On Combining Theories with Shared Set Operations

Thomas Wies, Ruzica Piskac, and Viktor Kuncak

EPFL School of Computer and Communication Sciences, Switzerland

Abstract. We explore the problem of automated reasoning about the non-disjoint combination of theories that share set variables and operations. We prove a combination theorem and apply it to show the decidability of the satisfiability problem for a class of formulas obtained by applying propositional operations to quantified formulas belonging to several expressive decidable logics.

1 Introduction

Automated abstraction techniques such as predicate abstraction are among the most promising approaches for verifying systems with large state spaces [14, 15]. Such techniques were enabled by the recent progress in SAT and SMT solvers [2, 6, 10, 24]. The range of verification problems that are amenable to such approaches depends on the expressive power of the logics supported by the SMT solvers. Current SMT solvers implement the disjoint combination of quantifier-free theories, in essence following the approach pioneered by Nelson and Oppen [26]. Such solvers serve as decision procedures for quantifier-free formulas, typically containing uninterpreted function symbols, linear arithmetic, and bitvectors. The limited expressiveness of SMT prover logics translates into a limited class of properties that automated verification tools can handle.

To enable broader applications of automated verification techniques, this paper considers decision procedures for the combination of quantified formulas in non-disjoint theories. The idea of combining rich theories within an expressive language has been explored in interactive provers [3, 5, 25, 27]. Such integration efforts are very useful, but do not result in complete decision procedures for the combined logics. The study of completeness for non-disjoint combination is relatively recent [34, 37] and provided foundation for the general problem. Our paper considers a particular combination of non-disjoint theories–theories sharing operations on *sets of uninterpreted elements*. To the best of our knowledge, this problem has not been considered before, despite the usefulness of sets for reasoning about dynamically created objects and concurrent processes.

Challenges in communicating constraints on sets. The idea of combining decision procedures is to check the satisfiability of a conjunction of formulas $A \wedge B$ by using one decision procedure, D_A, for A and another decision procedure, D_B, for B. To obtain a complete decision procedure, D_A and D_B must communicate to ensure that a model found by D_A and a model found by D_B can be merged into a model for $A \wedge B$. Craig's interpolation theorem for first-order logic implies that if $A \wedge B$ is unsatisfiable, then there exists an interpolant I such that 1) $A \rightarrow I$ is valid 2) $I \wedge B$ is unsatisfiable, and 3) I is a (potentially quantified) first-order formula containing only predicate symbols and variables common to A and B. The interpolant I can be used to communicate the information between D_A and D_B. When A and B belong to disjoint theories, I contains equalities as the only kind of atomic formulas. The class of such formulas admits quantifier elimination, so there are only finitely many non-equivalent formulas I, making it easier to construct a complete combined decision procedure.

The combination problem is more difficult for formulas that share sets of elements, because there are infinitely many constraints on sets definable in typical logics. For example, for every non-negative integer K, most logics can express the property that a shared set has exactly K elements. The set of possible interpolants I is thus not bounded by the number of symbols shared between A and B, but depends also on the structure of formulas A and B.

Decision procedure based on projections. In this situation we suggest that D_A computes the projection S_A of A onto shared set variables. This projection is equivalent to existentially quantifying over predicates and variables appearing in A but not in B, and corresponds to the strongest interpolant. D_B can similarly compute the projection S_B of B. This reduces the problem to checking the satisfiability of $A \wedge B$ to satisfiability of a formula with sets $S_A \wedge S_B$. (Alternatively, D_B could attempt to directly check $S_A \wedge B$.)

A logic for shared constraints on sets. The logic of sets used to express the projections S_A and S_B is a key parameter of such a combination approach, and depends on the logics of formulas A, B. Inspired by verification of linked data structures, we consider as the logics for A, B weak monadic second-order logic of two successors WS2S [33], two-variable logic with counting C^2 [13, 28, 32], Bernayes-Schönfinkel class of first-order logic [4], and BAPA [21]. Remarkably, in many of these cases, *the smallest logic needed to express the projection formulas has the expressive power of Boolean Algebra with Presburger Arithmetic (BAPA)* [22], Figure 4. We also show that the decision procedures for these logics can be naturally extended to compute a reduction to BAPA that preserves constraints on set variables. The existence of these reductions, along with good properties of BAPA (quantifier elimination, NP membership for quantifier-free fragment [22]) make it an ideal candidate as a common reduction target for expressive logics that share sets.

Contributions. We present a simple technique for showing decidability of theories that share sets of elements, and show that the logics

1. Boolean Algebra with Presburger Arithmetic itself [7, 21, 22]
2. weak monadic second-order logic of two successors WS2S [33];
3. two-variable logic with counting C^2 [32];
4. Bernays-Schönfinkel class [4]
5. quantifier-free multisets with cardinality constraints [31]

meet the conditions of the technique, which allows the use of their combination in verification. (An earlier version of these results was presented in [23].)

2 Example: Verifying a Code Fragment

Our example illustrates a formula arising from verifying unbounded linked data structures, and explains why our combination technique is complete for proving the validity

```
class Node {Node left,right; Object data;}
class Tree {
   private static Node root;
   private static int size; /*:
   private static specvar nodes :: objset;
   vardefs "nodes=={x. (root,x) ∈ {(x,y). left x = y ∨ right x = y}*}";
   private static specvar content :: objset;
   vardefs "content=={x. ∃ n. n ≠ null ∧ n ∈ nodes ∧ data n = x} " */

   private void insertAt(Node p, Object e) /*:
     requires "tree [ left , right ] ∧ nodes ⊆ Object.alloc ∧ size = card content ∧
               e ∉ content ∧ e ≠ null ∧ p ∈ nodes ∧ p ≠ null ∧ left p = null"
     modifies nodes,content,left, right, data,size
     ensures "size = card content"  */
   {
      Node tmp = new Node();
      tmp.data = e;
      p. left  = tmp;
      size = size + 1;
   }
}
```

Fig. 1. Fragment of insertion into a tree

of an interesting class of such formulas. Figure 1 shows our example, which is a fragment of Java code for insertion into a binary search tree, factored out into a separate `insertAt` method (we also verified the full code, containing loops). The search tree has fields (`left`, `right`) that form a tree, and field `data`, which is not necessarily an injective function (an element may be stored multiple times in the tree). The `insertAt` method is meant to be invoked when the insertion procedure has found a node `p` that has no left child. It inserts the given object `e` into a fresh node `tmp` that becomes the new left child of `p`.

Specification and verification in Jahob. In addition to Java statements, the example in Fig. 1 contains preconditions and postconditions, written in the notation of the Jahob verification system [20, 36, 38]. The *vardefs* notation introduces two sets: 1) the set of auxiliary objects *nodes*, denoting the `Node` objects stored in the binary tree, and 2) the set *content* denoting the useful content of the tree. To verify such examples in the previously reported approach [38], the user of the system had to manually provide the definitions of such sets, and to manually introduce certain lemmas describing changes to these sets. Our decidability result means that there is no need to manually introduce such lemmas.

Decidability of the verification condition. Figure 2 shows the verification condition formula for method `insertAt`. The validity of this formula implies that invoking a method in a state satisfying the precondition results in a state that satisfies the postcondition of `insertAt`. The formula contains the transitive closure operator, quantifiers, set comprehensions, and the cardinality operator. Nevertheless, there is a (syntactically defined) decidable class of formulas that contains the verification condition in Fig. 2.

```
tree[ left , right ] ∧ left p = null ∧ p ∈ nodes ∧
nodes={x. (root,x) ∈ {(x,y). left x = y| right x = y}ˆ∗} ∧
content={x. ∃ n. n ≠ null ∧ n ∈ nodes ∧ data n = x} ∧
e ∉ content ∧ nodes ⊆ alloc ∧
tmp ∉ alloc ∧ left tmp = null ∧ right tmp = null ∧
data tmp = null ∧ (∀ y. data y ≠ tmp) ∧
nodes1={x. (root,x) ∈ {(x,y). ( left (p:=tmp)) x = y) | right x = y} ∧
content1={x. ∃ n. n ≠ null ∧ n ∈ nodes1 ∧ (data(tmp:=e)) n = x} −→
             card content1 = card content + 1
```

Fig. 2. Verification condition for Fig. 1

```
SHARED SETS: nodes, nodes1, content, content1, {e}, {tmp}

WS2S FRAGMENT:
 tree[ left , right ] ∧ left p = null ∧ p ∈ nodes ∧ left tmp = null ∧ right tmp = null ∧
 nodes={x. (root,x) ∈ {(x,y). left x = y| right x = y}ˆ∗} ∧
 nodes1={x. (root,x) ∈ {(x,y). ( left (p:=tmp)) x = y) | right x = y}
CONSEQUENCE: nodes1=nodes ∪ {tmp}

C2 FRAGMENT:
  data tmp = null ∧ (∀ y. data y ≠ tmp) ∧ tmp ∉ alloc ∧ nodes ⊆ alloc ∧
  content={x. ∃ n. n ≠ null ∧ n ∈ nodes ∧ data n = x} ∧
  content1={x. ∃ n. n ≠ null ∧ n ∈ nodes1 ∧ (data(tmp:=e)) n = x}
CONSEQUENCE: nodes1 ≠ nodes ∪ {tmp} ∨ content1 = content ∪ {e}

BAPA FRAGMENT: e ∉ content ∧ card content1 ≠ card content + 1
CONSEQUENCE: e ∉ content ∧ card content1 ≠ card content + 1
```

Fig. 3. Separated conjuncts for negation of Fig. 2, with consequences about shared sets

This decidable class is a set-sharing combination of three decidable logics, and can be decided using the method we present in this paper.

To understand the method for proving the formula in Fig. 2, consider the problem of showing the unsatisfiability of the negation of the formula. Figure 3 shows the conjuncts of the negation, grouped according to three decidable logics to which the conjuncts belong: 1) weak monadic second-order logic of two successors WS2S [33], 2) two-variable logic with counting C^2 [32], and 3) Boolean Algebra with Presburger Arithmetic (BAPA) [7, 21, 22]. For the formula in each of the fragments, Fig. 3 also shows a consequence formula that contains only shared sets and statements about their cardinalities. (We represent elements as singleton sets, so we admit formulas sharing elements as well. Cardinality constraints appear already with sets of elements and we believe our approach could be combined with [19].)

A decision procedure. Note that the conjunction of the consequences of three formula fragments is an unsatisfiable formula. This shows that the original verification condition is valid. In general, our decidability result shows that the decision procedures of logics such as WS2S and C^2 can be naturally extended to compute strongest consequences of formulas involving given shared sets. These consequences are expressed in BAPA, which is decidable. One possible decision procedure for satisfiability of combined formulas is 1) split the formula into fragments (belonging to WS2S, C^2, or BAPA); 2) for each fragment compute its strongest QFBAPA consequence; 3) check the satisfiability of the conjunction of consequences.

3 Syntax and Semantics of Formulas

Multi-sorted logic. We present our problem in multi-sorted logic with equality and disjoint sorts (which can naturally be viewed as a fragment of higher-order logic [1] with a particular set of types, which we call sorts). The primitive sorts we consider include: 1) bool, interpreted as the two-element set $\{\mathsf{true}, \mathsf{false}\}$ of booleans; 2) int, interpreted as the set of integers $\mathbb{Z}$; and 3) obj, interpreted as a non-empty set of elements. Each variable and constant has an associated sort. We represent a function mapping elements of sorts $s_1, \ldots, s_n$ into an element of sort s_0 as a term of sort $s_1 \times \ldots \times s_n \to s_0$. When $s_1, \ldots, s_n$ are all the same sort s, we denote $s_1 \times \ldots \times s_n$ as s^n. We represent a relation between elements of sorts $s_1, \ldots, s_n$ as a function $s_1 \times \ldots \times s_n \to \mathsf{bool}$. We use set as an abbreviation for the sort $\mathsf{obj} \to \mathsf{bool}$. In the formulas that we consider, all quantified variables have one of the sorts int, obj, set. Free variables can additionaly have sorts $\mathsf{obj}^n \to \mathsf{bool}$ for $n > 1$. Propositional operations connect terms of sort bool. We write $\forall x : s.F$ to denote a universally quantified formula where the quantified variable has sort s (analogously for $\exists x : s.F$). We denote by $\mathsf{FV}(F)$ the set of all variables that occur free in F and we write $\mathsf{FV}_s(F)$ for the free variables of sort s. The equality symbol applies only to terms of the same sort. We can assume to have a distinct equality symbol for each sort of interest, but we use the same symbol to denote all of them.

Structures and semantics. A structure α specifies a finite set, which is also the meaning of obj, and we denote it $\alpha(\mathsf{obj})$. We focus on the case of finite $\alpha(\mathsf{obj})$ primarily for simplicity of notation; extension to the case where domains are either finite or countable is possible and can be done using results from [21, Section 8.1], [32, Section 5], [33]. Each structure α that interprets a set of formulas is determined by $\alpha(\mathsf{obj})$ as well as the values $\alpha(x)$ for each variable x free in the set of formulas, where the sort of x is among obj and $\mathsf{obj}^n \to \mathsf{bool}$ for $n \geq 1$. When α is understood we use $[\![X]\!]$ to denote $\alpha(X)$, where X denotes a sort, a term, a formula, or a set of formulas. If S is a set of formulas then $\alpha(S) = \mathsf{true}$ means $\alpha(F) = \mathsf{true}$ for each $F \in X$. In every structure we let $[\![\mathsf{bool}]\!] = \{\mathsf{false}, \mathsf{true}\}$. Instead of $\alpha(F) = \mathsf{true}$ we often write simply $\alpha(F)$. We interpret terms of the sort $s_1 \times \ldots \times s_n \to s_0$ as total functions $[\![s_1]\!] \times \ldots [\![s_n]\!] \to [\![s_0]\!]$ and identify a function $f : A \to \{\mathsf{false}, \mathsf{true}\}$ with the set $\{x \mid f(x) = \mathsf{true}\}$. We thus interpret variables of the sort $\mathsf{obj}^n \to \mathsf{bool}$ as subsets of $[\![\mathsf{obj}]\!]^n$. We interpret propositional operations $\wedge, \vee, \neg$ as usual in classical logic. A quantified variable of sort s ranges over all elements of $[\![s]\!]$.

3.1 Boolean Algebra with Presburger Arithmetic

Figure 4 shows the syntax of Boolean Algebra with Presburger Arithmetic (BAPA) [7, 21]. The following are the sorts of constants appearing in BAPA formulas: $\subseteq$: $\mathsf{set}^2 \to \mathsf{bool}$, $<$: $\mathsf{int}^2 \to \mathsf{bool}$, dvd_K : $\mathsf{int} \to \mathsf{bool}$ for each integer constant K (with $\mathsf{dvd}_K(t)$ denoted by $K \,\mathsf{dvd}\, t$), $\emptyset, \mathsf{Univ}$: set, $\mathsf{singleton}$: $\mathsf{obj} \to \mathsf{set}$ (with $\mathsf{singleton}(x)$ denoted as $\{x\}$), $\cap, \cup$: $\mathsf{set}^2 \to \mathsf{set}$, $\mathsf{complement}$: $\mathsf{set} \to \mathsf{set}$ (with $\mathsf{complement}(A)$

$$\begin{aligned}
F &::= A \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid \neg F \mid \forall x{:}s.F \mid \exists x{:}s.F\\
s &::= \mathsf{int} \mid \mathsf{obj} \mid \mathsf{set}\\
A &::= B_1 = B_2 \mid B_1 \subseteq B_2 \mid T_1 = T_2 \mid T_1 < T_2 \mid K \,\mathsf{dvd}\, T\\
B &::= x \mid \emptyset \mid \mathsf{Univ} \mid \{x\} \mid B_1 \cup B_2 \mid B_1 \cap B_2 \mid B^c\\
T &::= x \mid K \mid \mathsf{CardUniv} \mid T_1 + T_2 \mid K \cdot T \mid \ \mathsf{card}(B)\\
K &::= \ldots -2 \mid -1 \mid 0 \mid 1 \mid 2 \ldots
\end{aligned}$$

Fig. 4. Boolean Algebra with Presburger Arithmetic (BAPA)

denoted by A^c), K : int for each integer constant K, CardUniv : int, $+$: int$^2 \rightarrow$ int, mul$_K$: int $\rightarrow$ int for each integer constant K (with mul$_K(t)$ denoted by $K \cdot t$), and card : set $\rightarrow$ int.

Let $\mathcal{F}_{\mathsf{BAPA}}$ be the set of all formulas in Figure 4. We interpret the formulas over structures with an arbitrary finite set $[\![\mathsf{obj}]\!]$. In each structure, $[\![\mathsf{set}]\!]$ is the set of all subsets of $[\![\mathsf{obj}]\!]$. The symbol Univ denotes the universal set, that is, $[\![\mathsf{Univ}]\!] = [\![\mathsf{obj}]\!]$. card$(A)$ denotes the cardinality of the set A. CardUniv is interpreted as card(Univ). The formula $K \,\mathsf{dvd}\, t$ denotes that the integer constant K divides the integer t. We note that the condition $x \in A$ can be written down in this language as $\{x\} \subseteq A$. In the rest of this paper we only consider structures that interpret the BAPA operations as defined above. [1]

A *semilinear set* is a finite union of *linear sets*. A linear set is a set of the form $\{\boldsymbol{a} + k_1\boldsymbol{b}_1 + \ldots + k_n\boldsymbol{b}_n \mid k_1, \ldots, k_n \in \{0, 1, 2 \ldots\}\}$ where $\boldsymbol{a}, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{Z}^M$. We represent a linear set by its generating vectors $\boldsymbol{a}, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$, and a semilinear set by the finite set of representations of its linear sets. It was shown in [12] that a set of integer vectors $S \subseteq \mathbb{Z}^M$ is a solution set of a Presburger arithmetic formula P i.e. $S = \{(v_1, \ldots, v_n).P\}$ iff S is a semilinear set.

We have the following characterization of relationships between sets expressible in BAPA, which follows from [21].

Lemma 1 (BAPA-expressible means Venn-cardinality-semilinear). *Given a finite set U and a relation $\rho \subseteq (2^U)^p$ the following are equivalent:*

1. *there exists a BAPA formula F whose free variables are $A_1, \ldots, A_p$ such that*

$$\rho = \{(s_1, \ldots, s_p) \mid \{A_1 \mapsto s_1, \ldots, A_p \mapsto s_p\}(F)\}$$

2. *the following subset of $\mathbb{Z}^M$ for $M = 2^p$ is semilinear:*

$$\{(|s_1^c \cap s_2^c \cap \ldots \cap s_p^c|, |s_1 \cap s_2^c \cap \ldots \cap s_p^c|, \ldots, |s_1 \cap s_2 \cap \ldots \cap s_p|) \mid (s_1, \ldots, s_p) \in \rho\}.$$

[1] We note that BAPA properly extends the first-order theory of Boolean Algebras over finite structures, which in turn subsumes the first-order logic with unary predicates and no function symbols, because e.g. $\exists x.F(x)$ can be simulated by $\exists X.\mathsf{card}(X){=}1 \wedge F'(X)$ where in F' e.g. $P(x)$ is replaced by $X \subseteq P$.

4 Combination by Reduction to BAPA

4.1 The Satisfiability Problem

We are interested in algorithms for the satisfiability problem of quantifier-free combinations of quantified formulas that share sets of elements. More precisely, we are interested in an algorithm to determine whether there exists a structure $\alpha \in \mathcal{M}$ in which the following formula is true

$$B(F_1, \ldots, F_n) \tag{1}$$

where

1. $F_1, \ldots, F_n$ are formulas with $\mathsf{FV}(F_i) \subseteq \{A_1, \ldots, A_p, R_1, \ldots, R_q\}$
2. $V_S = \{A_1, \ldots, A_p\}$ are variables of sort set, whereas $R_1, \ldots, R_q$ are the remaining variables, of sorts $\mathsf{obj}^n \to \mathsf{bool}$ for $n > 1$ (for notational simplicity we do not consider variables of sort obj because they can be represented as singleton sets, of sort set);
3. each formula F_i belongs to a set of formulas $\mathcal{F}_i$ (in our case, one of: BAPA, monadic second order logic over finite trees, two-variable logic with counting, or the Bernayes-Schönfinkel class). For each $\mathcal{F}_i$ there is the corresponding theory $\mathcal{T}_i$;
4. $B(F_1, \ldots, F_n)$ denotes a formula built from $F_1, \ldots, F_n$ using the propositional operations $\wedge, \vee$ (the absence of negation is usually not a loss of generality because in allmost all cases the $\mathcal{F}_i$ are closed under negation so B is the negation-normal form of an arbitrary quantifier-free combination);
5. as the set of structures $\mathcal{M}$ we consider all finite multisorted models (Section 3) satisfying the union of theories $\cup_{i=1}^{n} \mathcal{T}_i$;
6. (set sharing condition) if $i \neq j$ and a variable or constant x occurs both in $\{F_i\} \cup \mathcal{T}_i$ and $\{F_j\} \cup \mathcal{T}_j$, then either $x \in V_S$ (a shared set variable) or x is a constant of BAPA (Section 3.1), such as $\subseteq$.

The set sharing condition in our case means that the formulas we consider from different theories do not share any relation variables.

4.2 Combination Theorem

The formula B in (1) is satisfiable iff one of the disjuncts in its disjunctive normal form is satisfiable. Consider a disjunct $F_1 \wedge \ldots \wedge F_m$ for $m \leq n$. By definition of the satisfiability problem 1, $F_1 \wedge \ldots \wedge F_m$ is satisfiable iff there exists a structure α such that for each $1 \leq i \leq m$, for each $G \in \{F_i\} \cup \mathcal{T}_i$, we have $\alpha(G) = \mathsf{true}$. Because α is given by the domain and the interpretation of its variables, let $\{\mathsf{obj} \mapsto u, x_1 \mapsto v_1, \ldots, x_n \mapsto v_n\}$ denote the structure α with domain u that interprets each variable x_i as v_i. Let each relation variable R_i have the sort $\mathsf{obj}^{k_i} \to \mathsf{bool}$. Then the satisfiability of $F_1 \wedge \ldots \wedge F_m$ is equivalent to the following condition:

$$\begin{array}{l}\exists \text{ finite set } u.\ \exists a_1, \ldots, a_p \subseteq u.\ \exists r_1 \subseteq u^{k_1}. \ldots\ \exists r_q \subseteq u^{k_q}.\ \bigwedge_{i=1}^{m} \\ \{\mathsf{obj} \to u, A_1 \mapsto a_1, \ldots, A_p \mapsto a_p, R_1 \mapsto r_1, \ldots, R_q \mapsto r_q\}(\{F_i\} \cup \mathcal{T}_i)\end{array} \tag{2}$$

By the set sharing condition, each of the variables $r_1, \ldots, r_q$ appears only in one conjunct and can be moved inwards from the top level to this conjunct. Using r_{ij} to denote the j-th variable in the i-th conjunct, we obtain the condition

$$\exists \text{ finite set } u.\ \exists a_1, \ldots, a_p \subseteq u.\ \bigwedge_{i=1}^{m} C_i(u, a_1, \ldots, a_p) \tag{3}$$

where $C_i(u, a_1, \ldots, a_p)$ is the condition

$$\begin{array}{l} \exists r_{i1} \ldots r_{iw_i}. \\ \{\mathsf{obj} \to u, A_1 \mapsto a_1, \ldots, A_p \mapsto a_p, R_{i1} \mapsto r_{i1}, \ldots, R_{iw_1} \mapsto r_{iw_i}\}(\{F_i\} \cup \mathcal{T}_i) \end{array}$$

The idea of our combination method is to simplify each condition $C_i(u, a_1, \ldots, a_p)$ into the truth value of a BAPA formula. If this is possible, we say that there exists a BAPA reduction.

Definition 2 (BAPA Reduction). *If $\mathcal{F}_i$ is a set of formulas and $\mathcal{T}_i \subseteq \mathcal{F}_i$ a theory, we call a function $\rho : \mathcal{F}_i \to \mathcal{F}_{\mathsf{BAPA}}$ a BAPA reduction for $(\mathcal{F}_i, \mathcal{T}_i)$ iff for every formula $F_i \in \mathcal{F}_i$ and for all finite u and $a_1, \ldots, a_p \subseteq u$, the condition*

$$\begin{array}{l} \exists r_{i1} \ldots r_{iw_i}. \\ \{\mathsf{obj} \to u, A_1 \mapsto a_1, \ldots, A_p \mapsto a_p, R_{i1} \mapsto r_{i1}, \ldots, R_{iw_1} \mapsto r_{iw_i}\}(\{F_i\} \cup \mathcal{T}_i) \end{array}$$

is equivalent to the condition $\{\mathsf{obj} \to u, A_1 \mapsto a_1, \ldots, A_p \mapsto a_p\}(\rho(F_i))$

A computable BAPA reduction is a BAPA reduction which is computable as a function on formula syntax trees.

Theorem 3. *Suppose that for every $1 \leq i \leq n$ for $(\mathcal{L}_i, \mathcal{T}_i)$ there exists a computable BAPA reduction ρ_i. Then the problem (1) in Section 4.1 is decidable.*

Specifically, to check satisfiability of $B(F_1, \ldots, F_n)$, compute $B(\rho_1(F_1), \ldots, \rho_n(F_n))$ and then check its satisfiability using a BAPA decision procedure [21, 22].

5 BAPA Reductions

5.1 Monadic Second-Order Logic of Finite Trees

Figure 5 shows the syntax of (our presentation of) monadic second-order logic of finite trees (FT), a variant of weak monadic second-order logic of two successors (WS2S) [18, 33]. The following are the sorts of constants appearing in FT formulas: $\mathsf{succ}_L, \mathsf{succ}_R : \mathsf{obj}^2 \to \mathsf{bool}$. The operations $\subseteq$, $\emptyset$, Univ, $\mathsf{singleton}$, $\cap, \cup$, and $\mathsf{complement}$ have the same sorts (and semantics) as in BAPA.

We interpret the sort obj over finite, prefix-closed sets of binary strings. More precisely, we use $\{1, 2\}$ as the binary alphabet, and we let $[\![\mathsf{obj}]\!] \subseteq \{1, 2\}^*$ such that

$$\forall w \in \{1, 2\}^*.\ (w1 \in [\![\mathsf{obj}]\!] \vee w2 \in [\![\mathsf{obj}]\!]) \to w \in [\![\mathsf{obj}]\!]$$

In each model, $[\![\mathsf{set}]\!]$ is the set of all subsets of $[\![\mathsf{obj}]\!]$. We let $[\![\epsilon]\!]$ be the empty string which we also denote by ϵ. We define

$$[\![\mathsf{succ}_L]\!] = \{(w, w1) \mid w1 \in [\![\mathsf{obj}]\!]\} \quad \text{and} \quad [\![\mathsf{succ}_R]\!] = \{(w, w2) \mid w2 \in [\![\mathsf{obj}]\!]\}$$

$$\begin{aligned}
F &::= P \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid \neg F \mid \forall x{:}s.F \mid \exists x{:}s.F \\
s &::= \mathsf{obj} \mid \mathsf{set} \\
P &::= B_1 = B_2 \mid B_1 \subseteq B_2 \mid r(x, y) \\
r &::= \mathsf{succ}_L \mid \mathsf{succ}_R \\
B &::= x \mid \epsilon \mid \emptyset \mid \mathsf{Univ} \mid \{x\} \mid B_1 \cup B_2 \mid B_1 \cap B_2 \mid B^c
\end{aligned}$$

Fig. 5. Monadic Second-Order Logic of Finite Trees (FT)

The remaining constants and operations on sets are interpreted as in BAPA.

Let $\mathcal{F}_{\mathsf{FT}}$ be the set of all formulas in Figure 5. Let $\mathcal{M}_{\mathsf{FT}}$ be the set of all (finite) structures described above. We define $\mathcal{T}_{\mathsf{FT}}$ as the subset of $\mathcal{F}_{\mathsf{FT}}$ such that F is true in all interpretations from $\mathcal{M}_{\mathsf{FT}}$.

The models of the theory $\mathcal{T}_{\mathsf{FT}}$ correspond up to isomorphism with the interpretations in $\mathcal{M}_{\mathsf{FT}}$.

Lemma 4. *If α is a structure such that $\alpha(\mathcal{T}_{\mathsf{FT}})$ then α is isomorphic to some structure in $\mathcal{M}_{\mathsf{FT}}$.*

Note that any FT formula $F(x)$ with a free variable x of sort obj can be transformed into the equisatisfiable formula $\exists x : \mathsf{obj}.y = \{x\} \wedge F(x)$ where y is a fresh variable of sort set. For conciseness of presentation, in the rest of this section we only consider FT formulas F with $\mathsf{FV}_{\mathsf{obj}}(F) = \emptyset$ and we write $\mathsf{FV}(F)$ as short for $\mathsf{FV}_{\mathsf{set}}(F)$.

Finite tree automata. In the following, we recall the connection between FT formulas and finite tree automata. Let Σ be a finite ranked alphabet. We call symbols of rank 0 constant symbols and a symbol of rank $k > 0$ a k-ary function symbol. We denote by $\mathsf{Terms}(\Sigma)$ the set of all terms over Σ. We associate a position $p \in \{1, \ldots, r_{\max}\}^*$ with each subterm in a term t where $r_{\max}$ is the maximal rank of all symbols in Σ. We denote by $t[p]$ the topmost symbol of the subterm at position p. For instance, consider the term $t = f(g(a, b, c), a)$ then we have $t[\epsilon] = f$ and $t[13] = c$.

A finite (deterministic bottom-up) tree automaton A for alphabet Σ is a tuple (Q, Q_f, ι) where Q is a finite set of states, $Q_f \subseteq Q$ is a set of final states, and ι is a function that associates with each constant symbol $c \in \Sigma$ a state $\iota(c) \in Q$ and with each k-ary function symbol $f \in \Sigma$ a function $\iota(f) : Q^k \to Q$. We homomorphically extend ι from symbols in Σ to Σ-terms. We say that A accepts a term $t \in \mathsf{Terms}(\Sigma)$ if $\iota(t) \in Q_f$. The language $\mathcal{L}(A)$ accepted by A is the set of all Σ-terms accepted by A.

Let F be an FT formula. Define Σ_F to be the alphabet consisting of the constant symbol $\bot$ and all binary function symbols f such that f itself is a function $f : \mathsf{FV}(F) \to \{0, 1\}$. We inductively associate a Σ_F-term $t_{\alpha,w}$ with every structure $\alpha \in \mathcal{M}_{\mathsf{FT}}$ and string $w \in \{1, 2\}^*$ as follows:

$$t_{\alpha,w} = \begin{cases} f_{\alpha,w}(t_{\alpha,w1}, t_{\alpha,w2}) & \text{if } w \in \alpha(\mathsf{obj}) \\ \bot & \text{otherwise} \end{cases}$$

such that for all $x \in \mathsf{FV}(F)$, $f_{\alpha,w}(x) = 1$ iff $w \in \alpha(x)$. The language $\mathcal{L}(F) \subseteq \mathsf{Terms}(\Sigma_F)$ of F is then defined by $\mathcal{L}(F) = \{\, t_{\alpha,\epsilon} \mid \alpha \in \mathcal{M}_{\mathsf{FT}} \wedge \alpha(F) \,\}$.

The following theorem states the connection between the structures satisfying FT formulas and the languages accepted by finite tree automata[2].

Theorem 5 (Thatcher and Wright [33]). *For every FT formula F there exists a finite tree automaton A_F over alphabet Σ_F such that $\mathcal{L}(F) = \mathcal{L}(A_F)$ and A_F can be effectively constructed from F.*

Parikh image. We recall Parikh's commutative image [29]. The Parikh image for an alphabet Σ is the function $\mathsf{Parikh} : \Sigma^* \to \Sigma \to \mathbb{N}$ such that for any word $w \in \Sigma^*$ and symbol $\sigma \in \Sigma$, $\mathsf{Parikh}(w)(\sigma)$ is the number of occurrences of σ in w. The Parikh image is extended pointwise from words to sets of words: $\mathsf{Parikh}(W) = \{\, \mathsf{Parikh}(w) \mid w \in W \,\}$. In the following, we implicitely identify $\mathsf{Parikh}(W)$ with the set of integer vectors $\{\, (\chi(\sigma_1), \dots, \chi(\sigma_n)) \mid \chi \in \mathsf{Parikh}(W) \,\}$ where we assume some fixed order on the symbols $\sigma_1, \dots, \sigma_n$ in Σ.

Theorem 6 (Parikh [29]). *Let G be a context-free grammar and $\mathcal{L}(G)$ the language generated from G then the Parikh image of $\mathcal{L}(G)$ is a semilinear set and its finite representation is effectively computable from G.*

We generalize the Parikh image from words to terms as expected: the Parikh image for a ranked alphabet Σ is the function $\mathsf{Parikh} : \mathsf{Terms}(\Sigma) \to \Sigma \to \mathbb{N}$ such that for all $t \in \mathsf{Terms}(\Sigma)$ and $\sigma \in \Sigma$, $\mathsf{Parikh}(t)(\sigma)$ is the number of positions p in t such that $t[p] = \sigma$. Again we extend this function pointwise from terms to sets of terms.

Lemma 7. *Let A be a finite tree automaton over alphabet Σ. Then the Parikh image of $\mathcal{L}(A)$ is a semilinear set and its finite representation is effectively computable from A.*

5.2 BAPA Reduction for Monadic Second-Order Logic of Finite Trees

In the following, we prove the existence of a computable BAPA reduction for the theory of monadic second-order logic of finite trees.

Let F be an FT formula and let Σ_F^2 be the set of all binary function symbols in Σ_F, i.e., $\Sigma_F^2 \overset{\text{def}}{=} \Sigma_F \setminus \{\bot\}$. We associate with each $\sigma \in \Sigma_F^2$ the *Venn region* $\mathsf{vr}(\sigma)$, which is given by a set-algebraic expression over $\mathsf{FV}(F)$ as follows:

$$\mathsf{vr}(\sigma) \overset{\text{def}}{=} \bigcap_{x \in \mathsf{FV}(F)} x^{\sigma(x)} \quad .$$

Hereby x^0 denotes x^c and x^1 denotes x. Let $\alpha \in \mathcal{M}_{\mathsf{FT}}$ be a model of F. Then the term $t_{\alpha,\epsilon}$ encodes for each $w \in \alpha(\mathsf{obj})$ the Venn region to which w belongs in α, namely $\mathsf{vr}(t_{\alpha,\epsilon}[w])$. Thus, the Parikh image $\mathsf{Parikh}(t_{\alpha,\epsilon})$ encodes the cardinality of each Venn region over $\mathsf{FV}(F)$ in α.

Lemma 8. *Let F be an FT formula then*

$$\mathsf{Parikh}(\mathcal{L}(F))|_{\Sigma_F^2} = \{\, \{\, \sigma \mapsto |\alpha(\mathsf{vr}(\sigma))| \mid \sigma \in \Sigma_F^2 \,\} \mid \alpha \in \mathcal{M}_{\mathsf{FT}} \wedge \alpha(F) \,\} \quad .$$

[2] The theorem was originally stated for WS2S where the universe of all structures is fixed to the infinite binary tree $\{1,2\}^*$ and where all set variables range over finite subsets of $\{1,2\}^*$. It carries over to finite trees in a straightforward manner.

According to Theorem 5 we can construct a finite tree automaton A_F over Σ_F such that $\mathcal{L}(F) = \mathcal{L}(A_F)$. From Lemma 7 follows that $\mathsf{Parikh}(\mathcal{L}(F))$ is a semilinear set whose finite representation in terms of base and step vectors is effectively computable from A_F. From this finite representation we can construct a Presburger arithmetic formula ϕ_F over free integer variables $\{ x_\sigma \mid \sigma \in \Sigma_F \}$ whose set of solutions is the Parikh image of $\mathcal{L}(F)$, i.e.

$$\mathsf{Parikh}(\mathcal{L}(F)) = \{ \{ \sigma \mapsto k_\sigma \mid \sigma \in \Sigma_F \} \mid \{ x_\sigma \mapsto k_\sigma \mid \sigma \in \Sigma \} (\phi_F) \} \qquad (4)$$

Using the above construction of the Presburger arithmetic formula ϕ_F for a given FT formula F, we define the function $\rho_{\mathsf{FT}} : \mathcal{F}_{\mathsf{FT}} \to \mathcal{F}_{\mathsf{BAPA}}$ as follows:

$$\rho_{\mathsf{FT}}(F) \overset{\text{def}}{=} \exists \boldsymbol{x_\sigma}.\, \phi_F \wedge \bigwedge_{\sigma \in \Sigma_F^2} \mathsf{card}(\mathsf{vr}(\sigma)) = x_\sigma$$

where $\boldsymbol{x}_\sigma$ are the free integer variables of ϕ_F.

Theorem 9. *The function ρ_{FT} is a BAPA reduction for $(\mathcal{F}_{\mathsf{FT}}, \mathcal{T}_{\mathsf{FT}})$.*

5.3 Two-Variable Logic with Counting

Figure 6 shows the syntax of (our presentation of) two-variable logic with counting. As usual in two-variable logic with counting, we require that every sub-formula of a formula has at most two free variables. The interpretation of the counting quantifier $\exists^K x{:}\mathsf{set}.F$ for a positive constant K is that there exist at least K distinct elements x for which formula F holds. In the atomic formula $r(x_1, \ldots, x_k)$ we require the relation variable r to be of sort $\mathsf{obj}^k \to \mathsf{bool}$, and require $k > 1$. Following the general setup, the relation variables denote relations on the finite set $[\![\mathsf{obj}]\!]$. We do not impose additional restrictions on the relation variables, so the theory we consider is the theory of two-variable logic over all finite models.

$$\begin{aligned} F &::= P \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid \neg F \mid \exists^K x{:}\mathsf{obj}.F \\ P &::= x_1 = x_2 \mid \{x\} \subseteq A \mid r(x_1, \ldots, x_k) \end{aligned}$$

Fig. 6. Two-Variable Logic with Counting (C^2)

Let $\mathcal{F}_{C2}$ be the set of all formulas in Figure 5. Let $\mathcal{M}_{C2}$ be the set of all (finite) structures described above. We define $\mathcal{T}_{C2}$ as the subset of $\mathcal{F}_{C2}$ such that F is true in all interpretations from $\mathcal{M}_{C2}$.

5.4 BAPA Reduction for Two-Variable Logic with Counting

We next build on the results in [32] to define a BAPA reduction for two-variable logic with counting (C^2). We fix set variables $A_1, \ldots, A_p$ and relation variables $r_1, \ldots, r_q$. Throughout this section, let $\Sigma_A = \{A_1, \ldots, A_p\}$, $\Sigma_R = \{r_1, \ldots, r_q\}$, and $\Sigma_0 = \Sigma_A \cup$

Σ_R. We call $\Sigma_A, \Sigma_A, \Sigma_0$ signatures, because they correspond to notion of signature in in traditional first-order logic formulation of two-variable logic with counting.

Model theoretic types. Define the model-theoretic notion of n-type $\pi_\Sigma(x_1, \dots, x_n)$ in the signature Σ as the maximal consistent set of non-equality literals in Σ whose obj-sort variables are included in $\{x_1, \dots, x_n\}$. [3] Given a structure α such that $\alpha(x_1), \dots, \alpha(x_n)$ are all distinct, α *induces* the n-type

$$\mathsf{ityp}^{\alpha,\Sigma}(x_1, \dots, x_n) = \{L \mid \alpha(L) \wedge \mathsf{FV}(L) \subseteq \{x_1, \dots, x_n\},\ L \text{ is } \Sigma\text{-literal without '='}\}$$

We also define the set of n-tuples for which a type π holds in a given structure α:

$$S^\alpha(\pi(x_1, \dots, x_n)) = \{(e_1, \dots, e_n) \in \alpha(\mathsf{obj})^n \mid \alpha(x_1 := e_1, \dots, x_n := e_n)(\pi)\}$$

If $\Sigma \subseteq \Sigma'$ and π' is an n-type in signature Σ', by $\pi'|_\Sigma$ we denote the subset of π containing precisely those literals from π whose sets and relations belong to Σ. The family of sets $\{S^\alpha(\pi') \mid \pi'|_\Sigma = \pi\}$ is a partition of $S^\alpha(\pi')$.

We will be particularly interested in 1-types. We identify a 1-type $\pi(x)$ in the signature Σ_A with the corresponding Venn region

$$\bigcap \{A_i \mid (\{x\} \subseteq A_i) \in \pi(x)\} \cap \bigcap \{A_i^c \mid (\neg(\{x\} \subseteq A_i)) \in \pi(x)\}.$$

If $\pi_1, \dots, \pi_m$ is the sequence of all 1-types in the signature Σ and α is a structure, let $I^\alpha(\Sigma) = (|S^\alpha(\pi_1)|, \dots, |S^\alpha(\pi_m)|)$. If $\mathcal{M}$ is a set of structures let $I^{\mathcal{M}}(\Sigma) = \{I^\alpha(\Sigma) \mid \alpha \in \mathcal{M}\}$. Note that, if π is a 1-type in Σ and π' a 1-type in Σ' for $\Sigma \subseteq \Sigma'$, then

$$|I^\alpha(\pi)| = \sum_{\pi'|_\Sigma = \pi} |I^\alpha(\pi')|$$

Consequently, if $I^{\mathcal{M}}(\Sigma')$ is semilinear, then so is $I^{\mathcal{M}}(\Sigma)$ for $\Sigma \subseteq \Sigma'$. We will show below that for every C^2 formula ϕ in the signature Σ_0, the set $I^{\mathcal{M}}(\Sigma_0)$ is semilinear for $\mathcal{M} = \{\alpha \mid \alpha(\phi)\}$ the set of models of ϕ. Because $\Sigma_A \subseteq \Sigma_0$, the counts of Venn regions $I^{\mathcal{M}}(\Sigma_A)$ will then also form a semilinear set. By Lemma 1, we will conclude that C^2 is BAPA reducible.

Moving to differentiated chromatic sparse structures preserves 1-types. Let ϕ be a C^2 formula with signature Σ_0 of relation symbols. By Scott normal form transformation [32, Lemma 1] it is possible to introduce fresh set variables and compute another C^2 formula ϕ^* in an extended signature $\Sigma^* \supseteq \Sigma_0$, and compute a constant C_ϕ such that, for all sets u with $|u| \geq C_\phi$,

1. if α_0 is a Σ_0 interpretation with domain u such that $\alpha_0(\phi)$, then there exists its Σ^* extension $\alpha^* \supseteq \alpha_0$ such that $\alpha^*(\phi^*)$;
2. if α^* is a Σ^* interpretation with domain u such that $\alpha^*(\phi^*)$, then for its restriction $\alpha_0 = \alpha^*|_\Sigma$ we have $\alpha_0(\phi)$.

[3] For example, if Σ has one relation variable r, and two set variables A_1, A_2, then each 2-type with free variables x, y contains, for each of the atomic formulas with variables x, y (i.e. $\{x\} \subseteq A_1$, $\{y\} \subseteq A_1$, $\{x\} \subseteq A_2$, $\{y\} \subseteq A_2$, $(x, x) \in r$, $(y, y) \in r$, $(x, y) \in r$, $(y, x) \in r$) either the formula or its negation.

By introducing further fresh set- and relation- symbols, [32, lemmas 2 and 3] shows that we can extend the signature from Σ^* to Σ such that each model α^* in Σ^* extends to a model α in Σ, where α satisfies some further conditions of interest: α is *chromatic* and *differentiated*. [32, Lemma 10] then shows that it is possible to transform a model of a formula into a so-called X-*sparse* model for appropriately computed integer constant X. What is important for us is the following.

Observation 1 *The transformation for obtaining from α_0 a chromatic, differentiated, X-sparse structure α have the property that, for structures of size C_ϕ or more,*

1. *the domain remains the same: $\alpha_0(\mathsf{obj}) = \alpha(\mathsf{obj})$,*
2. *the induced 1-types in the signature Σ_0 remain the same: for each 1-type π in signature Σ_0, $S^{\alpha_0}(\pi) = S^{\alpha}(\pi)$.*

Star types. [32, Definition 9] introduces a star-type $(\pi, \boldsymbol{v})$ (denoted by letter σ) as a description of a local neighborhood of a domain element, containing its induced 1-type π as well as an integer vector $\boldsymbol{v} \subseteq \mathbb{Z}^N$ that count 2-types in which the element participates, where N is a function of the signature Σ. A star type thus gives more precise description of properties of a domain element than a 1-type. Without giving the actual definition of a star type, we note that we can similarly define the set $S^{\alpha}((\pi, \boldsymbol{v}))$ of elements that realize a given star type $(\pi, \boldsymbol{v})$. Moreover, for a given 1-type π, the family of the non-empty among the sets $S^{\alpha}((\pi, \boldsymbol{v}))$ partitions the set $S^{\alpha}(\pi)$.

Frames. The notion of Y-*bounded chromatic frame* [32, Definition 11] can be thought of as representation of a disjunct in a normal form for the formula ϕ^*. It summarizes the properties of elements in the structure and specifies (among others), the list of possible star types $\sigma_1, \ldots, \sigma_N$ whose integer vectors are bounded by Y. For a given ϕ^*, it is possible to effectively compute the set of C_ϕ-bounded frames $\mathcal{F}$ such that $\phi^* \models \mathcal{F}$ holds. The '$\models$' in $\phi^* \models \mathcal{F}$ is a certain syntactic relation defined in [32, Definition 13].

For each frame $\mathcal{F}$ with star-types $\sigma_1, \ldots, \sigma_N$, [32, Definition 14] introduces an effectively computable Presburger arithmetic formula $P_{\mathcal{F}}$ with N free variables; we write $P_{\mathcal{F}}(w_1, \ldots, w_N)$ if $P_{\mathcal{F}}$ is true when these variables take the values $w_1, \ldots, w_N$. The following statement is similar to the main [32, Theorem 1], and can be directly recovered from its proof and the proofs of the underlying [32, lemmas 12,13,14].

Theorem 10. *Given a formula ϕ^*, and the corresponding integer constant C_ϕ, there exists a computable constant X such that if $N \leq X$, if $\sigma_1, \ldots, \sigma_N$ is a sequence of star types in Σ whose integer vectors are bounded by C_ϕ, and $w_1, \ldots, w_N$ are integers, then the following are equivalent:*

1. *There exists a chromatic differentiated structure α such that $\alpha(\phi^*)$, $w_i = |S^{\alpha}(\sigma_i)|$ $1 \leq i \leq N$, and $\bigcup_{i=1}^{N} S^{\alpha}(\sigma_i) = \alpha(\mathsf{obj})$.*
2. *There exists a chromatic frame $\mathcal{F}$ with star types $\sigma_1, \ldots, \sigma_N$, such that $\mathcal{F} \models \phi^*$ and $P_{\mathcal{F}}(w_1, \ldots, w_N)$.*

We are now ready to describe our BAPA reduction. Fix $V_1, \ldots, V_M$ to be the list of all 1-types in signature Σ_A; let $s_1, \ldots, s_M$ be variables corresponding to their counts. By the transformation of models into chromatic, differentiated, X-sparse ones, the Observation 1, and Theorem10, we obtain

Corollary 11. *If $\mathcal{M} = \{\alpha \mid \alpha(\phi^*)\}$, then there is a computable constant X $I^{\mathcal{M}}(\Sigma_A) = \{(s_1, \ldots, s_M) \mid F_{\phi^*}(s_1, \ldots, s_M)\}$ where $F_{\phi^*}(s_1, \ldots, s_M)$ is the following Presburger arithmetic formula*

$$\bigvee_{N, \sigma_1, \ldots, \sigma_N, \mathcal{F}} \exists w_1, \ldots, w_N.\, P_{\mathcal{F}}(w_1, \ldots, w_N) \wedge \bigwedge_{j=1}^{M} s_j = \sum \{w_i \mid V_j = (\pi_i|_{\Sigma_A})\}$$

where N ranges over $\{0, 1, \leq X\}$, $\sigma_1, \ldots, \sigma_N$ range over sequences of C_ϕ-bounded star types (of which there are finitely many), and where $\mathcal{F}$ ranges over the finitely many C_ϕ-bounded frames with star types $\sigma_1, \ldots, \sigma_N$ such that $\mathcal{F} \models \phi^*$.

By adjusting for the small structures to take into account Scott normal form transformation, we further obtain

Corollary 12. *$I^{\mathcal{M}}(\Sigma_A) = \{(s_1, \ldots, s_M) \mid G_\phi(s_1, \ldots, s_M)\}$ for $\mathcal{M} = \{\alpha \mid \alpha(\phi)\}$, where $G_\phi(s_1, \ldots, s_M)$ is the Presburger arithmetic formula*

$$\begin{array}{l} \sum_{i=1}^{M} s_i \geq C_\phi \wedge F_{\phi^*}(s_1, \ldots, s_M) \quad \bigvee \\ \bigvee \{ \bigwedge_{i=1}^{M} s_i = d_i \mid \exists \alpha.\ |\alpha(\mathsf{obj})| < C_\phi \ \wedge (d_1, \ldots, d_M) \in I^\alpha(\Sigma_A)\} \end{array}$$

Theorem 13. *The following is a BAPA reduction for two-variable logic with counting over finite models: given a two-variable logic formula ϕ, compute the BAPA formula*

$$\exists s_1, \ldots, s_M.\, G_\phi(s_1, \ldots, s_M) \ \wedge \bigwedge_{i=1}^{M} \mathsf{card}(V_i) = s_i.$$

5.5 Bernays-Schönfinkel Fragment of First-Order Logic

Figure 7 shows the syntax of (our presentation of) the Bernays-Schönfinkel fragment of first-order logic with equality [4], often called effectively propositional logic (EPR). The interpretation of atomic formulas are the same as for two-variable logic with counting. Quantification is restricted to variables of sort obj and must obey the usual restriction of $\exists\forall$-prenex form that characterizes the Bernays-Schönfinkel class.

$$\begin{array}{l} F ::= \exists x_1 : \mathsf{obj}, \ldots, x_n : \mathsf{obj}.\, \forall y_1 : \mathsf{obj}, \ldots, y_m : \mathsf{obj}.\, B \\ B ::= P \mid B_1 \wedge B_2 \mid B_1 \vee B_2 \mid \neg B \\ P ::= x_1 = x_2 \mid \{x\} \subseteq A \mid r(x_1, \ldots, x_k) \end{array}$$

Fig. 7. Bernays-Schönfinkel Fragment of First-Order Logic

5.6 BAPA Reduction for Bernays-Schönfinkel Fragment

We first consider only EPR formulas without top-level existential quantifiers, but containing constants.

The key to our BAPA reduction for Bernays-Schönfinkel fragment (EPR) is the following theorem, whose proof is in Appendix A.4 and follows from basic properties of ground and non-ground resolution.

Theorem 14 (EPR Reduces to EPR). *Let $\forall \boldsymbol{x}.\phi$ be an EPR formula with set variables $A_1, \ldots, A_p$, relation variables $r_1, \ldots, r_q$, and constants $c_1, \ldots, c_T$. Then the condition $\exists r_1, \ldots, \exists r_q.\phi$ is equivalent to the conjunction of all EPR formulas B with 1) no nested quantifiers, 2) with set variables in B among $A_1, \ldots, A_p$ and with constants are among $c_1, \ldots, c_T$, and 3) for which $\phi \rightarrow B$ is valid.*

We observe that there are only finitely such formulas B in a given language of set variables and constants. We can also enumerate representatives of all such formulas. Note also that the validity check for $\phi \rightarrow B$ over finite models is decidable because $\neg(\phi \rightarrow B')$ is in EPR, and EPR has small model property.

Finally, consider a formula of the form $\exists \boldsymbol{y}.\forall \boldsymbol{x}.\phi(\boldsymbol{x}, \boldsymbol{y})$. By Skolemizing the formula we obtain $\forall \boldsymbol{x}.\phi(\boldsymbol{x}, \boldsymbol{c})$ for which Theorem 14 gives us the reduction B. The final result is then a BAPA formula obtained by unskolemizing B.

5.7 Sets and Multisets

The satisfiability of the quantifier free fragment of multisets with cardinality operators is decidable [30]. There is, in fact, also a BAPA reduction from a quantifier-free formula F containing sets $A_1, \ldots, A_p$ and multisets $M_1, \ldots, M_q$ to a BAPA formula on sets. To obtain a BAPA reduction, we apply the decision procedure in [30] to the formula $F \wedge \bigwedge_{i=1}^{w} \mathsf{card}(V_i){=}k_i$ with fresh integer variables $k_1, \ldots, k_w$. The result is a Presburger arithmetic formula P. If $x_1, \ldots, x_n$ are the variables in P other than $k_1, \ldots, k_w$, the result of the BAPA reduction is then the formula

$$\exists k_1. \ldots \exists k_w. \ (\bigwedge_{i=1}^{w} \mathsf{card}(V_i){=}k_i) \wedge (\exists x_1{:}\mathsf{int}. \ldots \exists x_n{:}\mathsf{int}.\ P)$$

6 Further Related Work

There are combination results for the disjoint combinations of non-stably infinite theories [8, 19, 35]. These results are based on the observation that such combinations are possible whenever one can decide for each component theory whether a model of a specific cardinality exists. Our combination result takes into account not only the cardinality of the models, i.e. the interpretation of the universal set, but cardinalities of Venn regions over the interpretations of arbitrary shared set variables. Thus, it is a natural generalization of the dijoint case leading to a non-disjoint combination of non-stably infinite theories.

Ghilardi [11] proposes a model-theoretic condition for decidability of the non-disjoint combination of theories based on quantifier elimination and local finiteness.

Our combination result also applies to theories that are not locally finite such as monadic second-order logic of finite trees.

Gabbay and Ohlbach [9] present a partial procedure for second-order quantifier elimination, which is not claimed to be complete for a well-defined class of formulas, but terminates on many specific example formulas.

The general combination of weak monadic second-order logics with linear cardinality constraints has been proven undecidable by Klaedtke and Rueß [16, 17]. They introduce the notion of Parikh automata to identify decidable fragments of this logic which inspired our BAPA reduction of MSOL of finite trees. Our combined logic is incomparable to the decidable fragments identified by Klaedtke and Rueß because it supports non-tree structures as well. However, by applying projection to C^2 and the Bernays-Schönfinkel class, we can combine our logic with [16, 17], obtaining an even more expressive decidable logic.

7 Conclusion

Many verification techniques rely on decision procedures to achieve a high degree of automation. The class of properties that such techniques are able to verify is therefore limited by the expressive power of the logics supported by the underlying decision procedures. We have presented a combination result for logics that share operations on sets. This result yields an expressive decidable logic that is useful for software verification. We therefore believe that we made an important step in increasing the class of properties that are amenable to automated verification.

References

1. P. B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof.* Springer (Kluwer), 2nd edition, 2002.
2. C. Barrett and C. Tinelli. CVC3. In *CAV*, volume 4590 of *LNCS*, 2007.
3. D. Basin and S. Friedrich. Combining WS1S and HOL. In *FROCOS*, volume 7 of *Studies in Logic and Computation*. 2000.
4. E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Springer-Verlag, 1997.
5. R. S. Boyer and J. S. Moore. Integrating decision procedures into heuristic theorem provers: A case study of linear arithmetic. In *Machine Intelligence*, volume 11, pages 83–124. Oxford University Press, 1988.
6. L. de Moura and N. Bjørner. Z3: An efficient SMT solver. In *TACAS*, pages 337–340, 2008.
7. S. Feferman and R. L. Vaught. The first order properties of products of algebraic systems. *Fundamenta Mathematicae*, 47:57–103, 1959.
8. P. Fontaine. Combinations of theories and the bernays-schönfinkel-ramsey class. In *VERIFY*, 2007.
9. D. M. Gabbay and H. J. Ohlbach. Quantifier elimination in second-order predicate logic. In B. Nebel, C. Rich, and W. Swartout, editors, *Principles of Knowledge Representation and Reasoning (KR92)*, pages 425–435. Morgan Kaufmann Publishers, Inc., 1992.
10. Y. Ge, C. Barrett, and C. Tinelli. Solving quantified verification conditions using satisfiability modulo theories. In *CADE*, 2007.
11. S. Ghilardi. Model theoretic methods in combined constraint satisfiability. *Journal of Automated Reasoning*, 33(3-4):221–249, 2005.

12. S. Ginsburg and E. Spanier. Semigroups, Pressburger formulas and languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966.
13. E. Grädel, M. Otto, and E. Rosen. Two-variable logic with counting is decidable. In *LICS*, 1997.
14. S. Graf and H. Saidi. Construction of abstract state graphs with PVS. In *Proc. 9th CAV*, pages 72–83, 1997.
15. T. A. Henzinger, R. Jhala, R. Majumdar, and K. L. McMillan. Abstractions from proofs. In *31st POPL*, 2004.
16. F. Klaedtke and H. Rueß. Parikh automata and monadic second-order logics with linear cardinality constraints. Technical Report 177, Institute of Computer Science at Freiburg University, 2002.
17. F. Klaedtke and H. Rueß. Monadic second-order logics with cardinalities. In *ICALP*, volume 2719 of *LNCS*, 2003.
18. N. Klarlund and A. Møller. *MONA Version 1.4 User Manual*. BRICS Notes Series NS-01-1, Department of Computer Science, University of Aarhus, January 2001.
19. S. Krstic, A. Goel, J. Grundy, and C. Tinelli. Combined satisfiability modulo parametric theories. In *TACAS*, volume 4424 of *LNCS*, pages 602–617, 2007.
20. V. Kuncak. *Modular Data Structure Verification*. PhD thesis, EECS Department, Massachusetts Institute of Technology, February 2007.
21. V. Kuncak, H. H. Nguyen, and M. Rinard. Deciding Boolean Algebra with Presburger Arithmetic. *J. of Automated Reasoning*, 2006.
22. V. Kuncak and M. Rinard. Towards efficient satisfiability checking for Boolean Algebra with Presburger Arithmetic. In *CADE-21*, 2007.
23. V. Kuncak and T. Wies. On set-driven combination of logics and verifiers. Technical Report LARA-REPORT-2009-001, EPFL, February 2009.
24. S. K. Lahiri and S. A. Seshia. The UCLID decision procedure. In *CAV'04*, 2004.
25. S. McLaughlin, C. Barrett, and Y. Ge. Cooperating theorem provers: A case study combining HOL-Light and CVC Lite. In *PDPAR*, volume 144(2) of *ENTCS*, 2006.
26. G. Nelson and D. C. Oppen. Simplification by cooperating decision procedures. *ACM TOPLAS*, 1(2):245–257, 1979.
27. S. Owre, J. M. Rushby, and N. Shankar. PVS: A prototype verification system. In D. Kapur, editor, *11th CADE*, volume 607 of *LNAI*, pages 748–752, jun 1992.
28. L. Pacholski, W. Szwast, and L. Tendera. Complexity results for first-order two-variable logic with counting. *SIAM J. on Computing*, 29(4):1083–1117, 2000.
29. R. J. Parikh. On context-free languages. *J. ACM*, 13(4):570–581, 1966.
30. R. Piskac and V. Kuncak. Decision procedures for multisets with cardinality constraints. In *VMCAI*, number 4905 in LNCS, 2008.
31. R. Piskac and V. Kuncak. Linear arithmetic with stars. In *CAV*, 2008.
32. I. Pratt-Hartmann. Complexity of the two-variable fragment with counting quantifiers. *Journal of Logic, Language and Information*, 14(3):369–395, 2005.
33. J. W. Thatcher and J. B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2(1):57–81, 1968.
34. C. Tinelli and C. Ringeissen. Unions of non-disjoint theories and combinations of satisfiability procedures. *Theoretical Computer Science*, 290(1):291–353, Jan. 2003.
35. C. Tinelli and C. Zarba. Combining nonstably infinite theories. *Journal of Automated Reasoning*, 34(3), 2005.
36. T. Wies. *Symbolic Shape Analysis*. PhD thesis, University of Freiburg, 2009.
37. C. G. Zarba. A tableau calculus for combining non-disjoint theories. In *TABLEAUX '02: Proc. Int. Conf. Automated Reasoning with Analytic Tableaux and Related Methods*, pages 315–329, 2002.
38. K. Zee, V. Kuncak, and M. Rinard. Full functional verification of linked data structures. In *ACM Conf. Programming Language Design and Implementation (PLDI)*, 2008.

A Additional Proofs

A.1 Proof of Lemma 4

First, define FT formulas $\mathit{Succ}(x,y)$ and $\mathit{Reach}(x,y)$ as follows, where x, y are variables of sort obj:

$$\begin{aligned}\mathit{Succ}(x,y) &= (\mathsf{succ}_L(x,y) \lor \mathsf{succ}_R(x,y))\\ \mathit{Reach}(x,y) &= (\forall S{:}\mathsf{set}.\, x \in S \land (\forall u,v{:}\mathsf{obj}.\, u \in S \land \mathit{Succ}(u,v) \to v \in S) \to y \in S)\end{aligned}$$

Note that in any structure α, Succ denotes the union of the relations succ_L and succ_R and Reach denotes the reflexive transitive closure of Succ. The following FT formulas are true in all structures $\mathcal{M}_{\mathsf{FT}}$:

$$\begin{aligned}\mathit{OneRoot} &= (\forall x : \mathsf{obj}.(\forall y : \mathsf{obj}.\neg \mathit{Succ}(y,x)) \to x = \epsilon)\\ \mathit{Acyclic} &= (\forall x : \mathsf{obj}, y : \mathsf{obj}.\mathit{Succ}(x,y) \to \neg \mathit{Reach}(y,x))\\ \mathit{NoShared} &= (\forall x : \mathsf{obj}, y : \mathsf{obj}, z : \mathsf{obj}.(\mathit{Succ}(x,z) \land \mathit{Succ}(y,z) \to x = y) \land\\ &\quad\;(\forall x : \mathsf{obj}, y : \mathsf{obj}.\neg \mathsf{succ}_L(x,y) \lor \neg \mathsf{succ}_R(x,y))\end{aligned}$$

Now, assume $\alpha(\mathcal{T}_{\mathsf{FT}})$. Let $r = \alpha(\epsilon)$ and let further $[\![\mathit{Succ}]\!]$ be the relation denoted by $\mathit{Succ}(x,y)$ in α and $[\![\mathit{Reach}]\!]$ the relation denoted by $\mathit{Reach}(x,y)$. Furthermore, let $h : \alpha(\mathsf{obj}) \times \{1,2\}^*$ be the smallest relation satisfying:

- $(r, \epsilon) \in h$
- for all $o_1, o_2 \in \alpha(\mathsf{obj})$, $(o_1, w) \in h$ and $(o_1, o_2) \in \alpha(\mathsf{succ}_L)$ implies $(o_2, w1) \in h$
- for all $o_1, o_2 \in \alpha(\mathsf{obj})$, $(o_1, w) \in h$ and $(o_1, o_2) \in \alpha(\mathsf{succ}_R)$ implies $(o_2, w2) \in h$

Note that for all $o \in \mathsf{dom}(h)$ we have $(r, o) \in [\![\mathit{Reach}]\!]$. Let $W = \mathsf{rng}(h)$. We claim that h is a bijection between $\alpha(\mathsf{obj})$ and W. First, assume that there exists $o_0 \in \alpha(\mathsf{obj}) \setminus \mathsf{dom}(h)$. Then $(r, o_0) \notin [\![\mathit{Reach}]\!]$. Assume there is some o_r, such that $(r, o_r) \notin [\![\mathit{Reach}]\!]$, $(o_r, o_0) \in [\![\mathit{Reach}]\!]$ and for all $o_1 \in \alpha(\mathsf{obj})$, $(o_1, o_r) \notin [\![\mathit{Succ}]\!]$. Thus, in particular $o_r \neq r$. This contradicts the fact that $\mathit{OneRoot}$ is true in α. Thus, assume there is no such o_r. Then there exists an infinite chain $o_0, o_1, o_2, \ldots$ of elements in $\alpha(\mathsf{obj})$ such that for all $i \in \mathbb{N}$, $(o_{i+1}, o_i) \in [\![\mathit{Succ}]\!]$. It follows for all $i, j \in \mathbb{N}$ with $i < j$ that $(o_j, o_i) \in [\![\mathit{Reach}]\!]$. Since $\alpha(\mathsf{obj})$ is finite, there exist $i, j \in \mathbb{N}$ such that $i < j$ and $o_1 = o_j$. Thus, $(o_i, o_{j-1}) \in [\![\mathit{Succ}]\!]$ and $(o_{j-1}, o_i) \in [\![\mathit{Reach}]\!]$. This contradicts the fact that $\mathit{Acyclic}$ is true in α. We conclude $\mathsf{dom}(h) = \alpha(\mathsf{obj})$.

For proving that h is functional, assume that there is $o \in \alpha(\mathsf{obj})$ and distinct $w_1, w_2 \in W$ such that $(o, w_1) \in h$ and $(o, w_2) \in h$. Then there exist distinct o_1 and o_2 such that $(o_1, o) \in [\![\mathit{Succ}]\!]$ and $(o_2, o) \in [\![\mathit{Succ}]\!]$. This contradicts the fact that $\mathit{NoShared}$ is true in α. Thus, h is functional. By similar reasoning we can prove that h is injective.

Since h is a bijection between $\alpha(\mathsf{obj})$ and W, it follows that W is finite. By construction, W is also prefix-closed. Let α' be the structure in $\mathcal{M}_{\mathsf{FT}}$ that is determined by W. Again by construction, h is isomomorphic with respect to the interpretations of ϵ, succ_L, and succ_R in α and α', which proves the lemma.

A.2 Proof of Lemma 7

Let $A = (Q, Q_f, \iota)$ be a tree automaton over ranked alphabet Σ. Consider the context-free grammar $G = (N, T, R, S)$ where S is a fresh start symbol disjoint from Σ and Q, $N = Q$, $T = \Sigma$, and R is the smallest set containing the production rules:

- $S \to q$: if $q \in Q_f$,
- $q \to c$: if c is a constant symbol in Σ and $\iota(c) = q$,
- $q \to f q_1 \dots q_k$: if f is a k-ary function symbol in Σ and $\iota(f)(q_1, \dots, q_k) = q$.

Then G generates all words in Σ^* that result from a pre-order traversal of some Σ-term accepted by A, i.e., $\mathsf{Parikh}(\mathcal{L}(G)) = \mathsf{Parikh}(\mathcal{L}(A))$. Then the lemma follows from Theorem 6.

A.3 Proof of Theorem 9

Let F be an FT formula. For proving the left-to-right direction assume that α is a structure such that $\alpha(\mathcal{T}_{\mathsf{FT}} \cup \{F\})$. From Lemma 4 follows that there exists some structure $\alpha' \in \mathcal{M}_{\mathsf{FT}}$ such that α' is isomorphic to α. Thus, α' is a model of F. According to Lemma 8 there exists $k \in \mathbb{N}$ such that

$$(\{\, \sigma \mapsto |\alpha'(\mathsf{vr}(\sigma))| \mid \sigma \in \Sigma_F^2 \,\} \cup \{\bot \mapsto k\}) \in \mathsf{Parikh}(\mathcal{L}(F))$$

From Equation 4 and the definition of ρ_{FT} follows $\alpha'|_{\mathsf{FV}(F)}(\rho_{\mathsf{FT}})$. Since α and α' agree on the interpretations of the free set variables of F up to isomorphism, we conclude $\alpha|_{\mathsf{FV}(F)}(\rho_{\mathsf{FT}})$.

For the right-to-left direction assume that α is a structure such that $\alpha(\rho_{\mathsf{FT}})$ holds. We need to find interpretations for succ_L and succ_R that extend α to a model of $\mathcal{T}_{\mathsf{FT}} \cup \{F\}$. From the definition of ρ_{FT} and Equation 4 follows that there exists $k \in \mathbb{N}$ such that

$$(\{\, \sigma \mapsto |\alpha(\mathsf{vr}(\sigma))| \mid \sigma \in \Sigma_F^2 \,\} \cup \{\bot \mapsto k\}) \in \mathsf{Parikh}(\mathcal{L}(F)) \enspace .$$

From Lemma 8 follows that there exists $\alpha' \in \mathcal{M}_{\mathsf{FT}}$ such that $\alpha'(F)$ and for all $\sigma \in \Sigma_F^2$, $|\alpha(\mathsf{vr}(\sigma))| = |\alpha'(\mathsf{vr}(\sigma))|$. Since α and α' agree on the cardinalities of all Venn regions over $\mathsf{FV}(F)$, there exists a bijection $h : \alpha(\mathsf{obj}) \to \alpha'(\mathsf{obj})$ which is isomorphic with respect to the interpretation of all $x \in \mathsf{FV}(F)$ in α and α'. Choose one such isomorphism h. Let $s_L, s_R : \alpha(\mathsf{obj})^2 \to \{\mathsf{true}, \mathsf{false}\}$ be such that for all $o_1, o_2 \in \alpha(\mathsf{obj})$, $(o_1, o_2) \in s_{L,R}$ iff $(h(o_1), h(o_2)) \in \alpha'(\mathsf{succ}_{L,R})$. Then $(\alpha \cup \{\mathsf{succ}_L \mapsto s_L, \mathsf{succ}_R \mapsto s_R, \epsilon \mapsto h(\epsilon)\})$ is a model of $\mathcal{T}_{\mathsf{FT}} \cup \{F\}$.

A.4 Proof of Theorem 14

Let $B_1, \dots, B_N$ be all such EPR formulas and let B^* be their conjunction. Let α be a finite structure such that $\alpha(B^*)$. Assume without loss of generality that the elements of $\alpha(\mathsf{obj}) = \{d_1, \dots, d_U\}$ are disjoint from the set of original constants $c_1, \dots, c_T$, and introduce them as fresh constants. Let ϕ_0 denote the grounding of ϕ for all possible N-tuples of concrete domain elements:

$$\bigwedge_{\boldsymbol{w} \in \alpha(\mathsf{obj})^N} \phi[\boldsymbol{x} := \boldsymbol{w}]$$

On this ground formula, consider the propositional resolution that systematically resolves all clauses along the ground atomic formulas L_i of the form $r(d_{i_1}, \ldots, d_{i_k})$ where r is a relation variable. Whenever two literals are resolved while containing L, we remove the clauses that participated in resolution and keep only the resolvent. If the current conjunction of all clauses is $C_1 \wedge \ldots \wedge C_n$, then (by a simple property of propositional resolution), performing the resolution on all pairs of clauses containing L results in a set of clauses (propositionally) equivalent to $\exists L.(C_1 \wedge \ldots \wedge C_n)$. If $L_1, \ldots, L_k$ are all possible atomic formulas containing relation symbols instantiated with all possible domain elements, let S denote the ground clause resulting from eliminating all of them; this set of clauses is equivalent to $\exists L_1. \ldots \exists L_k. \phi_0$.

Because ϕ_0 is a ground instance of ϕ, by the lifting lemma for non-ground resolution, there exists a resolution proof of length no longer than the proof for S, and that derives a finite set of clauses Q such that S is an instance of Q. Furthermore, there is such Q that does not contain relation symbols. Because resolution does not introduce new function symbols, Q is also an EPR formula. Therefore, B^* entails Q. Because $\alpha(B^*)$, we have $\alpha(Q)$ and thus $\alpha(S)$. Therefore, $\exists L_1. \ldots \exists L_k. \phi_0$, so there exists an interpretation for the clauses $L_1, \ldots, L_k$ that makes ϕ_0 true. Therefore, there exists an extension α' of α interpreting the relations r_1 to r_q such that ϕ is true in α'.