# Deciding Extensions of the Theory of Arrays by Integrating Decision Procedures and Instantiation Strategies

Silvio Ghilardi[1], Enrica Nicolini[2], Silvio Ranise[1,3], and Daniele Zucchelli[1,3]

[1] Dipartimento di Informatica, Università degli Studi di Milano (Italia)
[2] Dipartimento di Matematica, Università degli Studi di Milano (Italia)
[3] LORIA & INRIA-Lorraine, Nancy (France)

**Abstract.** The theory of arrays, introduced by McCarthy in his seminal paper "Towards a mathematical science of computation", is central to Computer Science. Unfortunately, the theory alone is not sufficient for many important verification applications such as program analysis. Motivated by this observation, we study extensions of the theory of arrays whose satisfiability problem (i.e. checking the satisfiability of conjunctions of ground literals) is decidable. In particular, we consider extensions where the indexes of arrays have the algebraic structure of Presburger Arithmetic and the theory of arrays is augmented with axioms characterizing additional symbols such as dimension, sortedness, or the domain of definition of arrays.
We provide methods for integrating available decision procedures for the theory of arrays and Presburger Arithmetic with automatic instantiation strategies which allow us to reduce the satisfiability problem for the extension of the theory of arrays to that of the theories decided by the available procedures. Our approach aims to reuse as much as possible existing techniques so to ease the implementation of the proposed methods. To this end, we show how to use both model-theoretic and rewriting-based theorem proving (i.e., superposition) techniques to implement the instantiation strategies of the various extensions.

## 1 Introduction

Since its introduction by McCarthy in [13], the theory of arrays ($\mathcal{A}$) has played a very important role in Computer Science. Hence, it is not surprising that many papers [4,17,20,10,12,19,2,3] have been devoted to its study in the context of verification and many reasoning techniques, both automatic - e.g., [2] - and manual [17], have been developed to reason in such a theory.

Unfortunately, as many previous works [20,10,12,3] have already observed, $\mathcal{A}$ alone or even extended with extensional equality between arrays (as in [19,2]) is not sufficient for many applications of verification. For example, the works in [20,10,12] tried to extend the theory to reason about sorted arrays. More recently, Bradley et al. [3] have shown the decidability of the satisfiability problem for a

M. Fischer et al. (Eds.): JELIA 2006, LNAI 4160, pp. 177-189, 2006.

restricted class of (possibly quantified) first-order formulae that allows one to express many important properties about arrays.

In this paper, we consider the theory of arrays with extensionality [19,2] whose indexes have the algebraic structure of Presburger Arithmetic ($\mathcal{P}$), and extend it with additional (function or predicate) symbols expressing important features of arrays (e.g., the dimension of an array or an array being sorted). The *main contribution* of the paper is a method to integrate two decision procedures, one for the theory of arrays without extensionality ($\mathcal{A}$) and one for $\mathcal{P}$, with instantiation strategies that allow us to reduce the satisfiability problem of the extension of $\mathcal{A} \cup \mathcal{P}$ to the satisfiability problems decided by the two available procedures.

Our approach to integrating decision procedures and instantiation strategies is inspired by model-theoretic considerations and by the rewriting-approach [2,1,11]. For the rewriting-based method, we follow the lines of [11], where, facing the satisfiability problem, it is suggested that the (ground) formulae derived by the superposition calculus [16] between ground literals and the axioms of a theory $T$ (extending the theory of equality $Eq$) can be passed to a decision procedure for $Eq$. In this paper, we use superposition to generate enough (ground) instances of an extension of $\mathcal{A}$ so to enable the decision procedures for $\mathcal{P}$ and $\mathcal{A}$ to decide its satisfiability problem. An immediate by-product of our approach is the fact that the various extensions of $\mathcal{A}$ can be combined together to decide the satisfiability of the union of the various extensions.

*Related work.* The work most closely related to ours is [3]. The main difference is that we have a semantic approach to extending $\mathcal{A}$ since we consider only the satisfiability of ground formulae and we introduce additional functions and predicates while in [3], a syntactic characterization of a class of full first-order formulae, which turns out to be expressive enough to specify many properties of interest about arrays, is considered. Our approach allows us to get a more refined characterization of some properties of arrays, yielding the decidability of the extension of $\mathcal{A}$ with injective arrays (see Section 5.1), which is left as an open problem in [3].

Our instantiation strategy based on superposition (see Section 5.2) has a similar spirit of the work in [7], where equational reasoning is integrated in instantiation-based theorem proving. The main difference with [7] is that we solve the state-explosion problem, due to the recombination of formulae caused by the use of standard superposition rules, by deriving a new termination result for an extension of $\mathcal{A}$ as recommended by the rewriting approach to satisfiability procedures of [2].

*Plan of the paper.* Section 2 introduces some formal notions necessary to develop the results in this paper. Section 3 gives some motivation for the first extension of $\mathcal{A}$ by a dimension function together with its formal definition while Section 4 describe an extensible decision procedure. Section 5 considers two extensions of the theory defined in Section 3. For lack of space, further extensions of $\mathcal{A}$ and the proofs of the results in this paper are included in a Technical Report [9].

## 2 Formal Preliminaries

We work in *many-sorted first-order logic with equality* and we assume the basic syntactic and semantic concepts as in, e.g., [6].

A *signature* $\Sigma$ is a non-empty set of sort symbols together with a set of function symbols and a set of predicate symbols (both equipped with suitable lists of sort symbols as arity). The set of predicate symbols contains a symbol $=_S$ for equality for every sort $S$ (we usually omit its subscript). If $\Sigma$ is a signature, a *simple* expansion of $\Sigma$ is a signature $\Sigma'$ obtained from $\Sigma$ by adding a set $\underline{a} := \{a_1, ..., a_n\}$ of "fresh" constants (each of them again equipped with a sort), i.e. $\Sigma' := \Sigma \cup \underline{a}$, where $\underline{a}$ is such that $\Sigma$ and $\underline{a}$ are disjoint. Below, we write $\Sigma^{\underline{a}}$ as the simple expansion of $\Sigma$ with a set $\underline{a}$ of fresh constant symbols.

First-order terms and formulae over a signature $\Sigma$ are defined in the usual way, i.e., they must respect the arities of function and predicate symbols and the variables occurring in them must also be equipped with sorts (well-sortedness). A *$\Sigma$-atom* is a predicate symbol applied to (well-sorted) terms. A *$\Sigma$-literal* is a $\Sigma$-atom or its negation. A *ground* literal is a literal not containing variables. A *constraint* is a finite conjunction $\ell_1 \wedge \cdots \wedge \ell_n$ of literals, which can also be seen as a finite set $\{\ell_1, \ldots, \ell_n\}$. A *$\Sigma$-sentence* is a first-order formula over $\Sigma$ without free variables.

A *$\Sigma$-structure* $\mathcal{M}$ consists of non-empty and pairwise disjoint domains $S^{\mathcal{M}}$ for every sort $S$, and interprets each function symbol $f$ and predicate symbol $P$ as functions $f^{\mathcal{M}}$ and relations $P^{\mathcal{M}}$, respectively, according to their arities. If $t$ is a ground term, we also use $t^{\mathcal{M}}$ for the element denoted by $t$ in the structure $\mathcal{M}$. Validity of a formula $\phi$ in a $\Sigma$-structure $\mathcal{M}$ (in symbols, $\mathcal{M} \models \phi$), satisfiability, and logical consequence are defined in the usual way. The $\Sigma$-structure $\mathcal{M}$ is a *model* of the $\Sigma$-theory $T$ iff all axioms of $T$ are valid in $\mathcal{M}$. A *$\Sigma$-theory* $T$ is a (possibly infinite) set of $\Sigma$-sentences. Let $T$ be a theory. We refer to the signature of $T$ as $\Sigma_T$. If there exists a set $Ax(T)$ of sentences in $T$ such that every formula $\phi$ of $T$ is a logical consequence of $Ax(T)$, then we say that $Ax(T)$ is a set of axioms of $T$. A theory $T$ is *complete* iff, given a sentence $\phi$, we have that $\phi$ is either true or false in all the models of $T$.

In this paper, we are concerned with the *(constraint) satisfiability problem* for a theory $T$, also called the $T$-satisfiability problem, which is the problem of deciding whether a $\Sigma_T$-constraint is satisfiable in a model of $T$. Notice that a constraint may contain variables: since these variables may be equivalently replaced by free constants, we can reformulate the constraint satisfiability problem as the problem of deciding whether a finite conjunction of ground literals in a simply expanded signature $\Sigma_T^{\underline{a}}$ is true in a $\Sigma_T^{\underline{a}}$-structure whose $\Sigma_T$-reduct is a model of $T$. We say that a $\Sigma_T$-constraint is *$T$-satisfiable* iff there exists a model of $T$ satisfying it. Two $\Sigma_T$-constraints $\phi$ and $\psi$ are $T$-equisatisfiable iff there exists a structure $\mathcal{M}_1$ such that $\mathcal{M}_1 \models T \wedge \phi$ iff the following condition holds: there exists a structure $\mathcal{M}_2$ such that $\mathcal{M}_2 \models T \wedge \psi$.

Without loss of generality, when considering a set $L$ of ground literals to be checked for satisfiability, we may assume that each literal $\ell$ in $L$ is *flat*, i.e. $\ell$ is required to be either of the form $a = f(a_1, \ldots, a_n)$, $P(a_1, \ldots, a_n)$, or

$\neg P(a_1, \ldots, a_n)$, where $a, a_1, \ldots, a_n$ are (sort-conforming) constants, $f$ is a function symbol, and $P$ is a predicate symbol (possibly also equality).

## 3 Finite Arrays with Dimension as a Combined Theory

Given a set $A$, by $Arr(A)$ we denote the set of finite arrays with natural numbers as indexes and whose elements are from $A$. We model such an array $a$ as a sequence $a : \mathbb{N} \longrightarrow A \cup \{\bot\}$ which is eventually equal to $\bot$ (here $\bot$ is an element not in $A$ denoting an "undefined" or "default" value). In this way, for every array $a \in Arr(A)$ there is a smallest index $n \geq 0$, called the *dimension* of $a$, such that the value of $a$ at index $j$ is equal to $\bot$ for $j \geq n$. Contrary to finite sequences, we do not require that any value of $a$ at $k < n$ be distinct from $\bot$: this is also the reason to use the word 'dimension' rather than 'length', as for sequences. There is just one array whose dimension is zero which we indicate by $\varepsilon$ and call it the *empty* array. Since many applications of verification require arithmetic expressions on indexes of arrays, we introduce Presburger arithmetic $\mathcal{P}$ over indexes: any other decidable fragment of Arithmetic would be a good alternative. Thus the relevant operations on our arrays include *addition over indexes, read, write, and dimension*. The resulting theory (to be formally introduced later on) $\mathcal{ADP}$ can be seen as a combination of well-known theories such as $\mathcal{P}$ and the theory $\mathcal{A}_e$ of arrays with extensionality (see, e.g., [2]), extended with a function for dimension which takes an array and returns a natural number. Because of the function for dimension, the combination is *non-disjoint* and cannot be handled by classical combination schemas such as Nelson-Oppen [15]. Nevertheless, following [8], it is convenient to see $\mathcal{ADP}$ as a combination of $\mathcal{P}$ with a theory of array with dimension $\mathcal{A}_{dim}$: $\mathcal{A}_{dim}$ extends $\mathcal{A}_e$ (both in the signature and in the axioms), but is contained in $\mathcal{ADP}$, because indexes are only endowed with a discrete linear poset structure (the next subsection fixes the details). In this way, we have that $\mathcal{ADP} = \mathcal{A}_{dim} \cup \mathcal{P}$ and the theories $\mathcal{A}_{dim}$ and $\mathcal{P}$ share the well-known complete theory $\mathcal{T}_0$ of natural numbers endowed with zero and successor (see e.g., [5]): this theory admits quantifier elimination, so that the $\mathcal{T}_0$-compatibility hypothesis of [8] needed for the non-disjoint Nelson-Oppen combination is satisfied. Unfortunately, the combination result in [8] cannot be applied to $\mathcal{ADP}$ for mainly two reasons. First, $\mathcal{T}_0$ is not locally finite (see, e.g., [8] for details). Secondly, $\mathcal{A}_{dim}$ is a proper extension of the theory $\mathcal{A}_e$, hence the decision procedures for the $\mathcal{A}_e$-satisfiability problem (such as, e.g., the one in [2]) must be extended. In the rest of the paper, we will show that it is sufficient to use decision procedures for the $\mathcal{P}$- and $\mathcal{A}_e$-satisfiability problem to solve the $\mathcal{ADP}$-satisfiability problem provided that a suitable pre-processing of the input set of literals is performed.

Here, we formally introduce the basic theories of interests for this paper.

$\boxed{\mathcal{T}_0}$ has just one sort symbol INDEX, the following function and predicate symbols: $0 :$ INDEX, $\mathrm{s} :$ INDEX $\rightarrow$ INDEX, and $<:$ INDEX $\times$ INDEX. It is axiomatized

by the the following formulae:[1]

$$y \neq 0 \rightarrow \exists z(y = \mathrm{s}(z)) \tag{1}$$
$$x < \mathrm{s}(y) \leftrightarrow (x < y \vee x = y) \tag{2}$$
$$\neg(x < 0) \tag{3}$$
$$x < y \vee x = y \vee y < x \tag{4}$$
$$x < y \rightarrow \neg(y < x) \tag{5}$$
$$x < y \rightarrow (y < z \rightarrow x < z) \tag{6}$$

where $x$, $y$ and $z$ are variables of sort INDEX. This theory admits elimination of quantifiers and it is complete, see [5] for details.

$\boxed{\mathcal{P}}$ is the well-known Presburger arithmetic, see, e.g., [5], over indexes. The signature is that of $\mathcal{T}_0$ extended with the function symbol for addition $+$ : INDEX $\times$ INDEX $\rightarrow$ INDEX, written infix. Since $\mathcal{P}$ is not finitely axiomatizable (see, again [5]), we assume as axioms all valid sentences in the theory. Notice that $\mathcal{T}_0 \subset \mathcal{P}$.

$\boxed{\mathcal{A}}$ is the theory of arrays (see, e.g., [2]) which has the following signature:

- sort symbols: INDEX, ELEM, ARRAY and
- function symbols: select : ARRAY $\times$ INDEX $\rightarrow$ ELEM and store : ARRAY $\times$ INDEX $\times$ ELEM $\rightarrow$ ARRAY

and it is axiomatized by the following formulae:

$$\mathrm{select}(\mathrm{store}(a, i, e), i) = e \tag{7}$$
$$i \neq j \rightarrow \mathrm{select}(\mathrm{store}(a, i, e), j) = \mathrm{select}(a, j) \tag{8}$$

$\boxed{\mathcal{A}_e}$ is the theory of arrays with extensionality (see, e.g., [2]) which has the same signature of $\mathcal{A}$ and it is axiomatized by (7), (8), and the axiom of extensionality:

$$\forall i(\mathrm{select}(a, i) = \mathrm{select}(b, i)) \rightarrow a = b \tag{9}$$

Notice that $\mathcal{A} \subset \mathcal{A}_e$.

$\boxed{\mathcal{A}_{dim}}$ is the simple theory of arrays with dimension whose signature is the union of the signatures of $\mathcal{T}_0$ and $\mathcal{A}_e$ extended with the following three symbols: $\perp$ : ELEM, $\varepsilon$ : ARRAY, and dim : ARRAY $\rightarrow$ INDEX. It is axiomatized by the axioms in $\mathcal{T}_0$, those in $\mathcal{A}_e$, and the following formulae:

$$\dim(a) \leq i \rightarrow \mathrm{select}(a, i) = \perp \tag{10}$$
$$\dim(a) = \mathrm{s}(i) \rightarrow \mathrm{select}(a, i) \neq \perp \tag{11}$$
$$\dim(\varepsilon) = 0 \tag{12}$$

Notice that $\mathcal{T}_0 \subset \mathcal{A}_{dim}$ and $\mathcal{A}_e \subset \mathcal{A}_{dim}$.

$\boxed{\mathcal{ADP}}$ is the theory of arrays with dimension whose signature is the union of the signatures of $\mathcal{A}_{dim}$ and $\mathcal{P}$ and is axiomatized by the axioms in $\mathcal{A}_{dim}$ and all valid sentences in $\mathcal{P}$.

[1] Here and in the following, we omit the outermost universal quantification for the sake of readability.

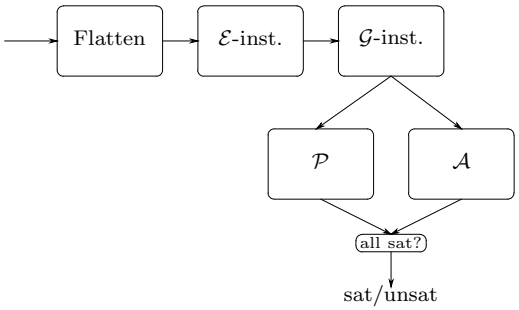


**Fig. 1.** The architecture of the decision procedure for $\mathcal{ADP}$

The constraint satisfiability problem for the theories $\mathcal{T}_0$, $\mathcal{P}$, $\mathcal{A}$, and $\mathcal{A}_e$ is decidable (see [5] for the first two and [2] for the last two). This is an important observation for the results of this paper, since the decision procedure for $\mathcal{ADP}$-satisfiability will assume the availability of two decision procedures for the constraint satisfiability problems of $\mathcal{P}$ and $\mathcal{A}$. The theories $\mathcal{A}_e$, $\mathcal{A}_{dim}$, and $\mathcal{ADP}$ admit a particular subclass of models, which we call the *standard* ones and are exactly those introduced above in order to motivate the definition of $\mathcal{ADP}$. Such models are characterized by the fact that the sort INDEX is always interpreted as the set $\mathbb{N}$ of natural numbers, and the sort ARRAY is interpreted as the set of all the sequences of elements from ELEM that are eventually equal to $\perp$; the dimension of each array is the successor of the index of the last element different from $\perp$. Of course, when investigating constraint satisfiability we are mainly interested in satisfiability of constraints in standard models and we shall in fact prove that a constraint is satisfiable in a model of $\mathcal{ADP}$ iff it is satisfiable in a standard model (see Lemma 4.3, below).

## 4 A Decision Procedure for Arrays with Dimension

We assume the availability of two decision procedures solving the $\mathcal{A}_e$- and $\mathcal{P}$-satisfiability problems. The overall schema of the procedure for $\mathcal{ADP}$-satisfiability problems is depicted in Figure 1. The idea is to reduce the $\mathcal{ADP}$-satisfiability problem to the constraint satisfiability problems for $\mathcal{A}_e$ and $\mathcal{P}$. The module Flatten pre-processes the literals in the input constraint so to make them flat and easily recognizable as belonging to one theory among those used to define $\mathcal{ADP}$, i.e. $\mathcal{T}_0$, $\mathcal{P}$, $\mathcal{A}$, or $\mathcal{A}_e$. The module $\mathcal{E}$-instantiation produces suitable instances of the extensionality axiom of arrays, i.e. (9), so that a simple satisfiability procedure for $\mathcal{A}$ is assumed available (rather than one for $\mathcal{A}_e$). The module $\mathcal{G}$-instantiation is non-deterministic and guesses sufficiently many instances of the axioms about dim, i.e. (10) and (11), as well as some facts entailed by the constraints in $\mathcal{P}$. The modules $\mathcal{P}$ and $\mathcal{A}$ implement the decision procedures for Presburger arithmetic and the theory of arrays without extensionality. The module 'all sat?' returns 'sat' if both decision procedures for $\mathcal{P}$ and $\mathcal{A}$ returned

'sat', and, otherwise, returns 'unsat'. We are now ready to describe the internal workings of each module.

### 4.1 Flattening

It is well-known (see, e.g., [2]) that it is possible to transform a constraint $\phi$ into an equisatisfiable constraint $\phi'$ containing only flat literals in linear time by introducing sufficiently many fresh constant symbols to name sub-terms. In our case, we assume that the module Flatten in Figure 1 transforms (in linear time) a set of arbitrary literals over the signature $\Sigma^{\underline{a}}_{\mathcal{ADP}}$, into an equisatisfiable set of flat literals on the signature $\Sigma^{\underline{c}}_{\mathcal{ADP}}$, for some set $\underline{c} \supseteq \underline{a}$ of constants (the constants in $\underline{c} \setminus \underline{a}$ are said to be fresh). Notice that a flattened set of literals $L$ can be represented as a set-theoretic union $L = L_{\mathcal{A}_{dim}} \cup L_{\mathcal{P}}$, where $L_{\mathcal{A}_{dim}}$ collects all the literals from $L$ whose signature is the signature of $\mathcal{A}_{dim}$ and $L_{\mathcal{P}}$ collects all the literals from $L$ whose signature is the signature of $\mathcal{P}$ (thus $L_{\mathcal{A}_{dim}} \cap L_{\mathcal{P}}$ contains precisely the literals from $L$ whose signature is the signature of $\mathcal{T}_0$).

### 4.2 $\mathcal{E}$-instantiation Closure

The $\mathcal{E}$-instantiation module in Figure 1 is based on the Skolemization of axiom (9).

**Definition 4.1 ($\mathcal{E}$-instantiation closed set of literals).** *A set $L$ of ground flat literals is $\mathcal{E}$-instantiation closed iff for every negative literal of the kind $a \neq b$ that belongs to $L$ (with $a, b :$ ARRAY), we have that $\{select(a, i) = e_1, select(b, i) = e_2, e_1 \neq e_2\} \subseteq L$, for some constants $i :$ INDEX, $e_1, e_2 :$ ELEM;*

The correctness of the module is stated below.

**Lemma 4.1.** *There exists a linear time algorithm which takes a set $L$ of flat literals over the signature $\Sigma^{\underline{a}}_{\mathcal{ADP}}$ and returns a $\mathcal{E}$-instantiation closed set $L^{\mathcal{E}}$ of flat literals over the signature $\Sigma^{\underline{c}}_{\mathcal{ADP}}$ such that (i) $L \subseteq L^{\mathcal{E}}$, (ii) $L$ and $L^{\mathcal{E}}$ are $\mathcal{ADP}$-equisatisfiable, and (iii) $\underline{a} \subseteq \underline{c}$.*

### 4.3 $\mathcal{G}$-instantiation Closure

The module $\mathcal{G}$-instantiation is non-deterministic and it is responsible to produce suitable instances of the axioms (10) and (11) as well as to guess (hence the name of $\mathcal{G}$-instantiation) enough facts of $\mathcal{P}$ entailed by the input constraint.

**Definition 4.2 ($\mathcal{G}$-instantiation closed set of literals).**
*A set $L$ of ground flat literals is $\mathcal{G}$-instantiation closed iff the following conditions are satisfied:*

1. *if $\varepsilon$ occurs in $L$, then $dim(\varepsilon) = 0 \in L$.*
2. *if $dim(a) = i \in L$, with $a :$ ARRAY and $i :$ INDEX, then $\{i = 0\} \subseteq L$ or $\{e \neq \bot, select(a, j) = e, s(j) = i\} \subseteq L$ for some constant $j :$ INDEX;*
3. *if $i, j$ occur in $L$, with $i, j :$ INDEX, then $i = j \in L$ or $i \neq j \in L$;*

```
T ⟵ {A, P}
function DP_ADP (L: set of flat literals)
    L^E ⟵ E-instantiation(L)
    for each L^G ⟵ G-instantiation(L^E) do begin
        for each T ∈ T do ρ_T ⟵ DP_T(L^G_T)
        if ⋀_{T∈T}(ρ_T = sat) then return sat
    end
    return unsat
end
```

**Fig. 2.** The (extensible) decision procedure for $\mathcal{ADP}$

4. *if $i, j$ occur in $L$, with $i, j$ : INDEX and $i \neq j \in L$, then $i < j \in L$ or $j < i \in L$;*
5. *if $\{dim(a) = i, i \leq j\} \subseteq L$, with $a$ : ARRAY and $i, j$ : INDEX, then $\{select(a, j) = \bot\} \subseteq L$ (here $i \leq j$ stands for $i < j$ or $i = j$).*

It is not difficult to see that, given a set of literals, it is always possible to compute its $\mathcal{G}$-instantiation in (non-deterministic) polynomial time.

**Lemma 4.2.** *There exists a non-deterministic polynomial time algorithm which takes as input a set $L$ of ground flat literals over a signature $\Sigma^{\underline{a}}_{\mathcal{ADP}}$ and returns a $\mathcal{G}$-instantiation closed set $L^{\mathcal{G}}$ of flat literals over the signature $\Sigma^{\underline{c}}_{\mathcal{ADP}}$ such that (i) $L \subseteq L^{\mathcal{G}}$, (ii) $L$ and $L^{\mathcal{G}}$ are $\mathcal{ADP}$-equisatisfiable, and (iii) $\underline{a} \subseteq \underline{c}$.*

For the correctness of our decision procedure, we need sets of literals that are both $\mathcal{E}$- and $\mathcal{G}$-instantiation closed. To this aim, one can check that the $\mathcal{E}$-instantiation module has to be invoked first, followed by the $\mathcal{G}$-instantiation module.

### 4.4 The Decision Procedure for $\mathcal{ADP}$

Figure 2 gives an algorithmic and non-deterministic description of the decision procedure to solve the $\mathcal{ADP}$-satisfiability problem. Without loss of generality (see Section 4.1), we assume that $L$ contains only flat literals. For a theory $T$ with decidable satisfiability problem, we write $DP_T$ for the decision procedure solving the $T$-satisfiability problem: $DP_T$ takes a set $L$ of $\Sigma_T$-literals and returns *sat* when $L$ is $T$-satisfiable; *unsat*, otherwise. If $L$ is a set of flat literals, then

$$L_T := \{\ell \mid \ell \in L \text{ is a } \Sigma_T\text{-literal}\},$$

where $T \in \{\mathcal{A}, \mathcal{P}\}$. So, for example, $L^{\mathcal{G}}_{\mathcal{P}}$ is the subset of the $\Sigma_{\mathcal{P}}$-literals in $L^{\mathcal{G}}$. The set $\mathsf{T}$ in Figure 2 contains the names of the theories for which a decision procedure is assumed available. It will be used for modularly extending the procedure in Section 5.

Let $L$ be a set of flat $\Sigma_{\mathcal{ADP}}$-literals to be checked for $\mathcal{ADP}$-satisfiability. The decision procedure $DP_{\mathcal{ADP}}$ first computes the $\mathcal{E}$-instantiation $L^{\mathcal{E}}$ of $L$ (recall

from Lemma 4.1 that this can be done in linear time). Then, it enumerates all possible $\mathcal{G}$-instantiations (cf. the **for each** loop in Figure 2). If it is capable of finding a $\mathcal{G}$-instantiation $L^{\mathcal{G}}$ such that its $\Sigma_{\mathcal{P}}$-literals are $\mathcal{P}$-satisfiable and its $\Sigma_{\mathcal{A}}$-literals are $\mathcal{A}$-satisfiable, then $DP_{\mathcal{ADP}}$ returns the $\mathcal{ADP}$-satisfiability of the input set $L$ of literals. Otherwise, if all possible $\mathcal{G}$-instantiations are enumerated and the test of the conditional in the body of the loop always fails, then $DP_{\mathcal{ADP}}$ returns the $\mathcal{ADP}$-unsatisfiability of the input set $L$ of literals.

### 4.5 Correctness of the Decision Procedure for $\mathcal{ADP}$

The termination of $DP_{\mathcal{ADP}}$ is immediate, whereas its soundness and completeness (Theorem 4.1 below) are consequences of the following Combination Lemma.

**Lemma 4.3 (Combination).** *Let $L$ be a $\mathcal{E}$- and $\mathcal{G}$-instantiation closed finite set of flat literals. Then, the following conditions are equivalent:*

(i) *$L$ is satisfiable in a standard model of $\mathcal{ADP}$;*
(ii) *$L$ is $\mathcal{ADP}$-satisfiable;*
(iii) *$L_{\mathcal{A}}$ is $\mathcal{A}$-satisfiable and $L_{\mathcal{P}}$ is $\mathcal{P}$-satisfiable.*

The soundness and correctness of $DP_{\mathcal{ADP}}$ is stated in the following

**Theorem 4.1.** *$DP_{\mathcal{ADP}}$ is a decision procedure for the $\mathcal{ADP}$-satisfiability problem, i.e. for any set $L$ of flat literals, $L$ is $\mathcal{ADP}$-satisfiable iff $DP_{\mathcal{ADP}}(L)$ returns sat. Furthermore, $DP_{\mathcal{ADP}}$ decides the satisfiability problem in the standard models of $\mathcal{ADP}$.*

## 5 Extensions of the Theory of Arrays with Dimension

We show the decidability of two interesting extensions of $\mathcal{ADP}$ (more extensions can be found in the Technical Report [9]).

### 5.1 Injective Arrays

The first extension of $\mathcal{ADP}$ is obtained by adding an axiom recognizing injective arrays which, according to [14], may characterize memory configurations where pointers satisfy the no-aliasing property. We extend the (empty) set of predicate symbols $\mathcal{ADP}$ by the unary predicate symbol $\mathsf{Inj}$ : ARRAY which holds for arrays containing no repeated elements, with the exception of the undefined element $\perp$ (the decidability of a similar problem is left open in [3]). To formalize the intended meaning of $\mathsf{Inj}$, we consider the theory $\mathcal{ADP}_{\mathsf{inj}}$ obtained by extending $\mathcal{ADP}$ with the following defining axiom:

$$\mathsf{Inj}(a) \leftrightarrow \forall i, j(\mathrm{select}(a, i) = \mathrm{select}(a, j) \rightarrow i = j \vee \mathrm{select}(a, i) = \perp) \quad (13)$$

where $a$ is a variable of sort ARRAY. In order to obtain a decision procedure for $\mathcal{ADP}_{\mathsf{inj}}$, it is necessary to find suitable extensions of Definitions 4.1 and 4.2 so

that enough instances of (13) are considered, and the results of the available decision procedures for $\mathcal{A}$ and $\mathcal{P}$ are conclusive about the satisfiability of the original constraint in the extended theory. We formalize the meaning of "enough instances" for $\mathcal{ADP}_{\mathsf{inj}}$ in the following two definitions.

**Definition 5.1 ($\mathcal{E}_{\mathsf{inj}}$-instantiation closed set of literals).** *A set $L$ of ground flat literals is $\mathcal{E}_{\mathsf{inj}}$-instantiation closed iff (i) $L$ is $\mathcal{E}$-instantiation closed (cf. Definition 4.1) and moreover for every negative literal $\neg\mathsf{Inj}(a) \in L$, there are constants $e : \text{ELEM}, i, j : \text{INDEX}$ such that $\{select(a,i) = e, select(a,j) = e, i < j, e \neq \bot\} \subseteq L$.*

**Definition 5.2 ($\mathcal{G}_{\mathsf{inj}}$-instantiation closed set of literals).** *A set $L$ of ground flat literals is $\mathcal{G}_{\mathsf{inj}}$-instantiation closed iff $L$ is $\mathcal{G}$-instantiation closed and the following conditions are satisfied:*

1. *if $\mathsf{Inj}(a) \in L$ then, for each constant $i$ of sort* INDEX *occurring in $L$, $select(a,i) = \bot \in L$ or $\{select(a,i) = e, e \neq \bot\} \subseteq L$ for some constant $e : \text{ELEM}$;*
2. *if $\{\mathsf{Inj}(a), i < j, select(a,i) = e_1, select(a,j) = e_2, e_1 \neq \bot, e_2 \neq \bot\} \subseteq L$, then $e_1 \neq e_2 \in L$.*

Lemmas 4.1 and 4.2 can easily be adapted to the theory $\mathcal{ADP}_{\mathsf{inj}}$. Since the combination Lemma 4.3 continues to hold with Definitions 5.1 and 5.2, we can show the correctness of the decision procedure $DP_{\mathcal{ADP}_{\mathsf{inj}}}$ for $\mathcal{ADP}_{\mathsf{inj}}$, which is obtained from $DP_{\mathcal{ADP}}$ by replacing the modules for $\mathcal{E}$- and $\mathcal{G}$-instantiation in Figure 1 with those taking into account Definitions 5.1 and 5.2.

**Theorem 5.1.** *$DP_{\mathcal{ADP}_{\mathsf{inj}}}$ is a decision procedure for the $\mathcal{ADP}_{\mathsf{inj}}$-satisfiability problem. Furthermore, $DP_{\mathcal{ADP}_{\mathsf{inj}}}$ decides the satisfiability problem in the standard models of $\mathcal{ADP}_{\mathsf{inj}}$.*

### 5.2 Arrays with Domain

The *second extension* of $\mathcal{ADP}$ we consider is again motivated by applications in program verification. As already observed in [17], it is quite helpful to regard arrays as functions equipped with an operator to compute their domains. This is used, for example, to define the semantics of separating connectives (supporting local reasoning) of Separation Logic [18]. So, we extend $\mathcal{ADP}$ with a set of axioms characterizing a function which, given an array $a$, returns the domain $dom(a)$ of $a$, i.e. $dom(a)$ is the set of indexes $i$ such that $\text{select}(a,i) \neq \bot$.

To formalize this extension of $\mathcal{A}_{dim}$, we need to introduce a very simple theory of sets of indexes, which is a straightforward extension of that used in [2]. Let $\mathcal{S}^{\emptyset}$ be the theory whose sort symbols are BOOL and SET, whose function symbols are $\text{true}, \text{false} : \text{BOOL}$, $\emptyset : \text{SET}$, $\text{mem} : \text{INDEX} \times \text{SET} \to \text{BOOL}$, $\text{ins} : \text{INDEX} \times \text{SET} \to \text{SET}$, and whose axioms are the following:

$$\text{mem}(i, \emptyset) = \text{false} \tag{14}$$

$$\text{mem}(i, \text{ins}(i, s)) = \text{true} \tag{15}$$

$$i_1 \neq i_2 \to \text{mem}(i_1, \text{ins}(i_2, s)) = \text{mem}(i_1, s) \tag{16}$$

$$\text{true} \neq \text{false} \wedge (\forall x : \text{BOOL}\ x = \text{true} \vee x = \text{false}) \tag{17}$$

where $i, i_1, i_2$ ($s$) are variables of sort INDEX (SET, respectively). Intuitively, $\emptyset$ denotes the empty set, mem is the test for membership of an index to a set, ins adds an index to a set if it is not already in the set. It is possible to adapt the decidability result of [2] to $\mathcal{S}^{\emptyset}$ (see [9] for details). Since we want to be able to compare sets by using the membership predicate mem, we need to consider the theory $\mathcal{S}^{\emptyset}_e$ obtained from $\mathcal{S}^{\emptyset}$ by adding the following axiom of extensionality for sets (here $s_1, s_2$ are variables of sort SET):

$$\forall i(\mathrm{mem}(i, s_1) = \mathrm{mem}(i, s_2)) \rightarrow s_1 = s_2. \tag{18}$$

Let $\mathcal{ADP}_{\mathsf{dom}}$ be the theory obtained by extending the (disjoint) union of $\mathcal{ADP}$ with $\mathcal{S}^{\emptyset}_e$ by the function symbol dom : ARRAY $\rightarrow$ SET together with the following axiom:

$$\mathrm{select}(a, i) = \bot \leftrightarrow \mathrm{mem}(i, \mathrm{dom}(a)) = \mathrm{false} \tag{19}$$

where $i$ and $a$ are variables of sort INDEX and ARRAY, respectively.

In order to obtain a decision procedure for $\mathcal{ADP}_{\mathsf{dom}}$, it is necessary to find suitable extensions of Definitions 4.1 and 4.2 so that enough instances of axioms (18) and (19) are considered and the results of the available decision procedures for $\mathcal{A}$, $\mathcal{P}$, and $\mathcal{S}^{\emptyset}$ are conclusive about the satisfiability of the original constraint in the extended theory. We formalize the meaning of "enough instances" for axiom (18) in the following definition.

**Definition 5.3 ($\mathcal{E}_{\mathsf{set}}$-instantiation closed set of literals).** *A set $L$ of ground flat literals is $\mathcal{E}_{\mathsf{set}}$-instantiation closed iff $L$ is $\mathcal{E}$-instantiation closed (cf. Definition 4.1) and for every literal of the kind $s_1 \neq s_2 \in L$ (with $s_1, s_2$ constants of sort* SET*), there are constants $b_1, b_2$ : BOOL, $i$ : INDEX *such that* $\{mem(i, s_1) = b_1, mem(i, s_2) = b_2, b_1 \neq b_2\} \subseteq L$.*

Instead of using guessing as for $\mathcal{ADP}_{\mathsf{inj}}$ in Section 5.2, we adopt the rewriting-approach to satisfiability procedures of [2]. We use the superposition calculus (from now on denoted by $\mathcal{SP}$) to build a rewriting-based decision procedure for the satisfiability problem in the union of the theories $\mathcal{A}_e$ and $\mathcal{S}^{\emptyset}_e$ extended with axiom (19). Such a procedure is then combined with a decision procedure for the satisfiability problem in $\mathcal{P}$ to build a decision procedure for $\mathcal{ADP}_{\mathsf{dom}}$.

In [2], it is shown how to use $\mathcal{SP}$ to build decision procedures for theories axiomatized by a finite set of first-order clauses. The key observation is that, in order to show that $\mathcal{SP}$ is a decision procedure, it is sufficient to prove that $\mathcal{SP}$ terminates on the set of clauses obtained by the union of the axioms of the theory and an arbitrary set of ground and flat literals. According to [2], $\mathcal{SP}$ terminates also for some of the theories considered in this paper, e.g., $\mathcal{A}$ and $\mathcal{S}^{\emptyset}_{-} := \mathcal{S}^{\emptyset} \setminus \{(17)\}$ (when considered in isolation). Modularity results in [1] allow us to conclude that $\mathcal{SP}$ also terminates for the union $\mathcal{A} \cup \mathcal{S}^{\emptyset}_{-}$. Unfortunately, this is not enough here since our goal is to build a decision procedure $\mathcal{ADP}_{\mathsf{dom}}$ whose set of axioms also contains (17) and (19).

Below, we develop the termination result for $\mathcal{SP}$ necessary to replace guessing as for $\mathcal{ADP}_{\mathsf{inj}}$ (cf. Section 5.1) with $\mathcal{SP}$. Notice that $\mathcal{SP}$ is used in two ways: to

check for unsatisfiability in the theory of equality and to find enough instances of the axioms of $\mathcal{A} \cup \mathcal{S}^{\emptyset}_{-}$ together with (17) and (19). A similar approach has also been investigated in [11] (for the theories already considered in [2]) to enable the efficient combination of rewriting-based satisfiability procedures with a decision procedure for $\mathcal{P}$.

Let $L$ be a set of ground and flat $\Sigma_{\mathcal{A} \cup \mathcal{S}^{\emptyset}}$-literals; we define $\mathcal{I}_L$ to be the following set of (partial) instances of axioms (17) and (19):

$$\begin{aligned} \mathrm{select}(a, x) \neq \bot \ \ \vee \ \ \mathrm{mem}(x, \mathrm{dom}(a)) \neq \mathrm{true}, \\ \mathrm{select}(a, x) = \bot \ \ \vee \ \ \mathrm{mem}(x, \mathrm{dom}(a)) = \mathrm{true}, \\ \mathrm{true} \neq \mathrm{false}, \text{ and } b = \mathrm{true} \vee b = \mathrm{false} \end{aligned}$$

for each $\mathrm{dom}(a) = s$ in $L$ and for each constant $b : \textsc{bool}$ occurring in $L$.

**Lemma 5.1.** *$\mathcal{SP}$ terminates on $\mathcal{A} \cup \mathcal{S}^{\emptyset}_{-} \cup \mathcal{I}_L \cup L$ for every set $L$ of $\Sigma_{\mathcal{A} \cup \mathcal{S}^{\emptyset}}$-literals.*

In the following, we denote with $DP_{\mathcal{SP}}$ the function taking an $\mathcal{E}_{\mathsf{set}}$-instantiation closed set $L$ of $\Sigma_{\mathcal{A} \cup \mathcal{S}^{\emptyset}}$-literals, computing $\mathcal{I}_L$, and then invoking $\mathcal{SP}$ on the clauses $\mathcal{A} \cup \mathcal{S}^{\emptyset}_{-} \cup \mathcal{I}_L \cup L$. If the empty clause is derived by $\mathcal{SP}$, then $DP_{\mathcal{SP}}$ returns *unsat*; *sat*, otherwise. The decision procedure $DP_{\mathcal{ADP}_{\mathsf{dom}}}$ for the theory $\mathcal{ADP}_{\mathsf{dom}}$ is obtained from $DP_{\mathcal{ADP}}$ by replacing the module for $\mathcal{E}$-instantiation in Figure 1 with a module for $\mathcal{E}_{\mathsf{set}}$-instantiation (cf. Definition 5.3) and by calling $DP_{\mathcal{SP}}$ instead of $DP_{\mathcal{A}}$ in the loop of Figure 2.

**Theorem 5.2.** *$DP_{\mathcal{ADP}_{\mathsf{dom}}}$ is a decision procedure for the $\mathcal{ADP}_{\mathsf{dom}}$-satisfiability problem.*

## 6 Conclusion

We have considered extensions of the theory of arrays which are relevant for many important applications such as program verification. These extensions are such that the indexes of arrays has the algebraic structure of Presburger Arithmetic and the theory of arrays is augmented with axioms characterizing additional symbols such as dimension, injectivity, or the domain of definition of arrays. We have obtained the decidability of all the considered extensions by a combination of decision procedures for the theories of arrays and Presburger Arithmetic with various instantiation strategies based both on model-theoretic and rewriting-based methods.

## References

1. A. Armando, M. P. Bonacina, S. Ranise, and S. Schulz. On a rewriting approach to satisfiability procedures: extension, combination of theories and an experimental appraisal. In *Proc. of 5th Int. Workshop on Frontiers of Combining Systems (FroCoS'05)*, volume 3717 of *LNCS*, 2005.
2. A. Armando, S. Ranise, and M. Rusinowitch. A rewriting approach to satisfiability procedures. *Information and Computation*, 183(2):140–164, 2003.

3. A. R. Bradley, Z. Manna, and H. B. Sipma. What's decidable about arrays? In *Proc. of 7th Int. Conf. on Verification, Model Checking, and Abstract Interpretation (VMCAI'06)*, volume 3855 of *LNCS*, 2006.
4. P. J. Downey and R. Sethi. Assignment commands with array references. *Journal of the ACM*, 25(4):652–666, 1978.
5. H. B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, New York-London, 1972.
6. J. H. Gallier. *Logic for Computer Science: Foundations of Automatic Theorem Proving*. Harper & Row, 1986.
7. H. Ganzinger and K. Korovin. Integrating equational reasoning in instantiation-based theorem proving. In *Proc. of Computer Science in Logic (CSL'04)*, volume 3210 of *LNCS*, 2004.
8. S. Ghilardi. Model-theoretic methods in combined constraint satisfiability. *Journal of Automated Reasoning*, 33(3-4):221–249, 2004.
9. S. Ghilardi, E. Nicolini, S. Ranise, and D. Zucchelli. Deciding extension of the theory of arrays by integrating decision procedures and instantiation strategies. Rapporto Interno DSI 309-06, Università degli Studi di Milano, Milano (Italy), 2006. Available at `http://homes.dsi.unimi.it/~zucchell/publications/techreport/GhiNiRaZu-RI309-06.pdf`.
10. J. Jaffar. Presburger arithmetic with array segments. *Information Processing Letters*, 12(2):79–82, 1981.
11. H. Kirchner, S. Ranise, C. Ringeissen, and D.-K. Tran. On superposition-based satisfiability procedures and their combination. In *Proc. of the 2nd Int. Conf. on Theoretical Aspects of Computing (ICTAC'05)*, volume 3722 of *LNCS*, 2005.
12. P. Mateti. A decision procedure for the correctness of a class of programs. *Journal of the ACM*, 28(2):215–232, 1981.
13. J. McCarthy. Towards a mathematical theory of computation. In *Proceedings of IFIP Congress*, 1962.
14. S. McPeak and G. Necula. Data structures specification via local equality axioms. In *Proc. of 17th Int. Conf. on Computer Aided Verification (CAV'05)*, volume 3576 of *LNCS*, 2005.
15. G. Nelson and D. C. Oppen. Simplification by cooperating decision procedures. *ACM Transaction on Programming Languages and Systems*, 1(2):245–257, 1979.
16. R. Nieuwenhuis and A. Rubio. Paramodulation-based theorem proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*. 2001.
17. J. C. Reynolds. Reasoning about arrays. *Communications of the ACM*, 22(5):290–299, 1979.
18. J. C. Reynolds. Separation logic: a logic for shared mutable data structures, 2002.
19. A. Stump, C. W. Barrett, D. L. Dill, and J. Levitt. A decision procedure for an extensional theory of arrays. In *Proc. of the 16th IEEE Symposium on Logic in Computer Science (LICS'01)*. IEEE Computer Society, 2001.
20. N. Suzuki and D. R. Jefferson. Verification decidability of Presburger array programs. *Journal of the ACM*, 27(1):191–205, 1980.