Research Statement Constantinos Bartzis

Formal specification and verification is becoming a crucial part of software and hardware systems design, due to the increasing complexity of such systems and the prohibitive cost of errors. I am mostly interested in model checking, an automated verification methodology based on the systematic exploration of the state-space of a system. Model checking is now a well established area of computer science and numerous tools have been implemented for both finite and infinite state systems. A common concern in the design of such tools is efficiency. Undoubtedly, symbolic model checking has been a major breakthrough towards this direction, since it has succeeded in analyzing systems with huge state-spaces. However the choice of a specific symbolic representation for sets of states greatly influences the performance of a symbolic model checker, which is exactly where my research focuses on.

For example, current implementations of BDD based symbolic model checkers are inefficient in representing linear arithmetic constraints on bounded integer variables. This is because the size of BDDs can be exponential in the number and size of the integer variables in the worst case. Handling linear arithmetic constraints efficiently is an important problem since such constraints are very common in both hardware and software systems. I solved this problem by giving linear time and space BDD construction algorithms, proving their complexity and experimentally demonstrating their efficiency [BB03a, BB06]. I experimentally demonstrated that these algorithms can be used to improve the performance of existing BDD based symbolic model checkers such as SMV [SMV] and NuSMV [NuSMV].

Infinite state systems whose behavior and properties can be described by linear arithmetic constraints on unbounded integer variables are quite common in software applications. Typical examples are concurrent, parameterized systems and hybrid systems. In [BB02, BB03b] I proposed an efficient automata-based representation for such constraints and demonstrated its superiority over other automata-based representations and the polyhedral representation that had been successfully used earlier. All symbolic construction and manipulation algorithms have been fully implemented and are now part of the Action Language Verifier (ALV), an infinite state model checker developed by our research group [YBB05].

In automata-based symbolic model checking, the most time consuming operation is image computation (either forward or backward), i.e., computation of the set of states resulting from a given set of states in one computational step. This is due to the fact that image computation involves determinization of automata, an operation with exponential worst case complexity. In [BB03c] I proposed new techniques for image computation that are provably efficient (polynomial time and space complexity) for a restricted but quite common class of transition systems, where the transition relation can be characterized as a set of guarded updates in disjunctive normal form. I implemented these algorithms and experiments showed that they improve the efficiency of automata based representations significantly. My experiments also indicated that, in a lot of cases, automata encoding with the proposed image computations is as efficient as other more restrictive canonical representations, such as Hilbert's basis used in the BRAIN tool [RV02].

Model checking of infinite state systems is undecidable; therefore, there are instances for which fixpoint computations used in infinite state model checkers do not converge. Given a widening operator one can compute an upper approximation of a least fixpoint in finite number of steps even if the least fixpoint itself is not computable. In [BB04] I proposed a widening operator for automata encoding integer sets. I demonstrated how widening can be used to verify safety properties that cannot be verified otherwise. I also showed that the dual of the widening operator can be used to detect counter examples for liveness properties. Experiments indicate that the same technique can be used to significantly improve the efficiency of our model checking tool ALV [YBB05] by reducing the necessary number of fixpoint iterations.

Future Research Directions

In spite of a number of positive results in my research work, there are still open problems, which I plan to look into in the future. For example, in [BB03c] we showed that for a restricted class of systems, image computation can be performed efficiently (without the expected exponential blowup). However, it is not known which is the widest class of systems with that property. As a second example, in [BB04] we presented an approximate verification algorithm based on widening, for which we would prove certain preciseness properties. Generalizing these properties would be an important result, since it would allow efficient verification of a broader class of systems. Experiments show that the algorithm always terminates. Nevertheless, there is no obvious theoretical justification for this fact.

State-of-the-art software analysis tools such as Java PathFinder are based on explicit-state model checking and therefore have to face the well known state explosion problem. To overcome this obstacle, several advanced techniques are incorporated, such as (user provided) abstraction and static analysis. On the other hand, symbolic model checking has been established as an efficient alternative to explicit-state approaches, mostly for hardware applications that do not involve arithmetic. My research results indicate that certain symbolic representations allow for efficient model checking of software systems with either finite or infinite state-spaces. Therefore, I am interested in integrating these symbolic representations and the accompanying algorithms in a software verification tool.

Automata can be used as a canonical representation for formulas with both boolean and integer variables. There are other symbolic representations for different types, such as shape graphs for representing the configurations of the heap. I plan to design an automata-based representation that will also incorporate more data types, such as pointers. In conjunction with widening techniques, I expect the new representation to allow for efficient verification of concurrent programs with linked lists and other complex data structures.

An alternative approach to analyzing heap manipulating software with dynamic memory allocation, pointers etc., is program analysis using a suitable assertion language such as Separation Logic. Unfortunately the full logic is undecidable and there is no known efficient algorithm for its decidable parts. I propose two independent decision procedures for a practically useful fragment via 1) An efficient translation to Quantified Boolean Formulae (QBF), for which many modern award winning tools exist 2) An encoding into the automata based representation [BB03b], along with a very efficient quantifier elimination procedure based on the ideas in [BB03c].

The World Wide Web is more and more used for application to application communication. The programmatic interfaces made available are referred to as web services. XML is the standard format used for data transmission of web services and XML query languages are used for manipulation of the data. In order to model check web services symbolically, one would need a symbolic representation capable to capture the tree-structure of XML data and handle the high expressiveness of XML query languages. I plan to design such a symbolic representation with efficiency in mind.

Beyond Model Checking

Recently, I acquired an interest in areas closely related to formal verification, which are described in this section.

Security A majority of attacks on computer systems result from a combination of vulnerabilities exploited by an intruder to break into the system. An Attack Graph is a general formalism used to model security vulnerabilities of a system and all possible sequences of

exploits which an intruder can use to achieve a specific goal. Attack Graphs can be constructed automatically using off the shelf model-checking tools. However, for real systems, the size and complexity of Attack Graphs greatly exceeds human ability to visualize, understand and analyze. Therefore, it is useful to identify relevant portions of an Attack Graph. To achieve this, we developed a ranking scheme for the states of an Attack Graph [MBZCW06]. Rank of a state shows its importance based on factors such as the probability of an intruder reaching that state. Given a Ranked Attack Graph, the system administrator can concentrate on relevant subgraphs to deploy appropriate security measures. We also defined a metric of security of the system based on ranks, which the system administrator can use to compare Attack Graphs and determine the effectiveness of various defense measures. We designed two algorithms to rank states of an Attack Graph based on the probability of an attacker reaching those states. The first algorithm is similar to the PageRank algorithm used by Google to measure importance of pages on the World Wide Web. It is flexible enough to model a variety of situations, efficiently computable for large sized graphs and offers the possibility of approximations using graph partitioning. The second algorithm ranks individual states based on the reachability probability of an attacker in a random simulation. Finally, we give examples of an application of ranking techniques to multi-stage cyber attacks.

We are currently investigating additional applications of our ranking technique in the areas of test case generation, guided simulation and semi-formal methods such as bounded model checking.

Propositional Satisfiability Most state-of-the-art SAT solvers are based on some variation of the Davis-Putnam algorithm (DP) and require the input formula to be in clausal form (CNF). However, typical formulae that arise in practice, in the context of formal verification, are non-clausal. We developed SatMate [JBC06], a new non-clausal SAT-solver based on the General Matings technique instead of DP search. Our technique is able to handle non-clausal formulae involving AND, OR and NOT operators without destroying their structure or introducing new variables. Application of several heuristics can improve the efficiency of our algorithm significantly. Experimental results showed that SatMate is more efficient than current state-of-the-art SAT solvers on a class of non-clausal benchmarks.

A different way to attack the complexity of the Satisfiability problem is to devise efficient parallel algorithms that make use of today's emerging multi-processor and multi-core architectures. Current parallel SAT algorithms fail to follow the progress made by their sequential counterparts. Most of these approaches parallelize subtasks of the sequential algorithm, such as exploration of the state-space or Boolean constraint propagation. As a result, some of the known successful heuristics lose their effectiveness. We propose a different strategy based on two fundamental ideas: 1) Dynamic splitting of the problem using well tested heuristics to achieve maximal processor utilization 2) Asynchronous communication of intermediate results in the form of learned clauses to avoid double work and synchronization overhead. Preliminary results are very encouraging: for many competition-quality benchmarks our approach achieves super-linear speedup.

Abstract Interpretation Formal verification techniques based on abstraction have had a lot of success in combating the well known state-space explosion problem. In particular, predicate abstraction has been widely used to verify both software and hardware systems. The main shortcoming of traditional methods of computing an abstraction is that they require a large number of expensive calls to a theorem prover, typically exponential in the number of predicates. We developed a BDD-based framework for predicate abstraction that takes advantage of the efficiency results in [BB06]. We extended our approach to handle bit vector operations such as shifts and bitwise Boolean operations. We also developed an ad hoc linear-time quantification algorithm for BDDs. Experimental evaluation of our technique shows that it scales very well with the number of predicates.

References

[BB02] Constantinos Bartzis and Tevfik Bultan. Automata-based representations for arithmetic constraints in automated verification. In Jean-Marc Champarnaud and Denis Maurel, editors, *Proceedings of the Seventh International Conference on Implementation and Application of Automata (CIAA 2002)*, volume 2608 of *Lecture Notes in Computer Science*, pages 282–288. Springer, 2003.

[BB03a] Constantinos Bartzis and Tevfik Bultan. Construction of efficient BDDs for bounded arithmetic constraints. In *Proceedings of the 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2003)*, volume 2619 of *Lecture Notes in Computer Science*, pages 394– 408. Springer, 2003.

[BB03b] Constantinos Bartzis and Tevfik Bultan. Efficient symbolic representations for arithmetic constraints in verification. *International Journal of Foundations of Computer Science*, 14(4):605–624, 2003.

[BB03c] Constantinos Bartzis and Tevfik Bultan. Efficient image computation in infinite state model checking. In *Proceedings of the 15th International Conference on Computer Aided Verification (CAV 2003)*, volume 2725 of *Lecture Notes in Computer Science*, pages 249–261. Springer, 2003.

[BB04] Constantinos Bartzis and Tevfik Bultan. Widening Arithmetic Automata. *Proceedings of the 16th International Conference on Computer Aided Verification (CAV 2004)*, volume 3114 of *Lecture Notes in Computer Science*, pages 321– 333. Springer, 2004.

[BB06] Constantinos Bartzis and Tevfik Bultan. Efficient BDDs for Bounded Arithmetic Constraints. *International Journal on Software Tools for Technology Transfer (STTT)*, 8(1):26–36, 2006.

[JBC06] Himanshu Jain, Constantinos Bartzis and Edmund Clarke. Satisfiability Checking of Non-clausal Formulae using General Matings. *Proceedings of the 9th International Conference in Theory and Applications of Satisfiability Testing (SAT 2006)*, volume 4121 of *Lecture Notes in Computer Science*, pages 75-89. Springer 2006.

[MBZCW06] Vaibhav Mehta, Constantinos Bartzis, Haifeng Zhu, Edmund Clarke and Jeannette Wing. Ranking Attack Graphs. *Proceedings of the 9th International Symposium Recent Advances in Intrusion Detection (RAID 2006)*, volume 4219 of *Lecture Notes in Computer Science*, pages 127-144. Springer 2006.

[NuSMV] NuSMV. http://nusmv.irst.itc.it/.

[RV02] Tatiana Rybina and Andrei Voronkov. Using canonical representations of solutions to speed up infinite state model checking. *Proceedings of the 14th International Conference on Computer Aided Verification (CAV 2002)*, pages 400-411, 2002.

[SMV] SMV. www.cs.cmu.edu/~modelcheck/smv.html.

[YCB05] Tuba Yavuz-Kahveci, Constantinos Bartzis and Tevfik Bultan. Action Language Verifier, Extended. *Proceedings of the 17th International Conference on Computer Aided Verification (CAV 2005)*, volume 3576 of *Lecture Notes in Computer Science*, pages 413–417. Springer 2005.