Electronic Notes in Theoretical Computer Science 125 (2005) 37–51 

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

Combining Non-stably Infinite, Non-first Order Theories [1](#_bookmark0)

# Pascal Fontaine[2](#_bookmark0) and Pascal Gribomont[3](#_bookmark0)

*University of Li`ege (Belgium)*

Abstract

A crucial step in the assertional verification of concurrent programs is deciding whether some sets of literals are satisfiable or not. In this context, the Nelson-Oppen combination scheme is often used. This scheme combines decision procedures for two disjoint theories into a decision procedure for the union of these theories. However, the standard version of the Nelson-Oppen technique tackles only one-sorted, stably infinite first-order theories. The scheme has previously been adapted to a many- sorted framework [[10](#_bookmark19)], and to handle non-stably infinite theories [[9](#_bookmark20)]. Those two enhancements were presented independently. We propose a unifying version in the continuity of both previous ones, which further relaxes the stably infinite requirement. Notably, some non-stably infinite theories can now be combined with the theory of arrays. Also, the combination scheme is presented here using a semantic notion of theory, allowing to handle non-first order theories.

*Keywords:* combination, decision procedure, non stably-infinite, non first-order

The Nelson-Oppen combination framework [[6](#_bookmark18)] aims at creating a decision procedure for the union of two theories on disjoint languages, from the decision procedures for the quantifier-free fragment in each theory. For instance the set of literals

*L* = {*x* ≤ *y, y* ≤ *x* + *f* (*x*)*, P* (*h*(*x*) − *h*(*y*))*,* ¬*P* (0)*, f* (*x*) = 0}

contains uninterpreted predicates and functions augmented with linear arith- metic on integer. The combination scheme provides a decision procedure for such sets of literals, from a decision procedure for uninterpreted predicates

1 This work was funded by a grant of the “Communaut´e franc¸aise de Belgique - Direction de la recherche scientifique - Actions de recherche concert´ees”

2 Email: [pfontain@montefiore.ulg.ac.be](mailto:pfontain@montefiore.ulg.ac.be)

3 Email: [gribomont@montefiore.ulg.ac.be](mailto:gribomont@montefiore.ulg.ac.be)

1571-0661 © 2005 Elsevier B.V. O pen access under [CC BY-NC-ND license.](http://creativecommons.org/licenses/by-nc-nd/3.0/)

doi:10.1016/j.entcs.2004.06.066

and functions (i.e., the empty first-order theory) and a decision procedure for linear arithmetic. This is possible because languages are disjoint, the only shared symbols being the equality and variables.

The Nelson-Oppen combination scheme is usually presented as a combina- tion scheme for *ﬁrst-order theories*. However, adopting a more semantic view allows to avoid the delicate problem of dealing with, for instance, “theories”

admitting only arbitrary large finite models [4](#_bookmark1) by precisely selecting the mod- els of the “theory” one wants to consider. An *I-theory* is an arbitrary set of interpretations on the considered language. For example, considering linear

arithmetic on integers, the I-theory will be the set of all interpretations corre- sponding to the single structure assigning setto the domain, and their usual meaning to symbols +, −, ≤, 0, 1. . . We present here a combination scheme for I-theories. As first-order theories can be expressed as I-theories, [5](#_bookmark2) this can be seen as a generalization of the usual first-order Nelson-Oppen combination scheme. A Shostak’s combination scheme with a similar notion of generalized theory has already been presented in [[4](#_bookmark16)].

In a verification context, it is natural to express verification conditions in a many-sorted framework [[3](#_bookmark15)]. Very recently, the Nelson-Oppen method has been reformulated and proved correct in an order-sorted framework [[10](#_bookmark19)]. To simplify the presentation, we will rather work in a basic many-sorted framework, with disjoint sorts (i.e., without subsorts).

Traditionally it is required for the theories in a combination to be *stably inﬁnite*. This means using the combination scheme is problematic with some

common theories (notably those having only finite models). In the philosophy of [[9](#_bookmark20)], it will be shown that — in a many-sorted framework — non-stably infinite theories combine easily with some very useful theories (notably the empty theory, and the theories of arrays).

# Preliminaries

A *many-sorted ﬁrst-order language* is a tuple L = ⟨S*,* V*,* F *,* P*, r, d*⟩ such that S is a countable non-empty set of sorts (or types), V is the (countable) union of disjoint countable sets V*τ* of variables of sort *τ* , F is a countably infinite set of function symbols, P is a countably infinite set of predicate symbols, *r* assigns an arity to each function symbol in F and each predicate symbol in P, and *d* assigns a sort in S*r*(*f*)+1 to each function symbol *f* ∈F and a sort in S*r*(*p*) to each predicate symbol *p* ∈ P. Nullary predicates are propositions, and

4 A first-order theory admitting arbitrary large finite models also has infinite models.

5 The I-theory corresponding to a first-order theory is just the set of the models of the first-order theory.

nullary functions are constants. A subset of L is a many-sorted language such that the sets of sorts, variables, function and predicate symbols are subsets of the corresponding sets of L.

It is assumed symbols are not overloaded. That is, a given function or predicate cannot be assigned to different sorts. It is however handy to use the same symbol with arguments of different sorts. In that case it suffices to consider that such symbols are implicitly “decorated” with their sort decla- ration. One example is the equality symbol “=”, which is decorated into one symbol for every sort. Terms, atoms (atomic formulas), literals, and formulas of a many-sorted language are defined in the usual way.

An *interpretation* of a formula in a many-sorted first-order language L is a pair I = ⟨*D, I*⟩ where *D* assigns a non-empty domain *Dτ* to each sort *τ* ∈S

and *I* assigns a meaning to each variable, function, and predicate symbol. As usual, the identity is assigned to the equality symbol. An interpretation I assigns a value I[*t*] in *Dτ* to every term *t* of sort *τ* . Similarly, interpretation I assigns a value I[*ϕ*] in {T*,* ⊥} to every formula *ϕ*. An interpretation I is

a model for formula *ϕ* if I[*ϕ*] = T. It is noted I H *ϕ*. A *restriction* of an

interpretation I in language L to a subset of L is the interpretation equal to

I for every domain, function symbol, predicate symbol in the subset of L.

A first-order theory is a set of axioms, which defines a set of interpretations, i.e., the models of the first-order theory. A natural extension is to consider I-theories: an I-theory is an arbitrary set of interpretations in a given many- sorted language. The I-theory corresponding to a first-order theory is the set of the models of the first-order theory. An I-theory may leave some predicates

and functions uninterpreted. A predicate *p* of sort ⟨*τ*1*,... τn*⟩ (a function *f* of sort ⟨*τ*1*,... τn,τ* ⟩) is uninterpreted in an I-theory T if for every interpretation I = ⟨*D, I*⟩ ∈ T , and for every predicate *q* (resp. function *g*) of suitable

sort there is an interpretation I' ∈ T such that I' is the same as I except that I'[*p*] = *q* (resp. I'[*f* ] = *g*). It is assumed that variables are always left uninterpreted in any I-theory, with a meaning similar to uninterpreted constants. Given an I-theory T , a formula *ϕ* is T -satisfiable if it has a model in T . Two I-theories are disjoint if their languages are disjoint. Disjoint languages are languages with disjoint sets of functions (and constants) and predicates (though the equality symbol is shared).

Formulas including predicates and functions of two disjoint I-theories are written in the union of the disjoint languages. The union of the disjoint languages L1 = ⟨S1*,* V1*,* F1*,* P1*, r*1*, d*1⟩ and L2 = ⟨S2*,* V2*,* F2*,* P2*, r*2*, d*2⟩ is the many-sorted language L = ⟨S1 ∪ S2*,* V1 ∪ V2*,* F1 ∪ F2*,* P1 ∪ P2*, r, d*⟩, where *r*

is the function equal to *r*1 when its argument is in the domain of *r*1, otherwise it is equal to *r*2. Function *d* is defined similarly.

An I-theory T in the union L of disjoint languages L1 and L2 can be seen as the union of two disjoint I-theories: the restrictions T*i* of T to each language L*i*, i.e., the set of all restrictions to L*i* of interpretations in T . However, it is more convenient to define the union of I-theories as:

Definition 1.1 The union T of I-theories T1 and T2 in disjoint languages

L1 and L2 is the set of all interpretations I such that interpretations I*i* =

⟨*Di, Ii*⟩∈ T*i* exist for *i* = 1*,* 2, with I*i* being the restriction to L*i* of I.

In this definition the restriction to L1 of T is not necessarily T1. Indeed, if for a common sort *τ* in L1 and L2, an interpretation I1 from T1 assigns to *τ* a domain which is never assigned by any interpretation in T2, then no interpretation in T will have I1 as a restriction.

Notice that if T1 and T2 are the set of models of first-order theories *T*1 and *T*2, the union of I-theories T1 and T2 is the set of models of the (first-order) theory corresponding to the set of axioms *T*1 ∪ *T*2. Our definitions for I-theory and union of I-theories extend the classical definitions of theory and union of theories but are not in contradiction with them.

As the combining method we propose applies to the general notion of I-theories introduced earlier, it is necessary to encapsulate the “combinable property” in a new notion. Combining two decision procedures for disjoint I- theories will require that, for each common sort, at least one of the I-theories

is *flexible*:

Definition 1.2 An I-theory T in a many-sorted language L is *flexible* on sort *τ* if *τ* is not a sort of L, or if for any interpretation I = ⟨*D, I*⟩ in T , any interpretation I' = ⟨*D*'*,I*'⟩ such that

* the sets *Dτ* and *D*' have the same cardinality, and *Dτ*' = *D* ' for any *τ* /= *τ* .

' '

*τ* ' *τ*

Function *b* defines a bijection from set *Dτ* to *Dτ* , and is the identity on *Dτ*'

for any *τ* ' /= *τ* ;

* for any variable *x*, I'[*x*] = *b*(I[*x*]);
* for any function symbol *f* ∈ F, I'[*f* ](*b*(*d* )*,... b*(*d* )) = *b* (I[*f* ](*d ,... d* ));

1 *n* 1 *n*

* for any predicate symbol *p* ∈ P, I'[*p*](*b*(*d* )*,... b*(*d* )) ≡ I[*p*](*d ,... d* ); also belongs to T .

1 *n* 1 *n*

Intuitively it means terms are assigned elements in the domain modulo a permutation. It allows to extend the use of the classic combination scheme to I-theories which do not necessarily correspond to first-order theories. As only one I-theory is needed to be flexible, the requirement is directly satisfied if one of the two I-theories corresponds to a first-order theory:

Theorem 1.3 *If I-theory* 7 *is the set of models of a ﬁrst-order theory in a given language* L*, then* 7 *is flexible on every sort.*

When combining more than two I-theories, it may seem the “flexible require- ment” will impose all but one I-theory in the combination to correspond to first-order theories. Fortunately that is not the case. The following theorem shows that no restriction occurs if any two individual non-flexible I-theories have different sorts:

Theorem 1.4 *Let* 7 *be the union of the disjoint I-theories* 71 *and* 72*. If* 71

*and* 72 *are flexible on τ, then* 7 *is flexible on τ.*

The set *L* of literals in the little example at the beginning of this chapter is unsatisfiable, and so is *L*1 ∪ *L*2, with

*L*1 = {*x* ≤ *y, y* ≤ *x* + *v*1*, v*1 = 0*, v*2 = *v*3 — *v*4*, v*5 = 0}

*L*2 = {*P* (*v*2) *,* ч*P* (*v*5) *, v*1 = *f* (*x*) *, v*3 = *h* (*x*) *, v*4 = *h* (*y*) }*.*

But every literal in *Li* (*i* = 1*,* 2) contains functions (and constants) and pred- icates from only one I-theory. Every literal in *L*1 is in the language of arith- metic, whereas every literal in *L*2 is in the language of uninterpreted predicates and functions. The shared symbols are variables only (and equality). The sets *L*1 and *L*2 are built from *L* simply by introducing new variables. A pair of sets of literals (*L*1*, L*2) is a separation of the finite set of literals *L* in the union of disjoint languages L1 and L2, if: first, *L* is 7 -satisfiable if and only if *L*1 ∪ *L*2 is 7 -satisfiable; second, *Li* is a finite set of literals in language L*i*, for *i* = 1*,* 2. The only shared symbols are variable symbols (and equality). In fact, given two languages and a set of literals in the union of those languages, a separation can always be built, using variable abstraction.

Given a partition C of the finite set of variables {*x*1*,... xn*}, the *arrange- ment* induced by C is the set of all equalities between variables of the same sort in the same class of C, and of all inequalities between two variables of the

same sort in two distinct classes of C. For instance, the arrangement of the three variables of the same sort *x*1*, x*2*, x*3 induced by partition {{*x*1*, x*2}*,* {*x*3}} is {*x*1 = *x*2*, x*1 /= *x*3*, x*2 /= *x*3}. In fact the last inequality is not essential: the set of literals {*x*1 = *x*2*, x*1 /= *x*3} is logically equivalent to the preceding ar- rangement.

# Cooperation of decision procedures

The following theorem provides the abstract basis for the combination proce- dure. With further requirements on the I-theories, it will allow to combine two decision procedures for disjoint languages into one for the union of languages:

Theorem 2.1 *Let I-theory* 7 *in language* L *be the union of I-theories* 71 *and* 72 *in disjoint languages* L1 *and* L2*. For each sort, either* 71 *or* 72 *is flexible. Let L be a set of literals on* L*, and let* (*L*1*, L*2) *be a separation of L according to* L1 *and* L2*. Let V be the set of common variables in L*1 *and L*2*.*

*Assume* [6](#_bookmark6) *that for every domain τ in* L*i and every term t of sort τ used in*

*L*2−*i there is an equality x* = *t* ∈ *L*2−*i where x is a variable in V* '*.*

*The set of literals L is* 7 *-satisﬁable if and only if there exists an arrange- ment* A *of V* ∪ *V* ' *and* 7*i-models* I*i of* A∪ *Li for i* = 1*,* 2 *such that, for any common sort τ in* L1 *and* L2*, the domains associated to τ in* I1 *and* I2 *have the same cardinality. In that case* A∪ *L*1 ∪ *L*2 *is* 7 *-satisﬁable.*

Proof. The proof can be found in appendix [A](#_bookmark24).

The classical way of solving the cardinality constraints like those in The- orem [6](#_bookmark4) is to consider only *stably inﬁnite* theories. In a many-sorted context, it means both theories should be stably infinite with respect to the set of

common sorts.

Definition 2.2 An I-theory 7 in a many-sorted language L is *stably inﬁnite with respect to a set of sorts S* if every 7 -satisfiable ground set of literals *L* on L has a 7 -model assigning a domain of cardinality ℵ0 for every sort *τ* ∈ *S* used in L.

Many useful I-theories are stably infinite: the I-theory of linear arithmetic on integer, I-theories corresponding to the first-order theories of arrays [[8](#_bookmark21)] or lists [[7](#_bookmark22)], etc. Also observe that, when talking about I-theories which are sets of models of first-order theories, the L¨owenheim-Skolem Theorem [7](#_bookmark7) allows to relax the condition on domain cardinality in this definition: the cardinalities should be greater or equal to ℵ0. But this remains a very strong requirement. Notably it prevents to combine I-theories with only finite domains.

We will rather adopt a pragmatic view, and study two I-theories in par- ticular. *Compatible I-theories* are I-theories requiring no “cardinality clause” in Theorem [6](#_bookmark4):

Definition 2.3 Two disjoint I-theories 71 and 72 in many-sorted languages L1 and L2 are *compatible* if for every separation (*L*1*, L*2) and arrangement A such that

* *V* is the set of common symbols in *L*1 and *L*2;

6 Every separation can be transformed into a separation verifying this condition.

7 A formulation of L¨owenheim-Skolem Theorem for many-sorted first-order logic may be found in [[10](#_bookmark19)].

* for every sort *τ* in L*i* and every term *t* of sort *τ* used in *L*2−*i* there is an equality *x* = *t* ∈ *L*2−*i* where *x* is a variable in *V* ';
* A is an arrangement of *V* ∪ *V* ';
* A∪ *Li* is 7*i*-satisfiable for *i* = 1*,* 2,

then there are 7*i*-models I*i* of A∪ *Li* for *i* = 1*,* 2 such that I1 and I2 assign sets with the same cardinality for every common sort.

If two I-theories are stably infinite with respect to the set of common sorts then they are compatible. But this is not a necessary condition: many I-theories are not stably infinite, but are compatible with some other I-theories.

Assume that 71 and 72 are compatible disjoint I-theories such that, for each sort, at least one of them is flexible. Then there is a satisfiability deci- sion procedure for sets of literals in the language of their union if there are satisfiability decision procedures for sets of literals in the language of each I- theory. Indeed, *L*1 ∪ *L*2 is unsatisfiable in the union if, for every arrangement A, A∪ *Li* is unsatisfiable for *i* = 1 or *i* = 2. In theory, the decision procedure can be, first, to compute the separation, and second, to check every arrange- ment of the suitable set of variables together with each part of the separation. The original set is unsatisfiable, if and only if no arrangement has been found to be satisfiable with both parts of the separation.

Given a set of variables *V* , there are as many arrangements as partitions for *V* (at least in a one-sorted logic). The number of partitions, also known as Bell numbers (see for example [[5](#_bookmark17)]), grows exponentially with respect to the size of |*V* |. In practice, checking every arrangement is not feasible, even for a small number of variables. [8](#_bookmark8) A more practical way for decision procedures to cooperate is by exchanging (disjunctions of) equalities. [9](#_bookmark9)

In Theorem [6](#_bookmark4), the set of variables on which the arrangement is built is larger than the set of *shared* variables, usually used in the Nelson-Oppen method. This is necessary for the combination with non-stably infinite I- theories, for instance, I-theories having only finite domains. A similar solution

was also used in [[11](#_bookmark23)], to combine the theory of sets with a theory for the elements, even if this last theory is not stably infinite.

Combining decision procedures for a finite number of I-theories not only requires those I-theories to have a decision procedure. Some I-theories in the combination should be flexible on some sorts in such a way that for any two I-theories in the combination, at least one is flexible for each (common) sort. This requirement is not very restricting, as first-order I-theories are

8 There are more than four million arrangements of 12 variables.

9 The justification is similar to the one presented in [[2](#_bookmark14)].

flexible on every sort (Theorem [1.3](#_bookmark3)). The requirement on domain cardinality is more restricting: the I-theories that are to be combined together should be compatible. Unfortunately, there is no simple and practical criteria (other than stable infiniteness) to find if I-theories are compatible. However, some common I-theories have nice properties regarding compatibility.

# The empty theory

The empty first-order theory in a language L defines the I-theory in which every constant, predicate, and function is uninterpreted, and no constraints are put on the domains. This I-theory will be referred to as the uninterpreted I-theory. It is certainly the simplest one, but it is nonetheless important as it allows uninterpreted predicates and functions to be added to the language of a decidable I-theory.

The uninterpreted I-theory has an optimal behaviour in a combination framework. Classically, only its stable infiniteness is exploited as all theories considered are assumed to be stably infinite. However, it was also noticed in

[[9](#_bookmark20)] that it has the required properties to be combined not only with stably infinite theories but also with every first-order theory. Here, we show that it can be combined with *any* I-theory, corresponding to a first-order theory or not. The deep reason for this is encapsulated in this lemma:

Lemma 3.1 *Let* 7 *be the uninterpreted I-theory in a many-sorted language* L*, and L be a set of literals on* L*. If* I = ⟨*D, I*⟩ ∈ 7 *is a model of L, then there is a model* I' = ⟨*D*'*,I*'⟩ ∈ 7 *of L such that, for any sort τ and any*

' ' '

*cardinality κ* ≥ |*Dτ* |*,* |*Dτ* | = *κ and* |*Dτ* ' | = |*Dτ*' | *for every τ*

/= *τ.*

A corollary of preceding Lemma is the stable infiniteness of the uninter- preted I-theory:

Theorem 3.2 *The uninterpreted I-theory in a language* L *is stably inﬁnite on every set of sorts of* L*.*

Proof. Assume *L* is a satisfiable finite set of literals in L. There is an inter- pretation I = ⟨*D, I*⟩ of *L* such that *Dτ* is finite for every sort *τ* of L. Using Lemma [3.1](#_bookmark10) on each sort successively, it is possible to build from I a model I'

assigning to every sort a domain of cardinality ℵ0.

As a consequence, the uninterpreted I-theory is compatible with every stably infinite I-theory. But a much stronger result can be deduced from Lemma [3.1](#_bookmark10):

Theorem 3.3 *The uninterpreted I-theory in a language* L *is compatible with every I-theory.*

Proof. Assume 71 is the uninterpreted I-theory in language L1 and 72 is any I-theory in language L2 disjoint from L1. Let (*L*1*, L*2) and A be a separation and an arrangement verifying the conditions of Definition [2.3](#_bookmark5).

For every common sort *τ* of L1 and L2, the arrangement A defines a partition of variables of sort *τ* : two variables of sort *τ* are in the same class if and only if they are assigned the same value by a model of A. This partition contains a given number *kτ* of classes. The models I*i* of A∪ *Li* attribute to each common sort *τ* a domain *Di,τ* such that |*Di,τ* | ≥ *kτ* . Furthermore, if A∪ *L*1 is satisfiable, then it has a model I1 which attributes to each common sort *τ* a domain *D*1*,τ* such that |*D*1*,τ* | = *kτ* .

Using Lemma [3.1](#_bookmark10) on each common sort successively, it is possible to build

'

from I1 a model I assigning to every sort a domain of the same cardinality

1

as interpretation I2.

Given any decidable I-theory 7 in language L, adding any finite number of uninterpreted functions, predicates, or constants keeps the I-theory decidable, for sets of literals in the resulting language.

# The theory of arrays

The first-order theory of arrays is one of the most popular theories used in combination schemes. A survey of results and history about it can be found in [[5](#_bookmark17),[8](#_bookmark21)]. The satisfiability problem for sets of literals in the language of a theory of arrays is decidable. In [[8](#_bookmark21)] the occurrences of “write” are eliminated and every atom read(*a,* ·) is translated to an application of an uninterpreted function associated to *a*, and finally congruence closure is used. In [[1](#_bookmark13)] it is proven that the superposition calculus is a satisfiability procedure for the (one-sorted) theory of arrays.

The theory contains “read-over-write” axioms:

∀*a*∀*i*∀*e* [read (write (*a, i, e*) *, i*) = *e*]

∀*a*∀*i*∀*j*∀*e* [*i* /= *j* ⇒ read (write (*a, i, e*) *, j*) = read (*a, j*)] and an extensionality axiom:

∀*a*∀*b* [∀*i* read (*a, i*) = read (*b, i*) ⇒ *a* = *b*] *.*

There are obviously three possible sorts: indices (*i*, *j*), arrays (*a*, *b*), and elements (*e*). But some sorts may be merged. It is natural and conservative to consider that the sort of arrays is disjoint from the sorts of elements and indices. [10](#_bookmark11) However, in numerous practical cases, it may be interesting to

10 Nevertheless elements in a theory of arrays can be arrays. But their language should be disjoint. For instance, read1, write1 and read2, write2.

merge sorts for indices and elements. Two theories should be thus considered: the two-sorted theory, and the three-sorted theory.

The I-theory corresponding to the first-order theory of arrays is not com- patible with every I-theory like the uninterpreted I-theory is. However, very positive results hold about the compatibility with the I-theory of arrays, thanks to the following lemma:

Lemma 4.1 *Let* 7 *be a two-sorted or three-sorted I-theory of arrays in lan- guage* L *and L be a set of literals on* L*. If* I = ⟨*D, I*⟩ ∈ 7 *is a model of L, then there is a model* I' = ⟨*D*'*,I*'⟩ ∈ 7 *of L such that, for sort τ of values*

*(or indices) and any cardinality κ* ≥ |*D* |*,* |*D*' | = *κ. If* 7 *is a three-sorted*

*τ τ*

' '

*I-theory and τ*

*is the sort of indices (resp. values), then* |*Dτ* ' | = |*Dτ*' |*.*

Proof. The proof can be found in appendix [B](#_bookmark25).

The stable-infiniteness is a direct consequence of preceding lemma:

Theorem 4.2 *The two-sorted (three-sorted) I-theory of arrays in language* L

*is stably inﬁnite on every set of sorts.*

It is often required to use the I-theory of arrays with a finite domain for elements or indices. Its stable-infiniteness is of no help in those cases. To guarantee that it is still possible to combine the I-theory of arrays with non- stably infinite theories, it is further required that no inequality is used on the array sort. This can be achieved by replacing, in the separation part in the language of arrays, every inequality *t*1 /= *t*2 where *t*1 and *t*2 are array terms

by read(*t*1*, i*) /= read(*t*2*, i*) where *i* is a new index variable. As a consequence,

new variables may have to be introduced in the set *V* ' to verify conditions of Theorem [6](#_bookmark4).

Theorem 4.3 *The two-sorted and three-sorted I-theories of arrays are com- patible with any I-theory in a language* L*, if the sort of arrays is not a sort of* L*, as long as no inequality is used on the array sort.*

Proof. Assume 71 is the two-sorted I-theory of arrays in language L1 and 72 is any I-theory in language L2 disjoint from L1. Let (*L*1*, L*2) and A be a separation and an arrangement verifying the conditions of Definition [2.3](#_bookmark5).

For every common sort *τ* of L1 and L2, the arrangement A defines a partition of variables of sort *τ* : two variables of sort *τ* are in the same class if and only if they are assigned the same value by a model of A. This partition contains a given number *kτ* of classes. The models I*i* of A∪ *Li* attribute to each common sort *τ* a domain *Di,τ* such that |*Di,τ* | ≥ *kτ* . As opposed to the empty theory, it is not direct to guarantee that, if A∪ *L*1 is satisfiable, then it

has a model I1 (in the array I-theory) which attributes to each common sort

*τ* a domain *D*1*,τ* such that |*D*1*,τ* | = *kτ* .

Consider the two-sorted theory. Every term of the common sort *τ* used in *L*1 is equal to a term used in A. Assume *d* ∈ *D*1*,τ* is associated to no

term used in A. Let I' be the same as I except that its domain for sort *τ* ,

1

1

'

*D* is *D*1*,τ* \ {*d*}, and I' [read(*a, i*)] = I [read(*a, i*)] if I [read(*a, i*)] /= *d*, or *d*'

1*,τ*

1 1 1

otherwise, where *d*' is an element of *D*'

chosen once and for all. Interpretation

' 1*,τ*

I1 is still a model of every literal in *L*1, and of both “read-over-write” axioms,

as those axioms are only universally quantified. If formula ∀*i* read (*a, i*) =

read (*b, i*) ⇒ *a* = *b* is made false by I'

for elements *a* and *b* of the array

' 1

domain in I1, then read (*a, i*) = read (*b, i*) for every *i*. It is possible to build

'' from I' such that only one representative of all such elements is kept.

I

1

1

''

Interpretation I1 still makes true both “read-over-write” axioms, and also the

extensionality axiom. Assuming no inequality is used on the array sort in *L*1,

I'' is also a model of every literal in *L* .

1 1

The same argument can be applied in the three-sorted case, for both ele-

ments and indices. Thus, if A∪ *L*1 is 71-satisfiable, it has a model I1 ∈ 71 which attributes to each common sort *τ* a domain *D*1*,τ* such that |*D*1*,τ* | = *kτ* . Using Lemma [4.1](#_bookmark12) it is thus possible to build a model I1 assigning to every common sort a domain of the same cardinality as interpretation I2.

Notice that previous theorem does not mean the I-theory of arrays is not compatible if the array sort is a common sort. It only states that if the array sort is a common sort, then the other I-theory must have a nice property to guarantee they are compatible. For instance, the array sort can be the sort of elements for another array I-theory. Also, the I-theory of arrays is compatible with the uninterpreted I-theory, even if both languages contain the sort of arrays.

# Conclusion

This new presentation of the Nelson-Oppen combination scheme enhances the classical scheme in two aspects. First, it handles I-theories, i.e., arbitrary sets of interpretations, not necessarily corresponding to first-order theories. This improvement allows to deal directly, for instance, with the linear arithmetic I- theories for,, with the arithmetic I-theory on, or with I-theories having only finite domains of unbounded size, used, for example in proof obligations issued in the context of verification of parameterized systems. Any quantifier- free decidable first-order language with interpreted functions and predicates corresponds to an I-theory which perfectly describes it.

An I-theory having only finite domains of unbounded size is essentially non-

stably infinite. The second improvement in this presentation is the combina- tion of non-stably infinite I-theories, following [[9](#_bookmark20)]. The many-sorted framework allows to combine in practice non-stably infinite I-theories with the I-theory corresponding to the empty theory, and with the (I-)theory of arrays.

As for the classical Nelson-Oppen framework, this framework for combining two I-theories can be applied to any number of I-theories, as long as they can be added one by one in the combination. A combination provides a decision procedure for the union of two I-theories, which is itself an I-theory. For instance, to deal with arrays containing lists of elements inwith some linear arithmetic on the sort of elements, the I-theory of lists can be combined with the I-theory for linear arithmetic on, thanks to the stable-infiniteness of both I-theories, and the obtained I-theory can be combined with the I-theory of arrays, using the good behaviour of the I-theory of arrays for combinations. Also, a three-sorted I-theory of arrays with functions read1 and write1 may have elements being arrays from another disjoint two-sorted I-theory of arrays with functions read2 and write2, if the only common sort is the sort of elements of for the first I-theory, i.e., the sort of arrays of the second one.

A limitation of this combination scheme is connected to the strict many- sorted framework. For instance, it does not allow to consider lists of elements which are either naturals *or* lists of naturals. Also, it remains to identify all cases where *V* ' can be safely ignored in Theorem [6](#_bookmark4). Identifying more theories

with nice compatibility properties is another issue for future research.

We would like to thank the anonymous reviewers for their comments.

# References

1. A. Armando, S. Ranise, and M. Rusinowitch. A rewriting approach to satisfiability procedures.

*Information and Computation*, 183(2):140–164, 2003.

1. S. Conchon and S. Krsti´c. Strategies for combining decision procedures. In P. Narendran and M. Rusinowitch, editors, *Proceedings of the 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (Warsaw, Poland)*, volume 2619 of *Lecture Notes in Computer Science*, pages 537–553. Springer-Verlag, Apr. 2003.
2. P. Fontaine and E. P. Gribomont. Decidability of invariant validation for paramaterized systems. In *Proc. Tools and Algorithms for Construction and Analysis of Systems*, volume 2619 of *Lecture Notes in Computer Science*, pages 97–112. Springer-Verlag, 2003.
3. H. Ganzinger. Shostak light. In A. Voronkov, editor, *Automated Deduction – CADE-18*, volume 2392 of *Lecture Notes in Computer Science*, pages 332–346. Springer-Verlag, July 27- 30 2002.
4. Z. Manna and C. G. Zarba. Combining decision procedures. In *Formal Methods at the Cross Roads: From Panacea to Foundational Support*, volume 2757 of *Lecture Notes in Computer Science*, pages 381–422. Springer, 2003.
5. G. Nelson and D. C. Oppen. Simplifications by cooperating decision procedures. *ACM Transactions on Programming Languages and Systems*, 1(2):245–257, Oct. 1979.
6. D. C. Oppen. Reasoning about recursively defined data structures. *Journal of the ACM*, 27(3):403–411, July 1980.
7. A. Stump, C. W. Barrett, D. L. Dill, and J. R. Levitt. A decision procedure for an extensional theory of arrays. In *16th Annual IEEE Symposium on Logic in Computer Science (LICS ’01)*, pages 29–37, Washington - Brussels - Tokyo, June 2001. IEEE.
8. C. Tinelli and C. G. Zarba. Combining non-stably infinite theories. In I. Dahn and L. Vigneron, editors, *Proceedings of the 4th International Workshop on First Order Theorem Proving, FTP’03 (Valencia, Spain)*, volume 86.1 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 2003.
9. C. Tinelli and C. G. Zarba. Combining decision procedures for theories in sorted logics. Technical Report 04-01, Department of Computer Science, The University of Iowa, Feb. 2004.
10. C. G. Zarba. Combining sets with elements. In N. Dershowitz, editor, *Veriﬁcation: Theory and Practice*, volume 2772 of *Lecture Notes in Computer Science*, pages 762–782. Springer, 2004.

# Proof of Theorem [6](#_bookmark4)

The condition is necessary. Let I ∈ 7 be a model of *L*. By definition of

separation, there is a model I' ∈7 of *L* ∪ *L* . This interpretation I' perfectly

1

2

defines an arrangement A of *V* ∪ *V* ', and I' is a model of A∪ *L*

1

∪ *L*2, and so

of A∪ *Li*.

Assume now interpretation I*i* ∈ 7*i* makes true A∪ *Li* for *i* = 1*,* 2. Inter- pretation I*i* defines a partition C*i* of all terms used in A∪ *Li* such that two terms *a* and *b* are in the same class if and only if I*i* H *a* = *b*. Consider *c*1 ∈ C1

'

and *c*2 ∈ C2, then *c*1 ∩ *c*2 ⊂ *V* ∪ *V* . But A perfectly defines which elements

of *V* ∪ *V* ' are in the same class. It is thus easy to make partitions C and C

1

2

coincide on classes containing a same element. Let C be a partition of terms in partitions C1 and C2 such that

* if *c*1 ∈ C1 and *c*2 ∈ C2 then *c*1 ∪ *c*2 ∈C if and only if *c*1 ∩ *c*2 is non empty;
* for every *ci* ∈ C*i* there exists *c* ∈ C such that *ci* ⊂ *c*. In other words, every class of C*i* is fully contained in one class of C;
* if *ci* /= *c*' ∈ C*i*, and if *c, c*' ∈C are such that *ci* ⊂ *c* and *c*' ⊂ *c*' then *c* /= *c*'.

*i* *i*

Partition C will allow to build an interpretation I from I1 and I2 such that

I H A∪ *L*1 ∪ *L*2.

First, let’s define the domains for I. For every sort *τ* in language L, at least 71 or 72 is flexible. It can thus be assumed I1 and I2 assign the same domain to every common sort. I naturally assigns this domain for every common sort, whereas it assigns the domain assigned by I*i* for every sort which is only in language L*i*.

Second, as for every sort *τ* in language L, at least 71 or 72 is flexible, I1 and I2 can be assumed to associate the same element to every variable in *V* ∩ *V* '.

Every class *c* of C contains elements of the same sort. If this is a common sort to both I-theories then *c* ∩ (*V* ∪ *V* ') /= ∅, and for every term *t* in *c*, I[*t*] = I1[*t*] = I2[*t*]. If it is not, then every term *t* ∈ *c* belongs to one language L*i*, and I[*t*] = I*i*[*t*].

# Proof of Lemma [4.1](#_bookmark12)

First assume 7 is the three-sorted I-theory and *τ* is the sort of values; *τ* ' is the sort of indices and *σ* the sort of arrays. Let ∆ be a set such that |∆∪ *Dτ* | = *κ*,

' ' '

and such that ∆ ∩ *Dτ*' = ∆ ∩ *Dσ* = ∅. The interpretation I

from interpretation I:

= ⟨*D ,I* ⟩ is built

* *τ* = *Dτ* ∪ ∆, *Dτ* ' = *Dτ*' ;

*D*

'

'

* for any constant *a*, *I*'[*a*] = *I*[*a*];
* for any variable *x*, *I*'[*x*] = *I*[*x*];
* if *v* ∈ *D* , *i* ∈ *D* ' and *a* ∈ *D*

then *I*'[read](*a, i*) = *I*[read](*a, i*) and

*τ τ σ*

*I*'[write](*a, i, v*) = *I*[write](*a, i, v*).

This suffices to make I' a model of *L*, but it is not yet perfectly defined, and not yet a model of 7 . For every element *a* in *Dσ*, there is an element *a*' in *D*' \ *Dσ* corresponding to an array similar to *a* but such that a finite number of elements have been replaced by elements of ∆. More formally:

*σ*

{write(*a, i, v*)*,* write(write(*a, i, v*)*, i*'*, v*')*,.. .*}∈ *D*'

*σ*

\ *Dσ*

' ' '

for {*v, v ,.. .*}⊂ *Dτ* \ *Dτ* and {*i, i ,.. .*}⊂ *Dτ*' . Special care must be taken to

verify the extensionality and read-over-write axioms. Functions “read” and “write” are extended in the natural way.

The two-sorted case, and the three-sorted case when *τ* is the sort of indices are similar. Also notice that, if *κ* ≤ ℵ0, |*Dτ* |≤ ℵ0, |*Dτ*' |≤ ℵ0, and |*Dσ*|≤ ℵ0, then |*D*' |≤ ℵ .

*σ* 0