Satisfiability Modulo Free Data Structures Combined with Bridging Functions

Raphaël Berthon[1], Christophe Ringeissen[2]

[1]ENS Rennes

[2]LORIA & Inria Nancy Grand Est

SMT 2016

Outline

1. Introduction
2. Free Data Structures with Bridging Functions
3. Combination Method
4. Conclusion

Outline

1 Introduction

2 Free Data Structures with Bridging Functions

3 Combination Method

4 Conclusion

Satisfiability Procedures in Deductive Verification

- verification tools need satisfiability procedures to reason modulo (the combination of) theories
  - little engines of Satisfiability Modulo Theories (SMT) solvers
- Recurrent task: solve satisfiability problems expressed in a combination of
  - fragments of Arithmetic
  - theories modeling data-structures: list, tree, set, multiset, array, record, UF, ...
  - bridging functions: length, size, cardinality, ...
  - data structures instantiated with arbitrary theories of elements: list[tree], array[int], list[bool], tree[bool], ...

Combining Satisfiability Procedures

- Nelson-Oppen combination method ubiquitous in SMT solvers
- classical limitations: the component theories are signature-disjoint and stably infinite
  - bridging functions: a form of non-disjoint combination

$$T_\ell = \left\{ \begin{array}{lcl} \ell(\mathit{cons}(X, Y)) & = & \ell(Y) + 1 \\ \ell(\mathit{nil}) & = & 0 \end{array} \right.$$

  - combination with a theory of finite elements

$$(\forall X : \mathtt{elem}.\ X = a \vee X = b) \wedge a \neq b$$

➽ *polite* data structures theories are combinable with arbitrary theories of elements [Ranise et al., 2005]

This talk: non-disjoint combinations with bridging functions

Non-disjoint Combinations with Bridging Functions

- Absolutely Free Data structures (AFDS) combined via bridging functions
  - Lists with length [Fontaine et al., 2005]
  - Term algebras with integer constraints [Zhang et al., 2006]
  - AFDS with bridging function [Sofronie-Stokkermans, 2009]
  - Trees with abstraction function [Suter et al., 2010]
  - AFDS with bridging function [Chocron et al., 2015]
    ➸ A combination approach à la Nelson-Oppen for polite theories
- Free Data Structures (FDS) combined via bridging functions
  ➸ Trees/Lists/(Multi)Sets with cardinality [Zarba, 2005]

This talk: A combination approach à la Nelson-Oppen for FDS?

Outline

1 Introduction

2 Free Data Structures with Bridging Functions

3 Combination Method

4 Conclusion

Free Data Structures (Finite Trees Modulo)

$$\Sigma = \left\{ \begin{array}{l} c : \texttt{struct} \times \texttt{struct} \rightarrow \texttt{struct} \\ u : \texttt{elem} \rightarrow \texttt{struct} \\ \mathit{nil} : \texttt{struct} \end{array} \right\}$$

$$\mathit{FDS}_E = \{\mathcal{A} \mid \texttt{struct}_{\mathcal{A}} = T(\Sigma \cup \texttt{elem}_{\mathcal{A}})/=_E\}$$

Remark

E can be combinations of regular axioms including Associativity, Commutativity, Unit, Idempotency.

Examples of Useful Theories

Examples of FDS instances: (Standard interpretation of) Sets, Multisets, Lists.

Example

$$E_{Multiset} = AC(\uplus) \cup \{\forall X.\, X \uplus \varnothing = X\}$$

$$E_{Set} = AC(\cup) \cup \{\forall X.\, X \cup \emptyset = X, X \cup X = X\}$$

$$E_{List} = A(c) \cup \{\forall X.\, c(X, nil) = X, c(nil, X) = X\}$$

Surprising Examples

Examples of more surprising FDS instances: (Standard interpretation of) Naturals and Booleans.

Example

$$E_{\mathbb{N}} = AC(+) \cup \{\forall X.\, X + 0 = X\} \cup \{\forall V, W.\, 1(V) = 1(W)\}$$

$$E_{\mathbb{B}} = AC(\vee) \cup \{\forall X.\, X \vee \bot = X, X \vee X = X\} \cup \{\forall V, W.\, \top(V) = \top(W)\}$$

Bridging Theory

$f : FDS_{E_1} \to FDS_{E_2}$ defined by structural induction

Definition

Given the signature $\Sigma_f = \Sigma_1 \cup \Sigma_2 \cup \{f : \mathtt{struct}_1 \to \mathtt{struct}_2\}$, a *bridging* Σ_f-theory T_f associated to f has the form:

$$T_f = \begin{cases} f(c(X,Y)) & = & f_c(f(X), f(Y)) \\ f(u(V)) & = & f_u(V) \\ f(nil) & = & f_{nil} \end{cases}$$

where $f_c, , f_u, f_{nil}$ are Σ_t-terms of respective arities $2, 1, 0$.

Assumption

f maps E_1-equal terms to E_2-equal terms

Bridging Theory: Examples

Example

The function calculating the set of the elements of a tree:

$$T_f = \begin{cases} f(nil) = \emptyset \\ f(u(x)) = \{x\} \\ f(c(X_1, X_2)) = f(X_1) \cup f(X_2) \end{cases}$$

Example

The function calculating the multiset of the elements of a tree:

$$T_f = \begin{cases} f(nil) = \varnothing \\ f(u(x)) = \langle x \rangle \\ f(c(X_1, X_2)) = f(X_1) \uplus f(X_2) \end{cases}$$

Combination of Theories

Definition

Let $T_1 = FDS_{E_1}$ and $T_2 = FDS_{E_2}$.
T denotes the class of Σ_f-structures $\mathcal{A}$ such that

- $\mathcal{A}^{\Sigma_1} \in T_1$,
- $\mathcal{A}^{\Sigma_2} \in T_2$,
- and $\mathcal{A} \models T_f$.

in other words, $T = T_1 \cup T_f \cup T_2$.

Problem

Develop a combination method for T-satisfiability

Our Approach: Reduction to Disjoint Combination

How to combine satisfiability procedures for theories connected by a bridging theory?

1. Instantiate the axioms of the bridging theory to get ground equalities
2. Hence, the problem is reduced to a disjoint combination: reuse of Nelson-Oppen procedure

Outline

1 Introduction

2 Free Data Structures with Bridging Functions

3 Combination Method

4 Conclusion

Overview of the Combination Method

1. **Purification:** Convert the input into a separate form $\varphi_1 \cup \varphi_{elem} \cup \varphi_2 \cup \varphi_f$, where φ_1 is flat and solved, $\varphi_f = \bigcup_x \{f_x = f(x)\}$
2. **Target Encoding:** Consider $\varphi_1 \cup \varphi_{elem} \cup \varphi_2 \cup CP_{\varphi_1} \cup \varphi_f$ where

$$\begin{aligned} CP_{\varphi_1} \quad = \quad & \{f_v = f_c(f_x, f_y) \mid v = c(x, y) \in \varphi_1\} \\ & \cup \{f_v = f_u(e) \mid v = u(e) \in \varphi_1\} \\ & \cup \{f_v = f_{nil} \mid v = nil \in \varphi_1\} \\ & \cup \{f_v = f_x \mid v = x \in \varphi_1\} \end{aligned}$$

3. **Guessing** range constraints for bridging variables ($f_x = i$ or $f_x \geq i$)
4. **Inverting** range constraints ($f_x = i \Leftrightarrow x = t_i$)
5. **Checking** satisfiability in component theories

Combination Method: An Example

Assume $f : \mathit{List} \to \mathit{Set}$ computes the set of elements in a list.

Consider the separate form

$$\varphi = \begin{cases} X = (e) \cdot Y, V = (e), X \neq V \\ f_X = \{e\}, f_X \neq f_Y \\ f_X = f(X), f_Y = f(Y), f_V = f(V) \end{cases}$$

Let us guess the following values for f_Y: $f_Y = \emptyset$ or $f_Y = \{e'\} \cup S$.

- The non-empty case is UNSAT in *Set* since $f_Y = \emptyset$ is entailed by $\{e\} \cup f_Y = f_X = \{e\}, f_X \neq f_Y$.
- The case $f_Y = \emptyset$ is SAT in *Set*. Then, we add $Y = \mathit{nil}$ and this leads to $X = (e) \cdot \mathit{nil} = (e) = V$, which is UNSAT in *List*.

Therefore, φ is UNSAT.

Case Studies

- A complete procedure (without any guessing for bridging variables), when there is no source disequalities
- A form of infinite surjectivity, when considering a counting function as bridging function:
  $T_f = \{f(c(X, Y)) = f(X) + f(Y), f(u(e)) = 1, f(nil) = 0\}$
- A form of sufficient surjectivity, when there are only disequalities between *isolated* variables

Infinite Surjectivity

Consider the counting function

$$T_f = \{f(c(X, Y)) = f(X) + f(Y), f(u(e)) = 1, f(nil) = 0\}$$

There are only two values to consider in the guessing:

- $f_x = 0$, equivalently the standard term x is empty
- $f_x > 0$, equivalently the standard term x is non-empty, and there are enough distinct elements so that x will be distinct to any other variable in the problem.

Why it works

A model of T_1 with a **infinite** interpretation domain for `elem` can be used to build a model of T.

Sufficient Surjectivity

Consider a notion of *measure* τ to order the standard terms s of the target FDS

Assumption: for any input, existence of a threshold k such that

1. If $\tau(s) \geq k$, then $|f^{-1}(s)|$ is large enough
2. If $\tau(s) < k$, then possibility to invert f:

$$f(x) = s \iff x = t_s$$

where both conditions are expressed as constraints in the target FDS

Related work

In [Chocron et al., 2015]	$f : AFDS \longrightarrow Int$
This talk	$f : FDS_{E_1} \longrightarrow FDS_{E_2}$

Outline

1 Introduction

2 Free Data Structures with Bridging Functions

3 Combination Method

4 Conclusion

Conclusion and Future Work

A combination method with a finite guessing phase, which is complete in some *simple* cases where source disequalities are easy to satisfy

Requirement: unification/matching algorithms to compute source solved forms

Next future:

- Find more general decidable inputs including source disequalities
- Combine T with a stably infinite theory of elements T_{elem}: classical NO
- Combine T with an arbitrary theory of elements T_{elem}: polite NO (if T is polite)


Chocron, P., Fontaine, P., and Ringeissen, C. (2015).
A Polite Non-Disjoint Combination Method: Theories with Bridging Functions Revisited.
In Felty, A. P. and Middeldorp, A., editors, *Proc. Conference on Automated Deduction (CADE)*, volume 9195 of *LNCS*, pages 419–433. Springer.


Fontaine, P., Ranise, S., and Zarba, C. G. (2005).
Combining lists with non-stably infinite theories.
In Baader, F. and Voronkov, A., editors, *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'04)*, volume 3452 of *LNCS*, pages 51–66. Springer-Verlag.


Nicolini, E., Ringeissen, C., and Rusinowitch, M. (2009).
Combinable extensions of Abelian groups.
In Schmidt, R. A., editor, *Proc. Conference on Automated Deduction (CADE)*, volume 5663 of *LNCS*, pages 51–66. Springer.


Ranise, S., Ringeissen, C., and Zarba, C. G. (2005).
Combining data structures with nonstably infinite theories using many-sorted logic.
In Gramlich, B., editor, *Frontiers of Combining Systems (FroCoS)*, volume 3717 of *LNCS*, pages 48–64. Springer.


Sofronie-Stokkermans, V. (2009).
Locality results for certain extensions of theories with bridging functions.
In Schmidt, R. A., editor, *Proc. Conference on Automated Deduction (CADE)*, volume 5663 of *LNCS*, pages 67–83. Springer.


Suter, P., Dotta, M., and Kuncak, V. (2010).
Decision procedures for algebraic data types with abstractions.

In Hermenegildo, M. V. and Palsberg, J., editors, *Principles of Programming Languages (POPL)*, pages 199–210. ACM.

Zarba, C. G. (2005).
Combining sets with cardinals.
J. Autom. Reasoning, 34(1):1–29.

Zhang, T., Sipma, H. B., and Manna, Z. (2006).
Decision procedures for term algebras with integer constraints.
Inf. Comput., 204(10):1526–1574.