An Algorithm for Deciding BAPA:
Boolean Algebra with Presburger Arithmetic*

Viktor Kuncak^1, Huu Hai Nguyen^2, and Martin Rinard^1,2

^1 MIT CSAIL, Cambridge, USA
^2 Singapore-MIT Alliance

Abstract. We describe an algorithm for deciding the first-order multisorted theory BAPA, which combines 1) Boolean algebras of sets of uninterpreted elements (BA) and 2) Presburger arithmetic operations (PA). BAPA can express the relationship between integer variables and cardinalities of a priory unbounded finite sets, and supports arbitrary quantification over sets and integers.

Our motivation for BAPA is deciding verification conditions that arise in the static analysis of data structure consistency properties. Data structures often use an integer variable to keep track of the number of elements they store; an invariant of such a data structure is that the value of the integer variable is equal to the number of elements stored in the data structure. When the data structure content is represented by a set, the resulting constraints can be captured in BAPA. BAPA formulas with quantifier alternations arise when verifying programs with annotations containing quantifiers, or when proving simulation relation conditions for refinement and equivalence of program fragments. Furthermore, BAPA constraints can be used for proving the termination of programs that manipulate data structures, and have applications in constraint databases.

We give a formal description of a decision procedure for BAPA, which implies the decidability of BAPA. We analyze our algorithm and obtain an elementary upper bound on the running time, thereby giving the first complexity bound for BAPA. Because it works by a reduction to PA, our algorithm yields the decidability of a combination of sets of uninterpreted elements with any decidable extension of PA. Our algorithm can also be used to yield an optimal decision procedure for BA through a reduction to PA with bounded quantifiers.

We have implemented our algorithm and used it to discharge verification conditions in the Jahob system for data structure consistency checking of Java programs; our experience with the algorithm is promising.

1 Introduction

Program analysis and verification tools can greatly contribute to software reliability, especially when used throughout the software development process. Such tools are even more valuable if their behavior is predictable, if they can be applied to partial programs, and if they allow the developer to communicate the design information in the form of specifications. Combining the basic idea of [18] with decidable logics leads to analysis tools that have these desirable properties. Such analyses are precise (because formulas

* CADE-20.
represent loop-free code precisely) and predictable (because the checking of verification conditions terminates either with a realizable counterexample or with a sound claim that there are no counterexamples).

A key challenge in this approach to program analysis and verification is to identify a logic that captures an interesting class of program properties, but is nevertheless decidable. In [29] we identify the first-order theory of Boolean algebras (BA) as a useful language for reasoning about dynamically allocated objects: BA allows expressing generalized typestate properties and reasoning about data structures as dynamically changing sets of objects. (We are interested in BA of all subsets of some set; this theory was shown decidable already in [31, 46], see [22] for the discussion of other models of Boolean algebra axioms.)

The motivation for this paper is the fact that we often need to reason not only about the data structure content, but also about the size of the data structure. For example, we may want to express the fact that the number of elements stored in a data structure is equal to the value of an integer variable that is used to cache the data structure size, or we may want to introduce a decreasing integer measure on the data structure to show program termination. These considerations lead to a natural generalization of the first-order theory of BA of sets, a generalization that allows integer variables in addition to set variables, and allows stating relations of the form $|A| = k$ meaning that the cardinality of the set $A$ is equal to the value of the integer variable $k$. Once we have integer variables, a natural question arises: which relations and operations on integers should we allow? It turns out that, using only the BA operations and the cardinality operator, we can already define all operations of PA. This leads to the structure BAPA, which properly generalizes both BA and PA.

As we explain in Section 2, a version of BAPA was shown decidable already in [14] (which also proves the well-known Feferman-Vaught theorem [19, Section 9.6] about the products of first-order theories). Recently, a decision procedure for a fragment of BAPA without quantification over sets was presented in [55], cast as a multi-sorted theory. Starting from [29] as our motivation, we have observed in [26] the decidability of the full BAPA (which was initially left open in [55]). An algorithm for a single-sorted version of BAPA was presented independently in [42] as a way of evaluating queries in constraint databases; [42] leaves open the complexity of the satisfiability problem.

Our paper gives the first formal description of a decision procedure for the full first-order theory of BAPA. Furthermore, we analyze our decision procedure and show that it yields an elementary upper bound on the complexity of BAPA. Our result is the first upper complexity bound on BAPA; along with a lower bound from PA, we obtain a good estimate of BAPA worst-case complexity. We have also implemented our decision procedure; we report on our initial experience in using the decision procedure in the context of a system for checking data structure consistency.

**Contributions.** We summarize the contributions of our paper as follows.

1. As a motivation for BAPA, we show in Section 3 how BAPA constraints can be used for program analysis and verification by expressing 1) data structure invariants, 2) the correctness of procedures with respect to their specifications, 3) simulation relations between program fragments, and 4) termination conditions for programs that manipulate data structures.