Satisfiability Modulo Non-Disjoint Combinations of Theories Connected via Bridging Functions

Work in progress

Paula Chocron[1,3], Pascal Fontaine[2], and Christophe Ringeissen[3*]

[1] Universidad de Buenos Aires, Argentina
[2] INRIA, Université de Lorraine & LORIA, Nancy, France
[3] INRIA & LORIA, Nancy, France

1 Introduction

Solving the satisfiability problem modulo a theory given as a union of decidable sub-theories naturally calls for combination methods. The Nelson-Oppen combination method [11] has been developed more than 30 years ago, and is now ubiquitous in SMT (Satisfiability Modulo Theories) solvers. However, this technique imposes strong assumptions on the theories in the combination; in the classical scheme [11,19], the theories notably have to be signature-disjoint and stably infinite. Many recent advances aim to go beyond these two limitations.

The design of a combination method for non-disjoint unions of theories is clearly a hard task [20,9]. To stay within the frontiers of decidability, it is necessary to impose restrictions on the theories in the combination; and at the same time, those restrictions should not be such that there is no hope of concrete applications for the combination scheme. For this reason, it is worth exploring specific classes of non-disjoint combinations of theories that appear frequently in software specification, and for which it would be useful to have a simple combination procedure. An example is the case of shared sets, sets being represented by unary predicates [21,6]. In this context, the cardinality operator can also be considered; notice that this operator is a bridging function from sets to natural numbers [24]. In this paper, we investigate the case of bridging functions between data structures and a target theory (e.g. a fragment of arithmetic). Here, non-disjointness arises from connecting two disjoint theories via a third theory defining the bridging function. This problem has attracted a lot of interest in the last few years [25,8,3,16,17] due to its importance for solving verification problems expressed in a combination of data structures and arithmetic. With this work, we want to provide a synthesis of several previous contributions by different authors based on different techniques and frameworks. We mainly focus on the following papers that are closely related in terms of the considered data structures:

* This work has been partially supported by the project ANR-13-IS02-0001 of the Agence Nationale de la Recherche, by the European Union Seventh Framework Programme under grant agreement no. 295261 (MEALS), and by the STIC AmSud MISMT

– Zarba presents a procedure for checking satisfiability of lists with length by using a reduction to the arithmetic [23]. The same kind of reduction is applied to multisets with multiplicity [22]. A goal was to relax the stably-infiniteness assumption in Nelson-Oppen's procedure; it is indeed possible to consider for instance multisets with a finite domain for elements. In his work, Zarba is able to reduce the problem into one expressed only in the theory of elements; the theory of lists (multisets) completely vanishes.
– Sofronie-Stokkermans [16] uses a locality property to show that the definition of the function connecting the theories can be instantiated (without loss of completeness) by the ground terms occurring in the input formula. She also considers the delicate problem of restricting models to standard ones (for data structures). A drawback of her solution is that it excludes cases where cardinality problems might arise from lack of elements to build structures that must be different.
– In [17], Suter and al. present a procedure to solve cardinality problems that is also based on a procedure for reducing bridging functions. As future work, they suggest to study relations between their work and the one in [16].

We investigate here an approach by reduction from non-disjoint to disjoint combination. The outcome of our approach is very close to that of the locality-based approach [16]. It is an alternative to a non-disjoint combination approach à la Ghilardi [9], for which some assumptions on the shared (target) theory are required. Ghilardi's approach has been applied to combine data structures with fragments of arithmetic, like Integer Offsets [13] and then Abelian groups [12]; it is however difficult to go beyond Abelian groups and consider for instance any decidable fragment of arithmetic as a shared theory. The approach by reduction does not impose such limitations, and any (decidable) fragment of arithmetic is suitable for the target (shared) theory.

The superposition calculi provide elegant and uniform ways to build satisfiability procedures for (combinations of) data-structures [2,1], possibly with bridging functions [13,12,10,4]. It appears that the approach by reduction is applicable to many data structures for which the standard superposition calculus can be used as an off-the-shelf underlying satisfiability procedure [2,1]. This approach by reduction leads to a combination procedure (see Section 3) which is indeed correct for a large class of data structure theories, ranging from the theory of equality to the theory of absolutely free data structures. Our correctness proof is not (directly) based on locality principles, but we rely on the form of Herbrand models we can expect from the data structure theories we are interested in.

When considering data structures, it is quite natural to restrict to *standard* interpretations. For instance, the standard interpretation for lists corresponds to the case where lists are interpreted as finite lists of elements. We show how to adapt the combination procedure to get a satisfiability procedure on standard interpretations, when the bridging function is *stable*. The notion of stable function encompasses both bijectivity and infinite surjectivity [17]. Moreover, we propose an enumeration procedure which has similarities with the procedure studied in [17,18,14]. This enumeration procedure allows to revisit the satisfiability problem in the standard interpretation of lists with length [8]. More generally, we conjecture that this procedure can be applied to data structures satisfying some gentle properties as defined in [7].

The work presented in this short paper corresponds to a part of [5], where the enumeration procedure mentioned above is detailed in the case of lists with a length function and its correctness is proven. We are now working on a full paper extending the short presentation given below.

2 The Combination Problem

We assume the reader is familiar with the classical notions and notations used in first-order logic with equality. By a slight abuse of notation, we write that a sort occurs in a signature if the sort belongs to the set of sorts of the signature.

Consider a many-sorted Σ_s-theory T_s and a many-sorted Σ_t-theory T_t (s and t stand for source and target respectively) such that Σ_s and Σ_t have no shared function symbols and no shared predicate symbols except the equality predicates: we have a shared equality predicate for each shared sort occurring in both Σ_s and Σ_t. Roughly speaking, we consider a function f mapping elements from T_s to elements in T_t. This function is defined by some axioms expressed in the signature $\Sigma_s \cup \Sigma_t \cup \{f\}$. The set of axioms defining f is called T_f.

The difficulty of building a decision procedure for the theories connected with the bridging function depends on many factors, for example, how f is defined. In some cases there exists a very simple solution: since we are dealing with first order logic, f always occurs applied to the appropriate number of terms. In all those occurrences f could be substituted by its definition. The result may then be a disjoint problem. This is possible when f is defined by an equality like $f(x) = e$, for some Σ_t-term e. This naive approach is not suitable for more complicated definitions. Particularly, it cannot be used for recursively defined bridging functions, like those commonly found for data structures. We introduce a procedure dedicated to that problem. The idea is to eliminate the function symbol f, expressing its definition using just $\Sigma_s \cup \Sigma_t$. If this maintains satisfiability, our problem has been reduced to a disjoint one, and any combination method we know for this problem can be used.

Let us now introduce the theories T_s, T_t and T_f we focus on. The theory T_s is the theory of Absolutely Free Data Structures [16] (AFDS, for short) as defined below, and T_f is a bridging theory connecting it to another theory T_t.

Definition 1. *Consider a set of sorts* $\mathtt{Elem}$*, and a sort* $\mathtt{struct} \notin \mathtt{Elem}$*. Let* Σ *be a signature whose set of sorts is* $\{\mathtt{struct}\} \cup \mathtt{Elem}$ *and whose function symbols* $c \in \Sigma$ *(called* constructors*) have arities of the form:*

$$c : s_1 \times \cdots \times s_m \times \mathtt{struct} \times \cdots \times \mathtt{struct} \to \mathtt{struct}$$

where $s_1, \dots, s_m \in$ `Elem`. Consider the following axioms (where variables are implicitly universally quantified)

$$\begin{cases} (\mathit{Inj}_c) & c(X_1, \dots, X_n) = c(Y_1, \dots, Y_n) \Rightarrow \bigwedge_{i=1}^{n} X_i = Y_i \\ (\mathit{Clash}_{c,d}) & c(X_1, \dots, X_n) \neq d(Y_1, \dots, Y_m) \\ (\mathit{Acyc}_{\Sigma}) & X \neq t[X] \text{ if } t \text{ is a non-variable } \Sigma\text{-term} \end{cases}$$

The theory of Absolutely Free Data Structures over Σ is

$$\mathit{AFDS}_{\Sigma} = \big(\bigcup_{c \in \Sigma} \mathit{Inj}_c\big) \cup \big(\bigcup_{c,d \in \Sigma, c \neq d} \mathit{Clash}_{c,d}\big) \cup \mathit{Acyc}_{\Sigma}$$

Example 1. The theory of lists is an example of AFDS where the constructors are $\mathit{cons} : \texttt{elem} \times \texttt{list} \to \texttt{list}$ and $\mathit{nil} : \texttt{list}$. The theory of pairs (of numbers) is another example of AFDS where the constructor is $\mathit{pair} : \texttt{num} \times \texttt{num} \to \texttt{struct}$.

For sake of simplicity, we only consider Absolutely Free Data Structures without selector functions, but it would be easy to consider these additional functions in the presented framework.

Given a tuple $\boldsymbol{e}$ of terms of sorts in `Elem` and a tuple $\boldsymbol{t}$ of terms of sort `struct`, the tuple $\boldsymbol{e}, \boldsymbol{t}$ may be written $\boldsymbol{e}; \boldsymbol{t}$ to distinguish terms of sort `struct` from the other ones.

Definition 2. *Let Σ be a signature as given in Definition 1 and let Σ_t be a signature such that Σ and Σ_t have distinct function symbols, and may share sorts, except* `struct`*. A* bridging *function $f \notin \Sigma \cup \Sigma_t$ has arity* $\texttt{struct} \to \texttt{t}$ *where* `t` *is a sort in Σ_t. A* bridging *theory T_f associated to a bridging function f is has the form:*

$$T_f = \bigcup_{c \in \Sigma} \left\{ \ \forall \boldsymbol{e} \forall t_1, \dots, t_n \ . \ f(c(\boldsymbol{e}; t_1, \dots, t_n)) = f_c(\boldsymbol{e}; f(t_1), \dots, f(t_n)) \ \right\}$$

where $f_c(\boldsymbol{x}; \boldsymbol{y})$ denotes a Σ_t-term.

Remark that the notation $f_c(\boldsymbol{x}; \boldsymbol{y})$ does not mean that all elements of $\boldsymbol{x}; \boldsymbol{y}$ must occur in the term $f_c(\boldsymbol{x}; \boldsymbol{y})$, as shown in the first case of the example below.

Example 2. (Example 1 continued). Many useful bridging theories fall into the above definition such as:

- Length of lists: $\ell(\mathit{cons}(e, y)) = 1 + \ell(y), \ \ell(\mathit{nil}) = 0$
- Sum of lists of numbers: $\mathit{lsum}(\mathit{cons}(e, y)) = e + \mathit{lsum}(y), \ \mathit{lsum}(\mathit{nil}) = 0$
- Sum of pairs of numbers: $\mathit{psum}(\mathit{pair}(e, e')) = e + e'$

3 A Combination Procedure for Bridging Functions

We introduce a combination method for a particular non-disjoint union of theories made of a source theory, a target theory, and a third one defining the bridging theory. Let T be the union of $T_s = AFDS_{\Sigma_s}$, T_t and T_f as given in Definition 2. For simplicity, we assume that T_t is stably infinite for sorts in $\Sigma_s \cap \Sigma_t$: any T_t-satisfiable set of literals is satisfiable in a model of T_t such that the domain associated to each sort in $\Sigma_s \cap \Sigma_t$ is infinite. The Nelson-Oppen combination method in its simplest presentation can then be reused. More generally, we could consider an arbitrary target theory T_t and rely on a property of the data structure theory T_s that may be stronger than stably infiniteness [15,7]. We describe below a decision procedure for checking the T-satisfiability of sets of ground literals.

First phase: Variable Abstraction and Partition. The first phase of our decision procedure takes an input set of mixed literals φ, and converts it into sets of flat (and so pure) literals. As usual, a *flat* equality is an equality $t_0 = f(t_1, \dots, t_n)$ where each term t_i is of depth 0 for $i = 0, \dots, n$ with $n \geq 0$ (a term of depth 0 is either a constant or a variable[4]); a *flat* disequality is a disequality between two terms of depth 0. The output of this phase is an equisatisfiable formula $\varphi_{struct} \cup \varphi_{elem} \cup \varphi_t \cup \varphi_f$ such that:

- φ_{struct} contains only flat literals of the following forms:
  - $x = y$, where x and y are of sort `struct`
  - $x \neq y$, where x and y are of sort `struct`
  - $x = k$, where k is an atomic constructor
  - $x = c(\boldsymbol{e}; x_1, \dots, x_n)$, where c is a non-atomic constructor
- φ_{elem} contains only flat literals of sorts in $\Sigma_s \backslash (\Sigma_t \cup \{\mathtt{struct}\})$
- φ_t contains only flat Σ_t-literals
- φ_f contains only flat literals of the form $u = f(x)$

The procedure uses flattening: it introduces fresh variables to define sub-terms in compound terms as a mean to obtain pure literals.

Second phase: Decomposition. In this phase, we make use of the following notion: an *arrangement* over a set of variable symbols S is a maximal satisfiable set of well-sorted equalities and inequalities $a = b$ or $a \neq b$, with $a, b \in S$. We build two sets of literals Γ_{struct} and Γ_t that will be necessary to maintain satisfiability: Γ_{struct} and Γ_t are initialized with the same arrangement (guessed non-deterministically) over the shared elements of sorts in $\Sigma_s \cap \Sigma_t$ occurring in both φ_{struct} and $\varphi_t \cup \varphi_f$.

Γ_{struct} will keep the information of equivalence between elements of sort `struct`. To do this, non-deterministically guess an arrangement over elements of sort `struct`, and add it to Γ_{struct}.

In Γ_t, add the collection of literals obtained by replacing all literals in $\varphi_{struct} \cup \varphi_f \cup \Gamma_{struct}$ with the following replacements:

[4] A variable can be considered as an uninterpreted constant if the satisfiability problem is viewed as a consistency problem in an expansion of the signature with fresh constants.

1. $x = y \to f_x = f_y$, where x, y are of sort `struct`
2. $u = f(x) \to u = f_x$
3. $x = k \to f_x = f_k$, where k is an atomic constructor
4. $x = c(\boldsymbol{e}; x_1, \ldots, x_n) \to f_x = f_c(\boldsymbol{e}; f_{x_1}, \ldots, f_{x_n})$, where c is a non-atomic constructor

Third phase: Check. The satisfiability check then reduces to two satisfiability check for the disjoint decision procedures, thanks to the following lemma[5]

Lemma 1. *Let $\varphi = \varphi_{struct} \cup \varphi_{elem} \cup \varphi_t \cup \varphi_f$ be a set of literals in separate form. The combination procedure described above computes Γ_{struct} and Γ_t such that φ is T-satisfiable if and only if*

- *$\varphi_{struct} \cup \varphi_{elem} \cup \Gamma_{struct}$ is T_s-satisfiable, and*
- *$\varphi_t \cup \Gamma_t$ is T_t-satisfiable.*

Example 3. Consider the theory of lists with a length function ℓ, and suppose we want to check the satisfiability of the set of literals φ:

$$\{x = cons(a, cons(b, z)),\ \ell(x) + 1 = \ell(z)\}$$

1. **Variable Abstraction and Partition.** φ will be divided into:
   - $\varphi_{list} : \{y = cons(b, z), x = cons(a, y)\}$
   - $\varphi_{elem} : \emptyset$
   - $\varphi_{\mathbb{Z}} : \{c + 1 = d\}$
   - $\varphi_{\ell} : \{\ell(x) = c, \ell(z) = d\}$, where c and d are new variables.
2. **Decomposition.** We create the variables ℓ_x, ℓ_y and ℓ_z, and the sets:
   - Γ_{list}: we need to guess an arrangement between the list variables. Let us choose the one in which they are all different, so we will add to Γ_{list} the set of literals: $\{x \neq y, y \neq z, z \neq x\}$. This is the only arrangement that is satisfiable together with φ_{list}, so it is the only choice that may lead to satisfiability.
   - $\Gamma_{\mathbb{Z}}$: after performing the replacements, we will have the set of literals $\{\ell_y = \ell_z + 1, \ell_x = \ell_y + 1, \ell_x = c, \ell_z = d\}$.
3. **Check.** The set $\varphi_{list} \cup \varphi_{elem} \cup \Gamma_{list}$ is satisfiable in the theory of lists. However $\varphi_{\mathbb{Z}} \cup \Gamma_{\mathbb{Z}}$ is unsatisfiable in the theory of linear arithmetic (over the integers). The original set of literals φ is thus unsatisfiable.

4 Conclusion

We briefly described a Nelson-Oppen like combination procedure for bridging functions. This procedure is not only restricted to absolutely free data structures (even if the current presentation only refers to this special case), but is also suitable for any theory in the spectrum between uninterpreted symbols and

[5] The non-deterministic choices in the second phase have all to be checked before concluding to unsatisfiability.

absolutely free data structures. A natural follow-up is to consider extensionality and the restriction to standard models [16,17]. For simplicity, the presentation here is non-deterministic. However, just like in the classical Nelson-Oppen scheme, implementations will be based on deterministic, and thus more practical, approaches of the same procedure.

Several powerful and successful frameworks [12,16,17] have already been provided to handle bridging functions. We believe our approach is more light-weight, and is thus more amenable to implementation inside SMT solvers, just like superposition calculi [1,4] are perfectly suited for saturation provers.

Acknowledgments: we would like to thank the reviewers for their insightful comments. These will help us to complete the works briefly described in this short version.

References

1. A. Armando, M. P. Bonacina, S. Ranise, and S. Schulz. New results on rewrite-based satisfiability procedures. *ACM Trans. Comput. Log.*, 10(1), 2009.
2. A. Armando, S. Ranise, and M. Rusinowitch. A rewriting approach to satisfiability procedures. *Inf. Comput.*, 183(2):140–164, 2003.
3. F. Baader and S. Ghilardi. Connecting many-sorted theories. *J. Symb. Log.*, 72(2):535–583, 2007.
4. P. Baumgartner and U. Waldmann. Hierarchic superposition with weak abstraction. In *Automated Deduction - CADE-24 - 24th International Conference on Automated Deduction, Lake Placid, NY, USA, June 9-14, 2013. Proceedings*, volume 7898 of *Lecture Notes in Computer Science*, pages 39–57. Springer, 2013.
5. P. Chocron. A study of the Combination Problem: dealing with multiple theories in SMT solving. Master's thesis, Universidad de Buenos Aires, Mar. 2014.
6. P. Chocron, P. Fontaine, and C. Ringeissen. A Gentle Non-Disjoint Combination of Satisfiability Procedures. In *Proc. of the 7th International Joint Conference on Automated Reasoning, IJCAR*. Springer, 2014. Extended version available as Inria Research Report, cf. `http://hal.inria.fr/hal-00985135`.
7. P. Fontaine. Combinations of theories for decidable fragments of first-order logic. In S. Ghilardi and R. Sebastiani, editors, *Frontiers of Combining Systems (FroCoS)*, volume 5749 of *LNCS*, pages 263–278. Springer, 2009.
8. P. Fontaine, S. Ranise, and C. G. Zarba. Combining lists with non-stably infinite theories. In F. Baader and A. Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'04)*, volume 3452 of *LNCS*, pages 51–66. Springer-Verlag, 2005.
9. S. Ghilardi. Model-theoretic methods in combined constraint satisfiability. *Journal of Automated Reasoning*, 33(3-4):221–249, 2004.
10. E. Kruglov and C. Weidenbach. Superposition decides the first-order logic fragment over ground theories. *Mathematics in Computer Science*, 6(4):427–456, 2012.
11. G. Nelson and D. C. Oppen. Simplification by cooperating decision procedures. *ACM Trans. on Programming Languages and Systems*, 1(2):245–257, Oct. 1979.
12. E. Nicolini, C. Ringeissen, and M. Rusinowitch. Combinable extensions of Abelian groups. In R. A. Schmidt, editor, *Proc. Conference on Automated Deduction (CADE)*, volume 5663 of *LNCS*, pages 51–66. Springer, 2009.

13. E. Nicolini, C. Ringeissen, and M. Rusinowitch. Combining satisfiability procedures for unions of theories with a shared counting operator. *Fundam. Inform.*, 105(1-2):163–187, 2010.
14. T.-H. Pham and M. W. Whalen. An improved unrolling-based decision procedure for algebraic data types. In E. Cohen and A. Rybalchenko, editors, *Verified Software: Theories, Tools, Experiments - 5th International Conference, VSTTE 2013, Menlo Park, CA, USA, May 17-19, 2013, Revised Selected Papers*, volume 8164 of *Lecture Notes in Computer Science*, pages 129–148. Springer, 2014.
15. S. Ranise, C. Ringeissen, and C. G. Zarba. Combining data structures with nonstably infinite theories using many-sorted logic. In B. Gramlich, editor, *Frontiers of Combining Systems (FroCoS)*, volume 3717 of *LNCS*, pages 48–64. Springer, 2005.
16. V. Sofronie-Stokkermans. Locality results for certain extensions of theories with bridging functions. In R. A. Schmidt, editor, *Proc. Conference on Automated Deduction (CADE)*, volume 5663 of *LNCS*, pages 67–83. Springer, 2009.
17. P. Suter, M. Dotta, and V. Kuncak. Decision procedures for algebraic data types with abstractions. In M. V. Hermenegildo and J. Palsberg, editors, *Principles of Programming Languages (POPL)*, pages 199–210. ACM, 2010.
18. P. Suter, A. S. Köksal, and V. Kuncak. Satisfiability modulo recursive programs. In E. Yahav, editor, *Static Analysis - 18th International Symposium, SAS 2011, Venice, Italy, September 14-16, 2011. Proceedings*, volume 6887 of *Lecture Notes in Computer Science*, pages 298–315. Springer, 2011.
19. C. Tinelli and M. T. Harandi. A new correctness proof of the Nelson–Oppen combination procedure. In F. Baader and K. U. Schulz, editors, *Frontiers of Combining Systems (FroCoS)*, Applied Logic, pages 103–120. Kluwer Academic Publishers, Mar. 1996.
20. C. Tinelli and C. Ringeissen. Unions of non-disjoint theories and combinations of satisfiability procedures. *Theoretical Comput. Sci.*, 290(1):291–353, Jan. 2003.
21. T. Wies, R. Piskac, and V. Kuncak. Combining theories with shared set operations. In S. Ghilardi and R. Sebastiani, editors, *Frontiers of Combining Systems (FroCoS)*, volume 5749 of *LNCS*, pages 366–382. Springer, 2009.
22. C. Zarba. Combining multisets with integers. In A. Voronkov, editor, *Automated DeductionCADE-18*, volume 2392 of *Lecture Notes in Computer Science*, pages 363–376. Springer Berlin Heidelberg, 2002.
23. C. G. Zarba. Combining lists with integers. In *International Joint Conference on Automated Reasoning (Short Papers), Technical Report DII 11/01*, pages 170–179. University of, 2001.
24. C. G. Zarba. Combining sets with cardinals. *J. Autom. Reasoning*, 34(1):1–29, 2005.
25. T. Zhang, H. B. Sipma, and Z. Manna. Decision procedures for term algebras with integer constraints. *Inf. Comput.*, 204(10):1526–1574, 2006.