conditions**
Subsumption	$\dfrac{S \cup \{C, C'\}}{S \cup \{C\}}$	for some substitution $\theta(C) \subseteq C'$, and there is no substitution ρ such that $\rho(C') = C$
Simplification	$\dfrac{S \cup \{C[l'], l = r\}}{S \cup \{C[\theta(r)], l = r\}}$	$l' = \theta(l)$, $\theta(l) \succ \theta(r)$, and $C[\theta(l)] \succ (\theta(l) = \theta(r))$
Deletion	$\dfrac{S \cup \{\Gamma \Rightarrow \Delta, t = t\}}{S}$	

We will make use of a superposition calculus, $\mathcal{SP}$, comprising the inference rules of Table 1 and the simplification rules of Table 2. $\mathcal{SP}$ is taken from [NR01]. It extends the system from [Rus91] by the *equality factoring rule* [BG94], so that more ordering restrictions are possible (in the non-Horn case). The relation $\succ$ is a reduction ordering [DJ90], which is total on ground terms. $\succ$ is extended to literals in the following way: $(a \bowtie b) \succ (c \bowtie d)$ if $\{a, b\} \succ\!\succ \{c, d\}$, where $\succ\!\succ$ is the multiset extension of $\succ$. Multisets of literals are compared using the multiset extension of $\succ$ on literals.

An inference system including simplification rules is refutationally complete if *any fair application of the rules to an unsatisfiable set of clauses will derive the empty clause.* Fairness means that if some inference is possible it will be performed at some step unless one of the parent clauses gets simplified, subsumed, or deleted. The calculus $\mathcal{SP}$ is known to be refutationally complete for general first-order equational logic [BG94,NR01]. (Note that for Horn clauses *Equality Factoring* is useless [KR91].) In Table 1 the substitution σ is the most general unifier of u and u', and u' is not a variable in *Superposition* and *Paramodulation*. We shall write Factoring instead of Equality Factoring for conciseness. In this paper, a *saturation* of a set of clauses by $\mathcal{SP}$ is the final set of clauses generated by a fair derivation from S using rules in $\mathcal{SP}$ with higher priority given to the simplification rules. If the saturation terminates for the union of T and any set of ground flat literals then it is a decision procedure for T: if the final set of clauses contains the empty clause then the input set of literals is unsatisfiable; it is satisfiable, otherwise. This is a direct consequence of the refutational completeness of

$\mathcal{SP}$. From now on, we shall call $\mathcal{SP}$ any fair application of the inference system with priority given to the simplification rules.

3.1 A Decision Procedure for the Quantifier-Free Theory of Equality

The following result says that $\mathcal{SP}$ can be used as a decision procedure for the quantifier-free theory of equality $\mathcal{E}$.[1] In fact, the decision procedure we obtain is just a variant of the Knuth-Bendix completion procedure (similar to the rational reconstruction of Nelson and Oppen's congruence closure algorithm of [BT00]). We shall assume now and in the remainder of this paper that the ordering $\succ$ **is s.t.** $t \succ c$ **for each constant** c **and for each ground term** t **that contains a symbol of arity greater than** 0. Note that it is easy to satisfy this requirement with a suitable precedence ordering.

Lemma 2. *Let S be a finite set of flat $T(\Sigma)$-literals. All the saturations of S by $\mathcal{SP}$ are finite.*

Proof. Note that *Simplification* is applicable whenever *Superposition* is. Hence *Superposition* is useless since *Simplification* has higher priority. *Simplification* and *Paramodulation* generate ground flat literals. *Reflection* generates the empty clause (which subsumes all other clauses). Since the number of possible ground flat literals is finite, it readily follows that all saturations are finite. □

Theorem 1. *$\mathcal{SP}$ is a decision procedure for $\mathcal{E}$.*

Let n be the size of the input set of flattened literals. Each *Simplification* or *Paramodulation* replaces a subterm by a $\succ$-smaller constant (i.e. a term of type $f(c_1, \ldots, c_n)$ or c' by some c). Hence the maximal number of inference steps is equal to the number of subterms times the number of constants in Σ, i.e. $O(n^2)$. Since finding a *Simplification* or *Paramodulation* inference is polynomial, the whole saturation is polynomial.

4 A Decision Procedure for the Theory of Lists

Let $\Sigma_{\mathcal{L}}$ be a signature containing the function symbols car (unary), cdr (unary), and cons (binary), and let $\mathcal{L}$ be the theory obtained by adding the following two axioms, denoted with $Ax(\mathcal{L})$, to $\mathcal{E}$:

$$\mathsf{car}(\mathsf{cons}(x, y)) = x \tag{1}$$

$$\mathsf{cdr}(\mathsf{cons}(x, y)) = y. \tag{2}$$

For simplicity, $\mathcal{L}$ is only a sub-theory of the "LISP list structure" considered in [NO80]. However, a decision procedure for such a theory can be derived by preprocessing the set of ground literals using the technique of [NO80] to eliminate negative occurrences of the predicate recognizing atoms and by applying $\mathcal{SP}$.

[1] We do not claim this result to be new; it is stated here only to give the flavor of our approach in the simple case of the pure equational theory.

Lemma 3. *Let S be a finite set of flat $T(\Sigma_{\mathcal{L}})$-literals. The clauses occurring in the saturations of $S \cup Ax(\mathcal{L})$ by $\mathcal{SP}$ can only be the empty clause, ground flat literals, or the equalities in $Ax(\mathcal{L})$.*

Proof. The proof is by induction on the length of the derivations. No inference between axioms in $Ax(\mathcal{L})$ is possible. Thus, by inspection of the rules in $\mathcal{SP}$, there are four cases to consider: (a) a *Simplification* between a ground flat equality and a ground flat literal,[2] (b) application of *Reflection* to a ground distinction, (c) a *Superposition* between an equality in $Ax(\mathcal{L})$ and a ground flat equality of the form $\mathsf{cons}(c_1, c_2) = c_3$ (where c_i is an individual constant for $i = 1, 2, 3$), or (d) a *Paramodulation* from a ground flat equality into a ground distinction. It is straightforward to verify that in case (a) only ground flat literals are generated, in case (b) the empty clause is generated, in case (c) ground flat equalities are generated, and finally in case (d) ground distinctions are generated. □

Lemma 4. *Let S be a finite set of flat $T(\Sigma_{\mathcal{L}})$-literals. All the saturations of $S \cup Ax(\mathcal{L})$ by $\mathcal{SP}$ are finite.*

Proof. By Lemma 3, we know that the saturations of $S \cup Ax(\mathcal{L})$ by $\mathcal{SP}$ can only contain the empty clause or ground flat literals. It is trivial to see that only a finite number of flat literals can be built out of a finite set of symbols and variables. □

Theorem 2. *$\mathcal{SP}$ is a decision procedure for $\mathcal{L}$.*

Let n be the size of the input set of flattened literals. At most $O(n^2)$ flat literals can be created by *Superposition* during saturation. The size of the current set of literals in a derivation is always bounded by a constant k which is $O(n^2)$. Other inferences take polynomial time in k according to Section 3.1. Hence overall the decision procedure is polynomial.

5 A Decision Procedure for the Theory of Arrays

Let $\Sigma_{\mathcal{A}}$ be a signature containing the function symbols select (binary) and store (ternary), and let $\mathcal{A}$ be the theory obtained by adding the following two axioms, denoted by $Ax(\mathcal{A})$, to $\mathcal{E}$:

$$\mathsf{select}(\mathsf{store}(a, i, e), i) = e \tag{3}$$

$$i \neq j \Rightarrow \mathsf{select}(\mathsf{store}(a, i, e), j) = \mathsf{select}(a, j) \tag{4}$$

(where a, i, j, and e are variables and (4) denotes $i = j \vee \mathsf{select}(\mathsf{store}(a, i, e), j) = \mathsf{select}(a, j)$). We shall assume that the ordering $\succ$ **is s.t. any term that contains** select **or** store **is $\succ$-bigger than all ground terms not containing them; moreover, all non constant symbols are greater than the constant ones**. Using an LPO ordering [DJ90], this can easily be ensured by a suitable precedence relation.

[2] Notice that *Superposition* can never apply to ground flat literals since *Simplification* has higher priority.

Lemma 5. *Let S be a finite set of flat $T(\Sigma_{\mathcal{A}})$-literals. The clauses occurring in the saturations of $S \cup Ax(\mathcal{A})$ by $\mathcal{SP}$ can only be:*

i) the empty clause; *ii) the axioms in $Ax(\mathcal{A})$;* *iii) ground flat literals;*
iv) clauses of the form $t \bowtie t' \vee c_1 = c'_1 \vee \cdots \vee c_n = c'_n$ where $c_1, c'_1, \ldots, c_n, c'_n$ ($n \geq 0$) are individual constants and $t \bowtie t'$ is either a distinction between two individual constants or an equality between individual constants or terms of the form $\mathsf{select}(c_i, i)$ (for some individual constants c and c_i);
v) clauses of the form $\mathsf{select}(c, x) = \mathsf{select}(c', x) \vee c_1 = k_1 \vee \cdots \vee c_n = k_n$, where k_i (for $i = 1, ..., n$) is either the variable x or is one among the individual constants $c, c', c_1, c'_1, \ldots, c_n, c'_n$ ($n \geq 0$).

Proof. The proof is by induction on the length of the derivations. The base case is simple and therefore omitted. By the induction hypothesis there are five types of clauses produced after n inference steps: *i)–v)*. For inferences with *Reflexion* or *Factoring* on one clause the result is obvious. *Deletion* and *Subsumption* do not create new clauses. For the sake of brevity, let *replacement* be either a *Superposition* or *Paramodulation* step. Let us consider inference steps involving two clauses. There are several cases to consider according to the categories the clauses belong to:

ii)-ii): A *Superposition* can be applied to the axioms in $Ax(\mathcal{A})$ but it generates the trivial clause $i = i \vee \mathsf{select}(a, i) = e$ which is immediately eliminated by *Deletion*. No new clause can be produced this way.
ii)-iii): A *Superposition* from a flat equality into axiom (3) produces a ground flat equality, i.e. a clause of type *iii)*, whereas a *Superposition* into axiom (4) produces a clause of type *v)*.
iii)-iii): The only possible inference is *Simplification* or *Paramodulation* between a ground flat equality and a ground flat literal. It produces only ground flat literals, i.e. a clause of type *iii)*.
iii)-iv): A replacement produces a clause of type *iv)*.
iii)-v): A replacement produces a clause of type *iv)* or *v)*.
iv)-iv): A replacement produces a clause of type *iv)*.
iv)-v): A replacement produces a clause of type *iv)*.
v)-v): A replacement produces a clause of type *iv)* or *v)*.

There are no possible inference between axioms and clauses of type *iv)* or *v)*. □

Lemma 6. *Let S be a finite set of flat $T(\Sigma_{\mathcal{A}})$-literals. All the saturations of $S \cup Ax(\mathcal{A})$ by $\mathcal{SP}$ are finite.*

The proof of this Lemma is analogous to that of Lemma 4 and therefore it is omitted.

Theorem 3. *$\mathcal{SP}$ is a decision procedure for $\mathcal{A}$.*

Let n be the size of the input set of flattened literals. At most $O(2^{n^k})$ clauses can be generated by saturation for some k (in fact $k = 2$). Hence the decision procedure takes time $O(2^{n^k})$.

Finally, it is worth noticing that the above decision procedure is similar to the algorithm described in [Nel81].

6 A Decision Procedure for the Theory of Arrays with Extensionality

Let $\mathcal{A}^s$ be the many-sorted version of the theory $\mathcal{A}$ of Section 5, i.e. the many-sorted theory with sorts ELEM, INDEX, and ARRAY, with function symbols store and select of type $\text{ARRAY}, \text{INDEX}, \text{ELEM} \longrightarrow \text{ARRAY}$ and $\text{ARRAY}, \text{INDEX} \longrightarrow \text{ELEM}$ respectively, and with the sorted version of (3) and (4) as axioms. (Notice that the use of sorts allows us to avoid problematic terms such as $\mathsf{store}(a, \mathsf{store}(a, i, e), \mathsf{select}(a, \mathsf{store}(a, i, e)))$.) Let $\mathcal{A}^s_e$ be the many-sorted theory of arrays with extensionality obtained from $\mathcal{A}^s$ by extending the set of axioms with

$$\forall i.(\mathsf{select}(a, i) = \mathsf{select}(b, i)) \Rightarrow a = b \tag{5}$$

where a and b are variables of sort ARRAY and i is a variable of sort INDEX (by abuse of notation, (5) denotes its clausal form). $\Sigma_{\mathcal{A}^e_s}$ denotes a signature containing the function symbols select, store, and a finite set of function symbols s.t. **if f is a function symbol of type $s_0, \ldots, s_{n-1} \longrightarrow s_n$ distinct from select and store, then s_i is either INDEX or ELEM, for all $i = 0, 1, ..., n$ and** $n \geq 1$. Furthermore, we assume that $\Sigma_{\mathcal{A}^e_s}$ admits at least one ground term for each sort, i.e. it is a sensible signature. Finally, let $Ax(\mathcal{A}^s)$ and $Ax(\mathcal{A}^s_e)$ be the set of axioms of $\mathcal{A}^s$ and of $\mathcal{A}^s_e$, respectively.

Lemma 7. *Let S be a set of $T(\Sigma_{\mathcal{A}^e_s})$-literals and let S' be obtained from S by replacing all the inequalities of the form $t \neq t'$ with $\exists i.\mathsf{select}(t, i) \neq \mathsf{select}(t', i)$, where t and t' are terms of sort* ARRAY. *Then S is $\mathcal{A}^s_e$-satisfiable iff S' is $\mathcal{A}^s$-satisfiable.*

Proof. We must show that $S \cup \mathcal{A}^s_e$ is satisfiable iff $S' \cup \mathcal{A}^s$ is or, equivalently, that $S \cup Ax(\mathcal{A}^s_e)$ is satisfiable iff $S' \cup Ax(\mathcal{A}^s)$ is. The 'only if' case is easy. For the 'if' case, let I be a (many-sorted) model of $S' \cup Ax(\mathcal{A}^s)$. We define the binary relation $\sim$ over ARRAY^I to hold whenever $\mathsf{select}^I(a, i) = \mathsf{select}^I(b, i)$ for all $i \in \text{INDEX}^I$, and we define $\sim$ over the INDEX^I and ELEM^I to be the identity relation. We now show that $\sim$ is a $\Sigma_{\mathcal{A}^s_e}$-congruence. It is clearly an equivalence. To prove that $\sim$ is a congruence it remains to show that if $a \sim b$, then $\mathsf{store}^I(a, i, e) \sim \mathsf{store}^I(b, i, e)$ for all $i \in \text{INDEX}^I$ and $e \in \text{ELEM}^I$.[3] Let us assume that $a \sim b$ but $\mathsf{store}^I(a, i, e) \not\sim \mathsf{store}^I(b, i, e)$ for some $i \in \text{INDEX}^I$ and $e \in \text{ELEM}^I$, i.e. that $\mathsf{select}^I(\mathsf{store}^I(a, i, e), k) \neq \mathsf{select}^I(\mathsf{store}^I(b, i, e), k)$ for some $i, k \in \text{INDEX}^I$ and $e \in \text{ELEM}^I$. There are two cases to consider. If $k = i$ then, since I is a model of (3), we can conclude that $e \neq e$, a contradiction. Otherwise (i.e. if $k \neq i$), since I is a model of (4), we can conclude that $\mathsf{select}^I(a, k) \neq \mathsf{select}^I(b, k)$. This is in contradiction with the assumption $a \sim b$. To conclude the proof, it is sufficient to check that $I' = I/\sim$ is a model of $S' \cup Ax(\mathcal{A}^s_e)$. □

[3] The case for select trivially follows from the definition of $\sim$. For a function symbol in $\Sigma_{\mathcal{A}^s_e}$ distinct from select and store, congruence immediately follows from the definition of $\sim$ and the properties of identity.

Lemma 8. *Let S be a conjunction of ground literals, then S is $\mathcal{A}^s$-satisfiable iff it is $\mathcal{A}$-satisfiable.*

The following theorem is the key of our reduction mechanism.

Theorem 4. *Let S be a set of $T(\Sigma_{\mathcal{A}_e^s})$-literals and let S' be obtained from S by replacing all the inequalities of the form $t \neq t'$ with $\mathsf{select}(t, sk(t,t')) \neq \mathsf{select}(t', sk(t,t'))$, where t and t' are terms of sort* ARRAY, *and sk is a Skolem function of type* ARRAY, ARRAY $\longrightarrow$ INDEX. *Then S is $\mathcal{A}_e^s$-satisfiable iff S' is $\mathcal{A}$-satisfiable.*

Proof. The Theorem readily follows from Lemma 7, Lemma 8, and basic properties of skolemization. □

A **decision procedure for the theory of arrays with extensionality** $\mathcal{A}_e^s$ is as follows. Given as input a finite set S of $T(\Sigma_{\mathcal{A}_e^s})$-literals, the procedure first replaces every occurrence of literals of the form $t \neq t'$ with $\mathsf{select}(t, sk(t,t')) \neq \mathsf{select}(t', sk(t,t'))$, where t and t' are terms of sort ARRAY, and sk is a Skolem function of type ARRAY, ARRAY $\longrightarrow$ INDEX. Then, it feeds the resulting set of literals to the decision procedure for $\mathcal{A}$ described in Section 5.

It is worth noticing that our decision procedure can be straightforwardly generalized to multi-dimensional arrays if we view them as arrays of arrays.

The worst-case time of the decision procedure for $\mathcal{A}_e^s$ is that of the procedure for $\mathcal{A}$, i.e. $O(2^{n^k})$ for a fixed natural number k, since the size of the set of input literals obtained by the pre-processing step described above is $O(n)$.

7 Combining Decision Procedures for Lists and Arrays

To emphasize the flexibility of our approach, we show how easy it is to combine the decision procedures for the theories of lists and arrays. Let $\Sigma_{\mathcal{U}}$ be a signature containing the function symbols select (binary), store (ternary), car (unary), cdr (unary), and cons (binary). Let $Ax(\mathcal{U})$ be the set of axioms obtained as the union of $Ax(\mathcal{A})$, $Ax(\mathcal{L})$, and $\mathcal{E}$. Furthermore, we shall assume that the simplification ordering $\succ$ (total on ground terms) satisfies the requirements of Section 5.

Lemma 9. *Let S be a finite set of ground flat $T(\Sigma_{\mathcal{U}})$-literals. The clauses occurring in the saturations of $S \cup Ax(\mathcal{U})$ by $\mathcal{SP}$ can only be of the type $i), iii), iv), v)$ given in Lemma 5, of the types given in Lemma 3, or elements of $Ax(\mathcal{U})$.*

Proof. Every *Superposition* or *Paramodulation* between axioms in $Ax(\mathcal{U})$ generate a clause that can be deleted. Hence the proof is as that of Lemma 3 and Lemma 5. □

Lemma 10. *Let S be a finite set of ground flat $T(\Sigma_{\mathcal{U}})$-literals. All the saturations of $S \cup Ax(\mathcal{U})$ by $\mathcal{SP}$ are finite.*

The proof of this Lemma is analogous to that of Lemma 4.

Theorem 5. *$\mathcal{SP}$ is a decision procedure for $\mathcal{U}$.*

8 A Decision Procedure for the Theory of Homomorphism

In this Section, we present an adaptation of the Knuth-Bendix completion procedure [KB70] to work modulo the theory of homomorphism. The completion process always terminates for ground equations and gives a decision procedure for this theory.[4]

Let $\Sigma_{\mathcal{H}}$ be a signature containing the unary function symbol h and let $\mathcal{H}$ be the theory obtained by adding instances of the following axiom schema, denoted with $Ax(\mathcal{H})$, to $\mathcal{E}$:

$$\mathsf{h}(f(x_1,\ldots,x_n)) = f(\mathsf{h}(x_1),\ldots,\mathsf{h}(x_n)) \tag{6}$$

where f is any n-ary function symbol ($n > 0$) in a subset Σ' of $\Sigma_{\mathcal{H}} \setminus \{\mathsf{h}\}$. We want to decide the $\mathcal{H}$-unsatisfiability of the set of ground literals ψ.

Example 3. $\{\mathsf{h}(c) = c', \mathsf{h}(c') = c, f(c,c') = \mathsf{h}(\mathsf{h}(a)), \mathsf{h}(\mathsf{h}(\mathsf{h}(a))) \neq a,\ f(c',c) = a\}$ is $\mathcal{H}$-unsatisfiable.

By Lemma 1, we can assume that ψ is a set of flat literals. Our decision procedure consists of two steps. First, we complete the set of ground equalities in ψ modulo $\mathcal{H}$ in order to get a rewrite system R. Second, for each inequality $s \neq t$ in ψ, we compute the normal form $s\downarrow_R$ of s and the normal form $t\downarrow_R$ of t (w.r.t. R). Then, if there exists an inequality $s' \neq t'$ in ψ s.t. $s'\downarrow_R$ is identical to $t'\downarrow_R$, ψ is $\mathcal{H}$-unsatisfiable; otherwise, ψ is $\mathcal{H}$-satisfiable.

8.1 Orientation

We introduce an ordering over ground terms which allows us to orient equalities as rewrite rules in such a way that a superposition between a ground equality and an equality in $Ax(\mathcal{H})$ can only generate a ground equality.

We first define a weight function on the symbols in $\Sigma_{\mathcal{H}}$, denoted with $[e]$ where e is in $\Sigma_{\mathcal{H}}$: $[c] = 1$, for each constant symbol c in $\Sigma_{\mathcal{H}}$; $[\mathsf{h}] = 0$; and $[f] = 1$, for f in $\Sigma_{\mathcal{H}}$ s.t. f is not a constant and f is not h. The weight of a ground term t, denoted with $[t]$, is the sum of the weight of the symbols (of $\Sigma_{\mathcal{H}}$) occurring in it. Then, we consider a total precedence $\succ$ on symbols s.t. $\mathsf{h} \succ f \succ c$, for all constant symbol c and all non constant symbol f distinct from h of $\Sigma_{\mathcal{H}}$. In the following $f^0(t)$ stands for t and $f^n(t)$ abbreviates $f(f^{n-1}(t))$ for $n > 1$, where f is a unary function symbol and t is any term. The ordering on ground terms we shall use is defined as follows (similarly to the Knuth-Bendix ordering [KB70]): $s \succ t$ iff

[4] Note that the word problem for ground Associative-Commutative (AC) theories is decidable [NR91] but for ground AC+Distributivity is undecidable [Mar92]. A direct modification of the proof of this last result would show that ground AC+Homomorphism is undecidable too.

1. $[s] > [t]$ or

2. $[s] = [t]$, s is of the form $f(s_1, \ldots, s_m)$, t is of the form $g(t_1, \ldots, t_n)$, and one of the following condition holds:

2.1. $f \succ g$

2.2. $f = g$, $m = n$ and $(s_1, \ldots, s_m) \succ\!\!\succ_{lex} (t_1, \ldots, t_m)$ (where $\succ\!\!\succ_{lex}$ denotes the lexicographic extension of $\succ$).

Lemma 11. *The relation $\succ$ is transitive, irreflexive, and monotonic (i.e. $s \succ t$ implies $f(.., s, ...) \succ f(..., t, ...)$, where f is in $\Sigma_{\mathcal{H}}$). Furthermore, $\succ$ is well-founded and it satisfies:*

- $f(c_1, ..., c_n) \succ \mathsf{h}^i(c_{n+1})$ *for all $i \geq 0$, all f that are not constants and are different from* h*,*
- $\mathsf{h}(f(x_1, ..., x_n)) \succ f(\mathsf{h}(x_1), ..., h(x_n))$ *for all ground terms x_i $(i = 1, ..., n)$, and*
- $\mathsf{h}^i(c) \succ \mathsf{h}^j(c')$ *for all $i > j$ and for all constants c, c' in $\Sigma_{\mathcal{H}}$.*

Proof. The lemma is proved in exactly the same way as for the Knuth-Bendix ordering [KB70].

We denote by $l \to r$ the rule obtained by orienting an equality $l = r$ when $l \succ r$. Given a rewrite system R, We shall sometimes write $s \downarrow_R t$ to express that t is the normal form of s by R.

8.2 Computation of Critical Pairs

Now, we are in the position to orient the equalities in ψ by means of the ordering $\succ$ defined in Section 8.1 and to perform a completion on the resulting set of rewrite rules using superposition rules. Unfortunately, with a naive approach, the number of rules generated by completion would be infinite. For instance, from $\mathsf{h}(c) = c$, $f(c, c') = c$, and $Ax(\mathcal{H})$ we can generate $f(c, \mathsf{h}^n(c')) = c$ for $n \geq 0$. To cope with this problem, we will consider any rewrite rule r as a rule scheme (denoted $Gen(r, R)$ or $Gen(r)$ and defined below) and we compute all superpositions between instances of two rule schemes in one step by using a special purpose inference rule (cf. *Homomorphism* rule below).

Some preliminary definitions and lemmas are mandatory. We define an *f-term* as a term with f as root symbol and for which the only other non-constant function symbol is h, where f can be any symbol in $\Sigma_{\mathcal{H}}$ (in particular, f can possibly be h). We define an *f-rule* as a rewrite rule with an f-term as left-hand side and an h-term or a constant symbol as right-hand side. For instance $f(c, \mathsf{h}^(c'))$ is an f-term and $f(c, \mathsf{h}^2(c')) = \mathsf{h}^3(c)$ or $f(c, \mathsf{h}^2(c')) = c$ is an f-rules. Examples of h-rules are $\mathsf{h}^2(c') = c$ or $\mathsf{h}^2(c') = \mathsf{h}(c)$.

In the following, let R_{h} be a convergent set of h-rules. We recall that Σ' is the subset of $\Sigma_{\mathcal{H}} \setminus \{\mathsf{h}\}$ such that if f of arity n is in Σ', then $\mathsf{h}(f(x_1, \ldots, x_n)) = f(\mathsf{h}(x_1), \ldots, \mathsf{h}(x_n))$ is in $Ax(\mathcal{H})$.

Lemma 12. *The set $R_{\mathsf{h}} \cup \{\mathsf{h}(f(x_1,\dots,x_n)) = f(\mathsf{h}(x_1),\dots,\mathsf{h}(x_n)) \mid f \in \Sigma'\}$ is convergent (we shall denote it by $R_{\mathsf{h}} \cup H$).*

Lemma 13. *Given constants c, c' and two h-terms $h^j(c), h^i(c')$, the set $\{n \mid n \in N,$ such that $h^n(h^j(c)) \rightarrow^*_{R_{\mathsf{h}}} h^i(c')\}$ is linear i.e. the union of a finite set of nonnegative integers and a finite set of arithmetic sequences. We denote it by $P_{j,c,i,c'}$.*

Proof. We may consider unary terms as words (for instance $\mathsf{h}^j(c)$ as $\mathsf{h}^j c$). Note that the set of ancestors $\{w | w \rightarrow^*_{R_{\mathsf{h}}} w'\}$ of a term w' by R_{h} can be effectively described by a context-free grammar. The set of h-terms with constant c is obviously regular. Hence the set of $\mathsf{h}^n\mathsf{h}^j c$ that reduces to $\mathsf{h}^i c'$ is the intersection of a regular language $\mathsf{h}^*\mathsf{h}^j c$ with a context-free language and therefore context-free. The set of lengths of words of a context-free language is linear.[5] □

Let J be the set of constants that do not occur in a left-hand side of R_{h}. If $c \notin J$ we say that c is *bounded* (in R_{h}).

Lemma 14. *Given an h-term $\mathsf{h}^j(c)$ and two constants c, c' s.t. c' is not bounded, the set $\{i \mid \exists n \in N, \mathsf{h}^n(\mathsf{h}^j(c)) \rightarrow_{R_{\mathsf{h}}} \mathsf{h}^i(c')\}$ is an interval $[u, \infty]$ denoted by $P_{j,c,\_,c'}$.*

Proof. Note that $\mathsf{h}^i(c')$ is R_{h}-irreducible. If there exists u, v with $\mathsf{h}^v(\mathsf{h}^j(c)) \downarrow_{R_{\mathsf{h}}} \mathsf{h}^u(c')$ then for all $g \in N$ we have $\mathsf{h}^{v+g}(\mathsf{h}^j(c)) \rightarrow^*_{R_{\mathsf{h}}} \mathsf{h}^{u+g}(c')$. □

Given an f-rule $r: \ f(t_1,\dots,t_n) \rightarrow t_{n+1}$, we define $\mathsf{h}^n(r) \downarrow_{R_{\mathsf{h}} \cup H}$ to be the rule $(\mathsf{h}^n(f(t_1,\dots,t_n)) \downarrow_{R_{\mathsf{h}} \cup H}) \ \rightarrow \ (\mathsf{h}^n(t_{n+1}) \downarrow_{R_{\mathsf{h}} \cup H})$. By the convergence of $R_{\mathsf{h}} \cup H$ this is well defined.

Definition 1. *For $f \in \Sigma'$, we define $Gen(r, R_{\mathsf{h}})$ as the set $\{\mathsf{h}^n(r) \downarrow_{R_{\mathsf{h}} \cup H} \mid n \in N\}$ where r denotes any f-rule $f(t_1,\dots,t_n) \rightarrow t_{n+1}$. For $f \notin \Sigma'$ we define $Gen(r, R_{\mathsf{h}}) = \{r\}$. We shall omit the argument R_{h} in Gen when it is clear from the context.*

Now, we derive a finite description for $Gen(r, R_{\mathsf{h}})$. We first classify the elements in $Gen(r)$ according to their bounded arguments. More specifically we introduce the equivalence relation $\sim$ on f-rules in $Gen(r)$:

Definition 2. *Given two normalized (by R_{h}) rules $r_1 : f(\mathsf{h}^{l_1}(c_1), ..., \mathsf{h}^{l_n}(c_n)) \rightarrow \mathsf{h}^{l_{n+1}}(c_{n+1})$ and $r_2 : f(\mathsf{h}^{j_1}(d_1), \dots, \mathsf{h}^{j_n}(d_n)) \rightarrow \mathsf{h}^{j_{n+1}}(d_{n+1})$ and such that $r_1, r_2 \in Gen(r)$, we have $r_1 \sim r_2$ iff for all k, $c_k = d_k$ and for all $c_k \notin J$, $l_k = j_k$.*

For instance if $R_{\mathsf{h}} = \{\mathsf{h}(c) \rightarrow c\}$ then $(g(\mathsf{h}^3(c'), c) = \mathsf{h}^2(c')) \ \sim \ (g(\mathsf{h}^2(c'), c) = \mathsf{h}^3(c'))$. We have the following simple lemma:

Lemma 15. *The equivalence $\sim$ defined on $Gen(r)$ has finite index (i.e. the number of classes is finite).*

[5] For details, see ex. 6.8 at page 142 of [UAH74].

Proof. Simple and therefore omitted.

We are now in the position to give a finite representation for the equivalence class of a rule r' in $Gen(r)$

Definition 3. *Let r be an f-rule $r : f(\mathsf{h}^{l_1}(c_1), \ldots, \mathsf{h}^{l_n}(c_n)) = \mathsf{h}^{l_{n+1}}(c_{n+1})$ and $r' : f(\mathsf{h}^{j_1}(d_1), \ldots, \mathsf{h}^{j_n}(d_n)) = \mathsf{h}^{j_{n+1}}(d_{n+1})$. Then, we define $C_{r,r'} = \{r" \in Gen(r) \mid r' \sim r"\}$.*

Let us compute $C_{r,r'}$ more explicitly. We introduce

$$P_{r,r'} = \Big(\bigcap_{\substack{1 \le m \le n+1 \\ d_m \in J}} P_{l_m, c_m, j_m, d_m} \Big) \cap \Big(\bigcap_{\substack{1 \le m \le n+1 \\ d_m \notin J}} P_{l_m, c_m, -, d_m} \Big)$$

Let $p_{r,r'}$ be the minimal element of $P_{r,r'}$. Note that $p_{r,r'}$ is computable since it can be defined by a formula of Presburger arithmetic:

$$P_{r,r'}(x) \wedge (\forall\ y\ P_{r,r'}(y) \Rightarrow x \le y)$$

We denote by $n(p, l, c, d)$ the natural number n (when it exists) such that $\mathsf{h}^p(\mathsf{h}^l(c)) \downarrow_{R_\mathsf{h}} \mathsf{h}^n(d)$. Then

$$\begin{aligned} C_{r,r'} = \{\ & f(\mathsf{h}^{t_1}(d_1), \ldots, \mathsf{h}^{t_n}(d_n)) = \mathsf{h}^{t_{n+1}}(d_{n+1}) \mid \text{for } 1 \le m \le n+1 \\ & t_m = j_m \text{ if } d_m \notin J \text{ and} \\ & t_m = p' - p_{r,r'} + n(p_{r,r'}, l_m, c_m, d_m) \text{ if } d_m \in J \text{ where } p' \in P_{r,r'} \} \end{aligned}$$

We define the size of an h-rule $\mathsf{h}^a(b) \to \mathsf{h}^c(d)$ to be $a + c$. By reduction to Presburger arithmetic, we can prove the following fact.

Lemma 16. *Given two f-rules r_1, r_2, the minimal non-trivial critical pairs between rules in $Gen(r_1)$ and $Gen(r_2)$, are computable.*

8.3 Completion Procedure

We now give the three inference rules defining the binary transition relation over sets of equalities (denoted with $\vdash$), which models our completion procedure (modulo $\mathcal{H}$). The first is the *Deletion* rule of Table 2. The second is the *Simplification* rule, obtained as an instance for unit clauses of the *Simplification* rule of Table 2 (i.e. $E \cup \{l[s] = r, s = t\} \vdash E \cup \{l[t] = r, s = t\}$, if $l[s] \succ r$ and $s \succ t$). The third is a special purpose inference which allows us to take into account finitely many selected instances of the axioms in $Ax(\mathcal{H})$ which suffices for correctness.

$$\textit{Homomorphism} : E \cup \{r_1, r_2\} \vdash E \cup \{r_1, r_2, h_1, \ldots, h_k\}$$

where the r_i are f-rules and the h_j are the minimal critical pairs of $Gen(r_1, R_\mathsf{h})$ and $Gen(r_2, R_\mathsf{h})$. We recall that by Lemma 1, we assume that the initial set of rules is *flat* , which means by definition that the arguments of the non-constant symbols are constants.

Lemma 17. *When initially given a set of flat rules, inference rules Simplification and Homomorphism only generate equations of type* $f(\mathsf{h}^{i_1}(c_1), ..., \mathsf{h}^{i_n}(c_n)) = \mathsf{h}^{i_{n+1}}(c_{n+1})$ *or of type* $\mathsf{h}^i(c) = \mathsf{h}^{i'}(c')$.

Theorem 6. *Completion with priority given to rule Simplification always terminates.*

Proof. Note that any sequence of *Simplification* applications always terminates. Let $E_0, E_1, E_2 \ldots$ be an infinite derivation such that E_i is the result of applying *Homomorphism* to E_{i-1} followed by a maximal sequence of *Simplification* applications. We assume that the set of constants is $\{c_1, \ldots, c_k\}$. Let $M_j = (m_1^j, \ldots, m_k^j)$ be the exponents of h in the h-rules of E_j. That is, if there is a rule in E_j with left-hand side $\mathsf{h}^m(c_i)$ then $m_i^j = m$. Note that there are no two rules of this type for the same constant c_i (otherwise one simplifies another) and therefore the vector M_j is well-defined. When no rule exists we put ∞ as a coordinate with $n < \infty$ for all integers.

The component-wise ordering on vectors M_j is well-founded and we always have $M_j \leq M_{j-1}$. Hence after some finite number of steps the left-hand sides of h-rules remain the same. Also the right-hand sides of rules may be simplified but only finitely many time (the reduction relation is well-founded too) Finally after some finite number of steps the set of h-rules is constant. Note also that this subset of rules is canonical. We shall denote it by R_h. In particular at most one rule applies to an h-term $\mathsf{h}^n(c)$.

Homomorphism generates only h-rules. Hence after a finite number of steps, say K, it will not produce any new rule. Note that the arguments of left-hand sides of f-rules are of type $\mathsf{h}^i(c_j)$ with $i < M_K(j)$ when c_j is bounded. □

Theorem 7. *Let E be the final finite set of rules obtained by the terminating completion procedure above. Let R_h be the final set of* h *rules in E. Then, $\overline{E} \cup H$ is convergent where $\overline{E}$ is the union of all sets $Gen(r, R_\mathsf{h})$ for all r in E.*

Corollary 1. *Given a set of ground equations E_0, and the set E derived from E_0 by completion then $E_0 \cup H \models a = b$ iff $a \downarrow_{\overline{E} \cup H} = b \downarrow_{\overline{E} \cup H}$.*

9 Conclusions and Future Work

We have shown how to apply a generic inference system to derive decision procedures for the theories of lists, arrays, arrays with extensionality, and combinations of them. A decision procedure (based on superposition) for the theory of homomorphism has been presented for the first time.

We envisage two main directions for future research. Firstly, our approach might be extended using different automated deduction techniques from e.g. [CP95,Lei90]. Secondly, we want to investigate possible cross-fertilizations with techniques used in heuristic theorem provers to effectively incorporating decision procedures, see e.g. [AR01].

References

AR01. A. Armando and S. Ranise. A Practical Extension Mechanism for Decision Procedures: the Case Study of Universal Presburger Arithmetic. *J. of Universal Computer Science*, 7(2):124–140, February 2001.

BG94. L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *J. of Logic and Comp.*, 4(3):217–247, 1994.

BRTV00. L. Bachmair, I. V. Ramakrishnan, A. Tiwari, and L. Vigneron. Congruence closure modulo associativity and commutativity. In *Frontiers of Comb. Sys.'s (FroCos'2000)*, LNCS 1794, pages 245–259, 2000.

BT00. L. Bachmair and A. Tiwari. Abstract congruence closure and specializations. In D. A. McAllester, editor, *17th CADE*, LNAI 1831, pages 64–78, 2000.

CP95. R. Caferra and Peltier. Decision procedures using model building techniques. In *CSL: 9th Workshop on Computer Science Logic.* LNCS 1092, 1995.

DJ90. N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Hand. of Theoretical Comp. Science*, pages 243–320. 1990.

End72. H. B. Enderton. *A Mathematical Introduction to Logic.* Academic Pr., 1972.

KB70. D. E. Knuth and P. E. Bendix. Simple word problems in universal algebra. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297, Oxford, 1970. Pergamon Press.

KR91. E. Kounalis and M. Rusinowitch. On Word Problems in Horn Theories. *JSC*, 11(1&2):113–128, January/February 1991.

Lei90. A. Leitsch. Deciding horn classes by hyperresolution. In *CSL: 3rd Workshop on Computer Science Logic.* LNCS, 1990.

Mar92. C. Marché. The word problem of ACD-ground theories is undecidable. *International Journal of Foundations of Computer Science*, 3(1):81–92, 1992.

Nel81. G. Nelson. Techniques for Program Verification. Technical Report CSL-81-10, Xerox Palo Alto Research Center, June 1981.

NO78. G. Nelson and D.C. Oppen. Simplification by Cooperating Decision Procedures. Technical Report STAN-CS-78-652, Stanford CS Dept., April 1978.

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

NR91. P. Narendran and M. Rusinowitch. Any ground associative-commutative theory has a finite canonical system. In *4th RTA Conf., Como (Italy)*, 1991.

NR01. R. Nieuwenhuis and A. Rubio. Paramodulation-based theorem proving. In A. Robinson and A. Voronkov, editors, *Hand. of Automated Reasoning.* 2001.

Rus91. M. Rusinowitch. Theorem-proving with Resolution and Superposition. *JSC*, 11(1&2):21–50, January/February 1991.

SDBL01. A. Stump, D. L. Dill, C. W. Barrett, and J. Levitt. A Decision Procedure for an Extensional Theory of Arrays. In *LICS'01*, 2001. To appear.

UAH74. J. D. Ullman, A. V. Aho, and J. E. Hopcroft. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, 1974.