Combination of disjoint theories: beyond decidability

Pascal Fontaine[1], Stephan Merz[2], and Christoph Weidenbach[3]

[1] Université de Lorraine & LORIA, Nancy, France
[2] INRIA Nancy & LORIA, Nancy, France
[3] Max-Planck-Institut für Informatik, Saarbrücken, Germany

Abstract. Combination of theories underlies the design of satisfiability modulo theories (SMT) solvers. The Nelson-Oppen framework can be used to build a decision procedure for the combination of two disjoint decidable stably infinite theories.
We here study combinations involving an arbitrary first-order theory. Decidability is lost, but refutational completeness is preserved. We consider two cases and provide complete (semi-)algorithms for them. First, we show that it is possible under minor technical conditions to combine a decidable (not necessarily stably infinite) theory and a disjoint finitely axiomatized theory, obtaining a refutationally complete procedure. Second, we provide a refutationally complete procedure for the union of two disjoint finitely axiomatized theories, that uses the assumed procedures for the underlying theories without modifying them.

1 Introduction

The problem of combining *decidable* first-order theories has been widely studied (e.g., [9, 10, 13]). The fundamental result due to Nelson and Oppen yields a decision procedure for the satisfiability (or dually, validity) problem concerning *quantifier-free* formulas in the union of the languages of two decidable theories, provided these theories are *disjoint* (i.e., they only share the equality symbol) and *stably infinite* (i.e., every satisfiable set of literals has an infinite model). This result, and its extensions, underly the design of automated reasoners known as SMT (Satisfiability Modulo Theories [2]) solvers.

The problem of combining theories for which there exist refutationally complete semi-decision procedures for the validity (unsatisfiability) problem has received less attention. In this paper, we will show that the fundamental results about combinations of disjoint decidable theories extend naturally to semi-decidable theories, and that refutationally complete procedures for such theories can be combined to yield a refutationally complete procedure for their union.

From a theoretical point of view, our observation may appear trivial. In particular, consider two theories presented by finitely many first-order axioms: complete first-order theorem provers provide semi-decision procedures for them. A refutationally complete procedure for the union of these theories is simply obtained by running the same prover on the union of the axioms. We believe

however that combining theories à la Nelson-Oppen is still valuable in the two scenarios in Sections 5 and 6. Specialized efficient decision procedures exist for some theories of high practical relevance such as arithmetic fragments, uninterpreted symbols or arrays. In Section 5 we consider the combination of a decidable theory with a disjoint finitely axiomatized theory (without further restriction on cardinalities or on the form of this theory). Using these results, the usual language of SMT solvers can be extended with symbols defined by finitely axiomatized theories, preserving refutational completeness. In Section 6 we consider combining two disjoint finitely axiomatized theories. Here the interest lies in the fact that the refutationally complete procedures for the theories in the combination share very little information; the procedure is essentially parallel.

There has already been work on extending automated first-order theorem provers in order to accommodate interpreted symbols from decidable theories, such as fragments of arithmetic, for which an encoding by first-order axioms does not yield a decision procedure [1, 3]. Bonacina et al. [5] give a calculus for a refutationally complete combination of superposition and SMT solvers. Instantiation-based frameworks (see [2] for more information) also have interesting completeness results. In [8], a complete instantiation procedure is given, even for some cases where theories are not disjoint. Compared to this approach, ours handles a less expressive fragment, but allows working in standard models. Also, our approach imposes no restrictions on the first-order theory and is independent of the actual presentation of the underlying theories or the nature of the semi-decision procedures. It uses the underlying semi-decision procedures as "black boxes" for the combination, in the spirit of the Nelson-Oppen approach.

Our results rely on two main restrictions inherited from Nelson-Oppen. First, we consider unsatisfiability of *quantifier-free* formulas, and second, those formulas are studied in the union of *disjoint* theories. Both restrictions appear crucial, specifically for theories that are not finitely axiomatized. Consider combining Presburger arithmetic (which is decidable) with a first-order and finitely axiomatizable but *non-disjoint* theory defining multiplication in terms of addition. One would expect the result to be non-linear arithmetic on naturals. Because of the unsolvability of Hilbert's tenth problem, there exists no refutationally complete decision procedure for this fragment. Consider also the disjoint union of Presburger arithmetic and the empty theory for uninterpreted symbols (both decidable); considering *quantified* formulas on the union of the languages, it is easy to define multiplication and hence to encode the Hilbert's tenth problem: there cannot be a refutationally complete decision procedure (on the standard model) for arbitrary quantified formulas in the union of these theories. We additionally assume as a reasonable simplification hypothesis that the theories are either decidable or are finitely axiomatized.

Outline. Section 2 fixes basic notations and introduces a pseudo-code language. Sections 3 and 4 present elementary results on combining theories, including the simple case of combining refutationally complete procedures for disjoint stably infinite theories. Lifting the restriction on cardinalities, Section 5 considers combinations of a decidable theory with a disjoint, finitely axiomatized theory.

Finally, we propose in section 6 an algorithm to combine two disjoint finitely axiomatized theories. The results in this paper can of course be used modularly to build complex combinations, involving several decidable theories and several finitely axiomatized theories.

2 Notations

A first-order language $\mathcal{L} = \langle \mathcal{V}, \mathcal{F}, \mathcal{P} \rangle$ consists of an enumerable set $\mathcal{V}$ of variables and enumerable sets $\mathcal{F}$ and $\mathcal{P}$ of function and predicate symbols, associated with their arities. Nullary function symbols are called constant symbols.

Terms and formulas over the language $\mathcal{L}$ are defined in the usual way. An atomic formula is either an equality statement $(t = t')$ where t and t' are terms, or a predicate symbol applied to the corresponding number of terms. Formulas are built from atomic formulas using Boolean connectives ($\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\equiv$) and quantifiers ($\forall$, $\exists$). A literal is an atomic formula or the negation of an atomic formula. A formula with no free variables is closed.

An interpretation $\mathcal{I}$ for a first-order language $\mathcal{L}$ provides a non-empty domain D, a total function $\mathcal{I}[f] : D^r \to D$ of appropriate arity for every function symbol f, a predicate $\mathcal{I}[p] : D^r \to \{\top, \bot\}$ of appropriate arity for every predicate symbol p, and an element $\mathcal{I}[x] \in D$ for every variable x. By extension, an interpretation defines a value in D for every term, and a truth value for every formula. The cardinality of an interpretation is the cardinality of its domain. The notation $\mathcal{I}_{x_1/d_1,\ldots,x_n/d_n}$ where $x_1, \ldots, x_n$ are different variables (or constants) denotes the interpretation that agrees with $\mathcal{I}$, except that it associates $d_i \in D$ to the variable (resp. constant) x_i, for $1 \leq i \leq n$. A model of a formula (resp., a set of formulas) is an interpretation in which the formula (resp., every formula in the set) evaluates to true. A formula is satisfiable if it has a model, and it is unsatisfiable otherwise.

Given an interpretation $\mathcal{I}$ for a first-order language $\mathcal{L} = \langle \mathcal{V}, \mathcal{F}, \mathcal{P} \rangle$, the restriction $\mathcal{I}'$ of $\mathcal{I}$ to $\mathcal{L}' = \langle \mathcal{V}', \mathcal{F}', \mathcal{P}' \rangle$ with $\mathcal{V}' \subseteq \mathcal{V}$, $\mathcal{F}' \subseteq \mathcal{F}$, $\mathcal{P}' \subseteq \mathcal{P}$, is the unique interpretation for $\mathcal{L}'$ similar to $\mathcal{I}$, i.e. $\mathcal{I}'$ and $\mathcal{I}$ have the same domain and assign the same value to symbols in $\mathcal{V}'$, $\mathcal{F}'$ and $\mathcal{P}'$.

A theory $\mathcal{T}$ in a first-order language is a set of interpretations such that, for every interpretation $\mathcal{I} \in \mathcal{T}$, every variable x of the language, and every element d of the domain, $\mathcal{I}_{x/d} \in \mathcal{T}$. A theory may also be defined by a set of closed formulas, in which case it is the set of all the models of the set of formulas. A finite theory or a finitely axiomatized theory is the set of models of a finite set of closed formulas. A constant a is uninterpreted in a theory if for every interpretation $\mathcal{I} \in \mathcal{T}$, and every element d of the domain, $\mathcal{I}_{a/d} \in \mathcal{T}$. The *spectrum* of a theory $\mathcal{T}$, denoted spectrum($\mathcal{T}$), is the set of all (finite or infinite) cardinalities of the interpretations in $\mathcal{T}$. A theory is satisfiable if it is a non-empty set of interpretations; it is unsatisfiable otherwise.

Two theories $\mathcal{T}_1$ and $\mathcal{T}_2$ in languages $\mathcal{L}_1 = \langle \mathcal{V}_1, \mathcal{F}_1, \mathcal{P}_1 \rangle$ and $\mathcal{L}_2 = \langle \mathcal{V}_2, \mathcal{F}_2, \mathcal{P}_2 \rangle$ respectively are disjoint if $\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$ and if $\mathcal{F}_1 \cap \mathcal{F}_2$ only contains constants that are uninterpreted in both $\mathcal{T}_1$ and $\mathcal{T}_2$. The union $\mathcal{T}_1 \cup \mathcal{T}_2$ of two theories $\mathcal{T}_1$

and $\mathcal{T}_2$ (respectively in languages $\mathcal{L}_1 = \langle \mathcal{V}_1, \mathcal{F}_1, \mathcal{P}_1 \rangle$ and $\mathcal{L}_2 = \langle \mathcal{V}_2, \mathcal{F}_2, \mathcal{P}_2 \rangle$) is the largest set of interpretations for language $\mathcal{L} = \langle \mathcal{V}_1 \cup \mathcal{V}_2, \mathcal{F}_1 \cup \mathcal{F}_2, \mathcal{P}_1 \cup \mathcal{P}_2 \rangle$ such that for every $\mathcal{I} \in \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{I}$ restricted to $\mathcal{L}_1$ ($\mathcal{L}_2$) belongs to $\mathcal{T}_1$ (resp., $\mathcal{T}_2$). Notice that the union of two theories defined by sets C_1 and C_2 of closed formulas is exactly the theory defined by the union $C_1 \cup C_2$.

A $\mathcal{T}$-model of a formula G is an interpretation in $\mathcal{T}$ which is a model of G. A formula G is $\mathcal{T}$-satisfiable if it has a $\mathcal{T}$-model, and it is $\mathcal{T}$-unsatisfiable otherwise. A decidable theory $\mathcal{T}$ is a theory such that the $\mathcal{T}$-satisfiability problem for finite sets of ground literals in the language of $\mathcal{T}$ is decidable.

A refutationally complete procedure for $\mathcal{T}$ is a (semi-)algorithm for the $\mathcal{T}$-unsatisfiability problem that will always terminate on an unsatisfiable formula by stating that it is unsatisfiable. Given a satisfiable formula, it may either terminate by stating that it is satisfiable or continue running forever. Thus, a refutationally complete procedure is a decision procedure if and only if it always terminates. A refutationally complete theory $\mathcal{T}$ is a theory such that there exists a refutationally complete procedure for the $\mathcal{T}$-unsatisfiability problem for sets of ground literals in the language of $\mathcal{T}$.

A theory is stably infinite if every $\mathcal{T}$-satisfiable set of literals has an infinite model of cardinality $\aleph_0$.[4]

For convenience, we define the notation $\text{card}_{\geq}(n)$ that denotes a set of literals satisfiable only on models of cardinality at least n (where n is a natural number). Such a cardinality constraint can be enforced by augmenting the set of literals by the set of disequalities $\{a_i \neq a_j \mid 1 \leq i < j \leq n\}$ for fresh constants a_i.

Pseudocode

We will describe our algorithms using pseudocode. Beyond familiar constructs whose semantics is well known, we use the construct **execute in parallel**, which spawns several child processes, explicitly identified using the **process** keyword. These processes can execute truly in parallel, or be subject to any fair interleaving. The **execute in parallel** construct and all its child processes terminate as soon as some child process terminates. Similarly, if child processes are spawned inside a function, any **return** e instruction executed by a child processes will terminate all child processes and make the function return the result e. In our applications, we never have to consider race conditions such as two sibling processes potentially returning different values.

For synchronization, processes can use the instruction **wait** C, where C is a Boolean expression. This instruction blocks the process while C is false, and allows the process to resume as soon as C becomes true.[5] In particular, we use

[4] Traditionally, a theory is said to be stably infinite if every $\mathcal{T}$-satisfiable set of literals has an infinite model. In fact, *a set of first-order formulas* in a countable language (i.e. with a enumerable set of variables, functions, and predicates) has a model with cardinality $\aleph_0$ if it has an infinite model, thanks to the Löwenheim-Skolem theorem.

[5] In our algorithms, we only use synchronization expressions C that never become false after being true, hence the underlying implementation of the **wait** mechanism is unimportant.

the **wait** instruction as syntactic sugar for **wait false** in order to definitely block processes. However, we assume that the **execute in parallel** construct and all its child process terminate if all child processes are waiting. Controlling concurrent accesses to shared variables will be explicitly specified when required.

3 Combining models

In the following, we will restrict our attention to the (un)satisfiability of finite sets of literals. Recall that the $\mathcal{T}$-satisfiability of quantifier-free formulas can be reduced to a series of $\mathcal{T}$-satisfiability checks for finite sets of literals [2]. For example, and disregarding efficient techniques used in SMT solvers, a quantifier-free formula G is satisfiable if and only if the set of literals corresponding to one of the cubes (i.e. one of the conjunctions of literals) in the disjunctive normal form (DNF) of G is satisfiable.

Assume that $\mathcal{T}$ is the union of two disjoint theories $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively in languages $\mathcal{L}_1$ and $\mathcal{L}_2$. By introducing new uninterpreted constants, it is possible to *purify* any finite set of literals L into a $\mathcal{T}$-equisatisfiable set of literals $L_1 \cup L_2$ where each L_i is a set of literals in language $\mathcal{L}_i$ (see e.g. [10]).

Definition 1. *An arrangement $\mathcal{A}$ for a set of constant symbols S is a maximal satisfiable set of equalities and inequalities $a = b$ or $a \neq b$, with $a, b \in S$.*

That is, an arrangement $\mathcal{A}$ for S cannot be consistently extended with any equality or disequality over S which is not already a consequence of $\mathcal{A}$. Obviously, there exist only finitely many arrangements for a finite set of constants.

The following theorem (see also [11, 12, 7]) underlies the completeness proof of combinations of decision procedures. It is also the cornerstone of the results for combining refutationally complete decision procedures.

Theorem 2. *Consider disjoint theories $\mathcal{T}_1$ and $\mathcal{T}_2$, and finite sets of literals L_1 and L_2, respectively in languages $\mathcal{L}_1$ and $\mathcal{L}_2$. $L_1 \cup L_2$ is $\mathcal{T}_1 \cup \mathcal{T}_2$-satisfiable if and only if there exist an arrangement $\mathcal{A}$ of constants shared in L_1 and L_2, a (finite or infinite) cardinality κ, and models $\mathcal{M}_1$ and $\mathcal{M}_2$ of cardinality κ, such that $\mathcal{M}_1$ is a $\mathcal{T}_1$-model of $\mathcal{A} \cup L_1$ and $\mathcal{M}_2$ is a $\mathcal{T}_2$-model of $\mathcal{A} \cup L_2$.*

Intuitively, if a set of literals is satisfiable in the combination of theories, a model of this set defines in a straightforward way an arrangement and two models with the same cardinality for the two sets of literals. The converse is also true: from models of the set of literals augmented with the arrangement, it is possible to build a model for the union, since both models agree on the cardinality and on the interpretation of the shared constants (thanks to the arrangement). The cardinality condition is essential to be able to map elements in the domains of the individual models into a unique domain.

Corollary 3. *Consider disjoint theories $\mathcal{T}_1$ and $\mathcal{T}_2$, and finite sets of literals L_1 and L_2, respectively in languages $\mathcal{L}_1$ and $\mathcal{L}_2$. $L_1 \cup L_2$ is $\mathcal{T}_1 \cup \mathcal{T}_2$-satisfiable if and only if there exists an arrangement $\mathcal{A}$ of constants shared in L_1 and L_2 such that the spectra of $\mathcal{T}_1 \cup L_1 \cup \mathcal{A}$ and $\mathcal{T}_2 \cup L_2 \cup \mathcal{A}$ have a non-empty intersection.*

```
Function check_sat(L1, L2)
1 foreach arrangement A of shared constants of L1 and L2 do
2     if check_sat_arrangement(A, L1, L2) = SAT then
3         return SAT;
4 return UNSAT;

Function check_sat_arrangement(A, L1, L2)
1 if A ∪ L1 is T1-unsatisfiable then
2     return UNSAT;
3 if A ∪ L2 is T2-unsatisfiable then
4     return UNSAT;
5 if spectrum(T1 ∪ A ∪ L1) ∩ spectrum(T2 ∪ A ∪ L2) = ∅ then
6     return UNSAT;
7 return SAT;
```

Algorithm 1: Combination of decidable theories: a generic algorithm.

For combining decision procedures, the cardinality or spectrum requirements of the above theorems are usually fulfilled by assuming properties of the theories in the combination. In the classical combination scheme, the theories are supposed to be stably infinite: if $\mathcal{A} \cup L_i$ has a $\mathcal{T}_i$-model, it also has a $\mathcal{T}_i$-model of infinite cardinality (more precisely, of cardinality $\aleph_0$), and thus the cardinality requirement is trivially fulfilled.

Theorem 2 and Corollary 3 do not require decision procedures to exist, and also apply to refutationally complete theories.

4 Combinations: decidable theories and beyond

Consider two decidable disjoint theories $\mathcal{T}_1$ and $\mathcal{T}_2$ in languages $\mathcal{L}_1$ and $\mathcal{L}_2$, such that, given sets of literals L_1 and L_2 (in $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively), it is computable whether $\text{spectrum}(\mathcal{T}_1 \cup L_1) \cap \text{spectrum}(\mathcal{T}_2 \cup L_2)$ is empty or not. Algorithm 1 is the generic combination algorithm for $\mathcal{T}_1 \cup \mathcal{T}_2$, based on a straightforward application of the theorem in the previous section. Notice that the code at lines 1–4 in function check_sat_arrangement is not required because of the spectrum test at lines 5–6: the combined theory is unsatisfiable if the intersection of the spectra is empty. In case of stably infinite theories however, the spectrum condition at line 5 is guaranteed to be false, and thus lines 5 and 6 are not necessary; without these lines, the algorithm corresponds to the Nelson-Oppen combination framework [9, 10]. If the theories are not stably infinite, there exist many specific results (e.g. [12, 7]) for which it is possible to compute the condition at line 5.

Assume now that the theories in the combination are refutationally complete but not decidable. For the combination procedure to be refutationally complete, it is necessary and sufficient that check_sat_arrangement($\mathcal{A}$, L_1, L_2) terminates

```
Function check_sat_arrangement(A, L1, L2)
1  execute in parallel
2      process
3          if A ∪ L1 is T1-unsatisfiable then
4              return UNSAT;
5          wait;
6      process
7          if A ∪ L2 is T2-unsatisfiable then
8              return UNSAT;
9          wait;
10 return SAT;
```

Algorithm 2: Nelson-Oppen for refutationally complete procedures.

if $\mathcal{A} \cup L_1 \cup L_2$ is $\mathcal{T}_1 \cup \mathcal{T}_2$-unsatisfiable. Otherwise one could not guarantee that a call to check_sat($\mathcal{A} \cup L_1$, $\mathcal{A} \cup L_2$) would return. The following algorithms for refutationally complete procedures are all based on the above check_sat function, but differ in the check_sat_arrangement function. In the following, we say that a function check_sat_arrangement is a refutationally complete procedure for a theory, if function check_sat together with the considered function yields a refutationally complete procedure for a theory.

The Nelson-Oppen schema traditionally gets rid of the spectrum condition in Algorithm 1 by considering only stably infinite theories. As a first step, we also restrict attention to stably infinite theories. Thus the intersection of the spectra is empty only if one of $\mathcal{T}_1 \cup \mathcal{A} \cup L_1$ or $\mathcal{T}_2 \cup \mathcal{A} \cup L_2$ is unsatisfiable.

In the case of refutationally complete theories, the sequentiality of Algorithm 1 may cause a completeness problem. Indeed it may happen that $\mathcal{A} \cup L_1$ is $\mathcal{T}_1$-satisfiable and the test at line 1 in function check_sat_arrangement never terminates; $\mathcal{A} \cup L_2$ would never be checked for unsatisfiability. This behavior breaks completeness. A natural way to circumvent this problem is to run the unsatisfiability tests in parallel, as in Algorithm 2.

Theorem 4. *Assume $\mathcal{T}_1$ and $\mathcal{T}_2$ are stably infinite and disjoint theories with refutationally complete procedures for finite sets of literals. Algorithm 2 yields a refutationally complete procedure for $\mathcal{T}_1 \cup \mathcal{T}_2$ for finite sets $L_1 \cup L_2$ of literals, where L_1 and L_2 are respectively literals in the language of $\mathcal{T}_1$ and $\mathcal{T}_2$. If $\mathcal{T}_1$ and $\mathcal{T}_2$ are furthermore decidable, Algorithm 2 is a decision procedure.*

Proof. The proof is similar to that for the Nelson-Oppen combination framework for disjoint stably infinite decidable theories. First notice that soundness is a direct consequence of Theorem 2: whenever the algorithm terminates, it provides the right answer. We only need to ensure termination in the unsatisfiable case.

The algorithm terminates for $\mathcal{T}_1 \cup \mathcal{T}_2$-unsatisfiable sets of literals. The finite sets of L_1 and L_2 only share a finite set of constants S. There exist only a finite number of arrangements $\mathcal{A}_1, \ldots \mathcal{A}_n$ of S, and these can be checked in sequence

for unsatisfiability. If every call to function check_sat_arrangement($\mathcal{A}$, L_1, L_2) terminates when $\mathcal{A} \cup L_1 \cup L_2$ is $\mathcal{T}_1 \cup \mathcal{T}_2$-unsatisfiable, then the algorithm is a refutationally complete procedure for $\mathcal{T}_1 \cup \mathcal{T}_2$.

A call to check_sat_arrangement($\mathcal{A}$, L_1, L_2) does not terminate for some $\mathcal{A} \cup L_1 \cup L_2$ only if $\mathcal{A} \cup L_1$ is $\mathcal{T}_1$-satisfiable and $\mathcal{A} \cup L_2$ is $\mathcal{T}_2$-satisfiable. In that case, there exist a $\mathcal{T}_1$-model $\mathcal{M}_1$ of $\mathcal{A} \cup L_1$ and a $\mathcal{T}_2$-model $\mathcal{M}_2$ of $\mathcal{A} \cup L_2$. The cardinality of the models can be assumed to be $\aleph_0$. Thus, according to Theorem 2, $\mathcal{A} \cup L_1 \cup L_2$ is $\mathcal{T}_1 \cup \mathcal{T}_2$-satisfiable. □

As a concrete example, consider Presburger arithmetic – which is decidable and has only models of infinite cardinality – and a finite set of first-order formulas with only infinite models (e.g. including axioms for dense order), for which refutationally complete procedures exist (any complete first-order logic prover is a suitable procedure). The above theorem yields a refutationally complete procedure for quantifier-free formulas in the union of the languages.

The above theorem imposes two major constraints, one on the cardinalities, the other on the disjointness. For decision procedures, relaxing the cardinality constraints is possible using asymmetric combinations, where one theory in the combination has strong properties that allow to relax the cardinality property on the other one. We investigate this solution in the next section.

5 Combining a decidable and an arbitrary theory

The restriction to stably infinite theories has proved to be useful in the decidable case: the Nelson-Oppen framework is simple and efficient, and several important decidable theories are indeed stably infinite. Still, being stably infinite is a strong constraint (e.g., the theory $\forall x \,.\, x = a \vee x = b$ is not stably infinite), and there is no general procedure to check whether a theory is stably infinite. In practice, most theories (and all theories actually implemented in SMT solvers) furthermore have other spectral properties that allow for less restrictive combinations.

When theories are not stably infinite, the spectrum condition of lines 5-6 in Algorithm 1 becomes important. Indeed, even if both $\mathcal{T}_1 \cup \mathcal{A} \cup L_1$ and $\mathcal{T}_2 \cup \mathcal{A} \cup L_2$ are satisfiable, $L_1 \cup L_2$ may still be $\mathcal{T}_1 \cup \mathcal{T}_2$-unsatisfiable because the spectra of the models are disjoint. However, the condition is not directly implementable in general. In this section, we consider the case of a decidable theory for which the spectrum is computable. It can then be translated into suitable constraints for the procedure for the (refutationally complete) theory $\mathcal{T}_2$.

Lemma 5. *Assume $\mathcal{T}$ is a finitely axiomatized theory. There exists a refutationally complete procedure for $\mathcal{T}$ restricted to models of cardinalities belonging to a given set of the following nature:*

1. *one or all infinite cardinalities;*
2. *a finite set of finite cardinalities;*
3. *all cardinalities larger than a fixed finite cardinality;*
4. *the complement of a finite set of finite cardinalities.*

```
Function check_sat_arrangement(A, L1, L2)
1 if A ∪ L1 is T1-unsatisfiable then
2     return UNSAT;
3 if spectrum(T1 ∪ A ∪ L1) ∩ spectrum(T2 ∪ A ∪ L2) = ∅ then
4     return UNSAT;
5 return SAT;
```

Algorithm 3: Combination of a decidable theory with a refutationally complete theory.

It is a decision procedure for the second case.

Proof. It suffices to show that the $\mathcal{T}$-unsatisfiability of a finite set of literals with the given cardinality constraints can be reduced to checking the unsatisfiability for another finitely axiomatized theory $\mathcal{T}'$. The theory $\mathcal{T}'$ would simply be the union of $\mathcal{T}$, the set of literals, and a formula encoding the restriction on the cardinality.

It is easy to restrict the unsatisfiability check to infinite cardinalities, e.g. by adding a formula of the form

$$\forall x \neg R(x,x) \wedge \forall x \exists y \Big(R(x,y) \wedge \forall z \left(R(y,z) \Rightarrow R(x,z) \right) \Big)$$

to the set, where R is a fresh relation symbol. To restrict the unsatisfiability check to a given finite cardinality n, it suffices to add a constraint of the form

$$\bigwedge_{1 \leq i < j \leq n} a_i \neq a_j \wedge \ \forall x \bigvee_{1 \leq i \leq n} x = a_i$$

where the a_i are fresh constants. In fact, checking the satisfiability of a finitely axiomatized theory with a given finite cardinality is trivially a decidable problem since there are essentially finitely many interpretations for a finite language with a given finite domain. For a finite set of finite cardinalities, it suffices to take the disjunction of constraints for each finite cardinality in the set. Again, this constitutes a finite set of decidable problems, which is therefore itself decidable.

The third case is simply handled by adding a constraint $\mathrm{card}_{\geq}(n)$. The complement of a finite set of finite cardinalities is the union of a finite set of finite cardinalities and all cardinalities larger than a finite cardinality. A suitable constraint for the final case is thus the disjunction of a constraint for the third case and a constraint for the second case. □

Theorem 6. *Assume $\mathcal{T}_2$ is a finitely axiomatized theory, and $\mathcal{T}_1$ is a decidable theory for which the spectrum is computable and falls in the cases referred in Lemma 5. Then $\mathcal{T}_1 \cup \mathcal{T}_2$ is refutationally complete, and Algorithm 3 yields a refutationally complete procedure for it. The procedure is decidable in the second case. It is also decidable in the two last cases if $\mathcal{T}_2$ is decidable.*

Proof. By the previous lemma, the test at line 3 is implementable by an unsatisfiability procedure that terminates whenever $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{A} \cup L_1 \cup L_2$ is unsatisfiable. Furthermore, the test at line 3 is guaranteed to terminate if the spectrum of $\mathcal{T}_1 \cup \mathcal{A} \cup L_1$ is a finite set of finite cardinalities. Finally, in the two last cases, checking that spectrum$(\mathcal{T}_2 \cup \mathcal{A} \cup L_2)$ has a non-empty intersection with a given co-finite set can be reduced to checking the satisfiability of a finite collection of sets of literals $\mathcal{T}_2 \cup \mathcal{A} \cup L_2 \cup C$ where C is a set of literals encoding a constraint on cardinality; this is decidable if the satisfiability problem for sets of literals in $\mathcal{T}_2$ is decidable. □

The above conditions on the spectrum of the decidable theories appear reasonable: the decidable theories considered in combinations of theories usually fall in one of the categories of the theorem. Gentle theories [6] have a spectrum which is computable and either a finite set of finite cardinalities or a co-finite set of cardinalities. Shiny theories [12] have a computable spectrum that falls into case (3). Linear arithmetic on integers or reals obviously belongs to the first category.

6 Parallel refutation of a union of disjoint theories

In the previous section, we concentrated on combining a decidable theory with another for which a refutationally complete decision procedure exists. In this section, we study the combination of two refutationally complete theories, dropping the cardinality requirement imposed in Section 4. In the context of SMT, it seems fairly natural to restrict our study to theories that can be represented by a finite number of first-order axioms. These theories not only have well-known refutationally complete procedures in the form of complete theorem provers for first-order logic, but it is also decidable if they have a model of a given finite cardinality. Another property of finitely axiomatized theories that we will use is the Skolem-Löwenheim Theorem: such theories have either a finite number of finite models, or they have models for all infinite cardinalities.

Algorithm 4 presents a refutationally complete procedure for the combination of two disjoint first-order theories. It basically interleaves or parallelizes the run of both a refutationally complete procedure and a finite model finder for a set of literals, for $\mathcal{T}_1$ and $\mathcal{T}_2$. The task of the finite model finder is to check if the set of literals is satisfiable on a model in the theory with a given finite cardinality. Very schematically, the finite model finders and the refutationally complete procedures may not terminate in case $\mathcal{A} \cup L_1$ and $\mathcal{A} \cup L_2$ are respectively $\mathcal{T}_1$- and $\mathcal{T}_2$-satisfiable and have no finite model. In such a case they must have infinite models; thanks to the Löwenheim-Skolem theorems one can ensure that the spectra have a non-empty intersection and so $\mathcal{A} \cup \mathcal{T}_1 \cup \mathcal{T}_2$ must be satisfiable.

The difficulties come from the facts that:

- it is necessary to stop and restart the unsatisfiability checker whenever a model is found for some cardinality. The unsatisfiability checker may run forever in that case, and it may be required to check if the set is unsatisfiable for *larger* cardinalities.

```
Function check_sat_arrangement(A, L1, L2)
 1  k1 := 1; S1 := ∅;
 2  k2 := 1; S2 := ∅;
 3  execute in parallel
 4      process
 5          for ever do
 6              k1' := k1;
 7              execute in parallel
 8                  process
 9                      while ¬find_model(k1, T1, A ∪ L1) do
10                          k1 := k1 + 1;
11                  process
12                      if A ∪ L1 ∪ card≥(k1') is T1-unsatisfiable then
13                          wait k2 ≥ k1';
14                          return UNSAT;
15                      wait;
16              S1 := S1 ∪ {k1};
17              if k1 ∈ S2 then
18                  return SAT;
19              k1 := k1 + 1;
20      process
21          for ever do
22              k2' := k2;
23              execute in parallel
24                  process
25                      while ¬find_model(k2, T2, A ∪ L2) do
26                          k2 := k2 + 1;
27                  process
28                      if A ∪ L2 ∪ card≥(k2') is T1-unsatisfiable then
29                          wait k1 ≥ k2';
30                          return UNSAT;
31                      wait;
32              S2 := S2 ∪ {k2};
33              if k2 ∈ S1 then
34                  return SAT;
35              k2 := k2 + 1;
```

Algorithm 4: Combination of two finitely axiomatized theories.

– for completeness, it is however necessary to eventually leave the unsatisfiability checker run undisturbed for ever longer periods;
– if one theory is found not to have models with cardinality greater than k, it is necessary, before returning the answer "unsatisfiable", to wait for the other procedure to check all interpretations of cardinality up to k.

The finite model finder is called at lines 9 and 25; find_model$(k_1, \mathcal{T}_1, \mathcal{A} \cup L_1)$ returns true if and only if $\mathcal{A} \cup L_1$ has a $\mathcal{T}_1$-model of cardinality k_1. A call to the finite model finder should eventually terminate. It is not required for the unsatisfiability checks at lines 12 and 28 to return, in case the considered formulas are satisfiable.

The shared variables are k_1, k_2, $\mathcal{S}_1$ and $\mathcal{S}_2$. The value of k_2 read by the process for $\mathcal{T}_1$ does not have to be up to date. It is however mandatory for the correctness of the algorithm that the age of the value (i.e. the difference between the current time and the last time for which k_2 had this value for the process for $\mathcal{T}_2$) stays bounded. The symmetric requirement exists for k_1. We assume that there is a critical section wrapping the reading and writing of $\mathcal{S}_1$ and $\mathcal{S}_2$, and that both processes have the same view of those variables. The order of lines 16 and 17–18 and of lines 32 and 33–34 matters. A completeness bug would result from switching those lines.

Theorem 7. *Assume $\mathcal{T}_1$ and $\mathcal{T}_2$ are two disjoint finitely axiomatized theories, then Algorithm 4 yields a refutationally complete procedure for $\mathcal{T}_1 \cup \mathcal{T}_2$.*

Proof. In the following, the process for $\mathcal{T}_1$ denotes lines 5 to 19, and the process for $\mathcal{T}_2$ denotes lines 21 to 35.

It is useful to show that the following invariant properties hold throughout the execution of the algorithm after the initialization at lines 1-2:

1. $I_1 : \mathcal{S}_1 \cap \{k \mid k \geq k_1\} = \emptyset$, at all times except when process for $\mathcal{T}_1$ is executing instructions at lines 16-19
2. $I_2 : \mathcal{S}_2 \cap \{k \mid k \geq k_2\} = \emptyset$, at all times except when process for $\mathcal{T}_2$ is executing instructions at lines 32-35
3. $I_3 : \mathcal{S}_1 \cap \{1, \ldots, k_1 - 1\} = \text{spectrum}(\mathcal{T}_1 \cup \mathcal{A} \cup L_1) \cap \{1, \ldots, k_1 - 1\}$
4. $I_4 : \mathcal{S}_2 \cap \{1, \ldots, k_2 - 1\} = \text{spectrum}(\mathcal{T}_2 \cup \mathcal{A} \cup L_2) \cap \{1, \ldots, k_2 - 1\}$
5. $I_5 : \mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$, at all times except when process for $\mathcal{T}_1$ (or $\mathcal{T}_2$) is executing instructions at lines 16-18 (resp. instructions at lines 32-34)

Note that all the above invariants are established by the initialization.

The first two invariants I_1 and I_2 are fairly easy to prove. To prove I_1 notice that every addition (of k_1) to $\mathcal{S}_1$ is done at line 16, and immediately followed (otherwise the function terminates) at line 19 by incrementing k_1. Symmetrically for I_2.

For invariant I_3, it is sufficient to show that, if the property is true before executing line 10, it is true after, and if it is true before executing line 19, it is true after. Adding k_1 to $\mathcal{S}_1$ (at line 16) cannot modify the truth status of I_1 since $\mathcal{S}_1 \cap \{1, \ldots, k_1 - 1\}$ does not change. At line 10, k_1 is incremented only if $\mathcal{A} \cup L_1$ does not have any $\mathcal{T}_1$-models with k_1 elements, i.e. only if $k_1 \notin$

spectrum($\mathcal{T}_1 \cup \mathcal{A} \cup L_1$); thanks to I_1, $k_1 \notin S_1$. And incrementing k_1 at line 19 is only done if the loop at lines 9-10 terminates, that is, if a $\mathcal{T}_1$-model with cardinality k_1 is found for $\mathcal{A} \cup L_1$. Symmetrically for I_4.

To show that I_5 is true, it is sufficient to consider the group of lines 16–18 and 32–34, where $\mathcal{S}_1$ and $\mathcal{S}_2$ are modified. Assume I_5 holds before executing lines at 16–18, and assume the process for $\mathcal{T}_2$ is not running the section between lines 32 and 34. While adding k_1 to $\mathcal{S}_1$, if k_1 also belongs to $\mathcal{S}_2$, the function will return at line 18. The argument is similar for modifications of $\mathcal{S}_2$. Notice however that it is necessary to guarantee that the values of $\mathcal{S}_1$ and $\mathcal{S}_2$ read by the processes for $\mathcal{T}_2$ and $\mathcal{T}_1$ respectively are up to date. Otherwise, the algorithm may "miss" the check of some cardinality.

As a consequence of invariants I_3 and I_4, and thanks to Corollary 3, the algorithm terminates by returning SAT only if $\mathcal{A} \cup L_1 \cup L_2$ is indeed $\mathcal{T}_1 \cup \mathcal{T}_2$-satisfiable (with a model of finite cardinality).

As a consequence of invariants I_3, I_4 and I_5, it can be deduced at line 14 that the spectra are disjoint. If the algorithm returns UNSAT at line 14, then $\mathcal{A} \cup L_1 \cup L_2$ is indeed $\mathcal{T}_1 \cup \mathcal{T}_2$-unsatisfiable. The discussion is symmetric for line 30. Thus the algorithm is sound: a returned answer is always correct. It remains to show that the algorithm is complete, that is, it eventually terminates if $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{A} \cup L_1 \cup L_2$ is unsatisfiable.

We prove termination by contradiction, and assume the function never terminates, although the spectra for $\mathcal{T}_1 \cup \mathcal{A} \cup L_1$ and $\mathcal{T}_2 \cup \mathcal{A} \cup L_2$ are disjoint. Remember the Löwenheim-Skolem Theorem that states that a finitely axiomatized theory with (a) an infinite spectrum has models of all infinite cardinalities; (b) an infinite model has models for every infinite cardinality. Thus, if spectrum($\mathcal{T}_1 \cup \mathcal{A} \cup L_1$) and spectrum($\mathcal{T}_2 \cup \mathcal{A} \cup L_2$) are disjoint, at least one of those sets must be a finite set of finite cardinalities. Assume without loss of generality that spectrum($\mathcal{T}_1 \cup \mathcal{A} \cup L_1$) is finite. Then, there exists some k (assume k is the smallest integer) such that spectrum($\mathcal{T}_1 \cup \mathcal{A} \cup L_1$) $\cap \{k' \mid k' \geq k\} = \emptyset$, that is, such that $\mathcal{A} \cup L_1 \cup \text{card}_{\geq}(k)$ is $\mathcal{T}_1$-unsatisfiable.

Notice that if $\mathcal{T}_1 \cup \mathcal{A} \cup L_1$ has a model of finite cardinality $k-1$ then k_1 will eventually reach k (provided the finite model finder at line 9 is terminating). After that, the process for $\mathcal{T}_1$ will let the finite model finder run forever searching for non-existent models of cardinality greater than or equal to k, while in the meantime, the refutationally complete procedure at line 12 for the $\mathcal{T}_1$-unsatisfiability of $\mathcal{A} \cup L_1 \cup \text{card}_{\geq}(k'_1)$ will run undisturbed (that is, will never be killed) for the amount of time necessary for it to terminate. The process for $\mathcal{T}_1$ will eventually reach line 13. If spectrum($\mathcal{T}_2 \cup \mathcal{A} \cup L_2$) is infinite, k_2 will grow to infinity and will eventually be greater than k'_1. The waiting process for $\mathcal{T}_1$ will eventually be leaving the waiting state and return UNSAT.

Notice (symmetrically as before) that if $\mathcal{T}_2 \cup \mathcal{A} \cup L_2$ has a model of finite cardinality then k_2 will eventually reach and overstep this cardinality (provided the finite model finder at line 26 is terminating). If spectrum($\mathcal{T}_2 \cup \mathcal{A} \cup L_2$) is finite, and if its maximal element is greater or equal than the value k'_1 reached by process $\mathcal{T}_1$ on the waiting state, the waiting process for $\mathcal{T}_1$ will eventually be

leaving the waiting state and return UNSAT. The case where its maximal element is strictly smaller than value $k'_1 - 1$ does not need to be considered, by symmetry.

Let us consider finally the degenerated case for which both spectra have the same maximal value, and thus $k'_1 = k'_2$, and suppose that both processes are waiting, at instruction 13 and 29 respectively. k'_1 and k'_2 must then both be strictly greater than 1. This cannot happen: spectrum($\mathcal{T}_1 \cup \mathcal{A} \cup L_1$) should contain $k'_1 - 1$ and spectrum($\mathcal{T}_2 \cup \mathcal{A} \cup L_2$) should contain $k'_2 - 1$, i.e. $k'_1 - 1$, which contradicts invariant I_5. □

Algorithm 4 eventually terminates if $\mathcal{A} \cup L_1 \cup L_2$ is $\mathcal{T}_1 \cup \mathcal{T}_2$-unsatisfiable, or if $\mathcal{A} \cup L_1 \cup L_2$ is $\mathcal{T}_1 \cup \mathcal{T}_2$-satisfiable in a finite model. It is both a complete refutation procedure and a finite model finder.

7 Conclusion

We studied two cases of refutationally complete combination of disjoint theories. In the first case, we considered combining a decidable theory with an arbitrary finitely axiomatized theory. In the second, we provided an algorithm to combine two finitely axiomatized theories. Both these algorithms are not yet efficiently implemented and would require further techniques and heuristics to be turned into useful solvers. Just as in the case of decidable theories, it is not realistic to consider every arrangement separately: techniques developed for the combination of decision procedures — e.g. cooperation by equality exchange, or delayed theory combination (see e.g. [2]) — will have to be transposed, but will require further research since it can not be expected that the components in the combinations will eventually terminate. Similarly, the negotiation of suitable cardinalities will require specific methods and heuristics.

Bonacina et al. [4] show that, in the case of decidable and universal finitely axiomatized theories, it is possible (and sufficient for some combinations) to extract cardinality constraints from the saturation of a superposition solver. We here considered refutationally complete procedures as black boxes, but, among the potential heuristics to make the approach work in practice, one could consider extracting cardinality hints from the saturation provers, to improve the cardinality negotiation in Algorithm 4.

In [14], tableaux are used to combine in a refutationally complete way two theories sharing a dense order. In this case, the difficulty of the combination does not lie in the agreement on cardinalities (since both theories can only have infinite models), but in the non-disjoint signature containing this order predicate. As an extension of the present work, it would be interesting to find practical ways to combine loosely connected rather than disjoint theories, such as theories sharing a few unary predicates.

Acknowledgment. We would like to thank Christophe Ringeissen and Michaël Rusinowitch for pointing out some related works. We also thank the anonymous reviewers for their comments.

References

1. E. Althaus, E. Kruglov, and C. Weidenbach. Superposition modulo linear arithmetic sup(la). In S. Ghilardi and R. Sebastiani, editors, *FroCoS*, volume 5749 of *LNCS*, pages 84–99. Springer, 2009.
2. C. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. Satisfiability modulo theories. In A. Biere, M. J. H. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 26, pages 825–885. IOS Press, Feb. 2009.
3. P. Baumgartner and C. Tinelli. Model evolution with equality modulo built-in theories. In N. Bjørner and V. Sofronie-Stokkermans, editors, *CADE*, volume 6803 of *LNCS*, pages 85–100. Springer, 2011.
4. M. P. Bonacina, S. Ghilardi, E. Nicolini, S. Ranise, and D. Zucchelli. Decidability and undecidability results for Nelson-Oppen and rewrite-based decision procedures. In U. Furbach and N. Shankar, editors, *IJCAR*, volume 4130 of *LNCS*, pages 513–527. Springer, 2006.
5. M. P. Bonacina, C. Lynch, and L. M. de Moura. On deciding satisfiability by DPLL(G+t) and unsound theorem proving. In R. A. Schmidt, editor, *CADE*, volume 5663 of *LNCS*, pages 35–50. Springer, 2009.
6. P. Fontaine. Combinations of theories for decidable fragments of first-order logic. In S. Ghilardi and R. Sebastiani, editors, *FroCoS*, volume 5749 of *LNCS*, pages 263–278. Springer, 2009.
7. P. Fontaine and E. P. Gribomont. Combining non-stably infinite, non-first order theories. In W. Ahrendt, P. Baumgartner, H. de Nivelle, S. Ranise, and C. Tinelli, editors, *Selected Papers from the Workshops on Disproving and the Second International Workshop on Pragmatics of Decision Procedures (PDPAR 2004)*, volume 125 of *ENTCS*, pages 37–51, July 2005.
8. Y. Ge and L. M. de Moura. Complete instantiation for quantified formulas in satisfiabiliby modulo theories. In A. Bouajjani and O. Maler, editors, *CAV*, volume 5643 of *LNCS*, pages 306–320. Springer, 2009.
9. G. Nelson and D. C. Oppen. Simplifications by cooperating decision procedures. *ACM Trans. on Programming Languages and Systems*, 1(2):245–257, Oct. 1979.
10. C. Tinelli and M. T. Harandi. A new correctness proof of the Nelson–Oppen combination procedure. In F. Baader and K. U. Schulz, editors, *FroCoS*, Applied Logic, pages 103–120. Kluwer Academic Publishers, Mar. 1996.
11. C. Tinelli and C. Ringeissen. Unions of non-disjoint theories and combinations of satisfiability procedures. *Theoretical Computer Science*, 290(1):291–353, Jan. 2003.
12. C. Tinelli and C. G. Zarba. Combining non-stably infinite theories. *Journal of Automated Reasoning*, 34(3):209–238, 2005.
13. T. Wies, R. Piskac, and V. Kuncak. Combining theories with shared set operations. In S. Ghilardi and R. Sebastiani, editors, *FroCoS*, volume 5749 of *LNCS*, pages 366–382. Springer, 2009.
14. C. G. Zarba, Z. Manna, and H. B. Sipma. Combining theories sharing dense orders. In *TABLEAUX, Position Papers and Tutorials*, pages 83–98, 2003. Technical Report RT-DIA-80-2003.