Research Statement

Tushar Sharma

July 17, 2017

Email: tsharma@cs.wisc.edu,
Homepage: http://pages.cs.wisc.edu/~tsharma

My research interests lie in program analysis and verification techniques with focus on abstract interpretation. Most critical applications, such as airplane and rocket controllers, need correctness guarantees. Usually these correctness guarantees can be described as safety properties in the form of assetions. Proving an assertion is, in general, an undecidable problem. Nevertheless, there exist static-analysis techniques that are able to verify automatically some kinds of program assertions. One such technique is abstract interpretation [2], which soundly abstracts the concrete executions of the program to elements in an abstract domain, and checks the correctness guarantees using the abstraction. The building block for the abstraction of the program is an *abstract domain*. An abstract domain defines the level and detail of the abstraction. My research advances the state-of-the-art in abstract interpretation by providing new abstract domains and, abstract-transformer construction techniques, that allow one to prove assertions and provide function summaries for programs with machine integers. In the following, I will describe these abstract domains and, associated abstract-transformer techniques, and present some directions that I plan to pursue in the future.

Broadly, my contributions to the numeric analysis of programs with machine integers fall into two categories: i) Bit-Vector Precise Equality Abstract Domains, ii) Bit-Vector Precise Inequality Abstract Domains.

- **Bit-Vector Precise Equality Abstract Domains**
  - **Abstract Domains of Affine Relations:** An affine relation is a *linear-equality* constraint over a given set of variables that hold machine integers. In this work [3, 4], I, along with several other contributors, compared the domain introduced by Müller-Olm/Seidl (denoted by MOS) [6] and the domain introduced by King/Søndergaard (denoted by KS) [5], along with several variants. These domains can express a class of affine relations over bitvectors. We showed that MOS and KS are, in general, *incomparable* and give sound interconversion methods for KS and MOS.

A third domain for representing affine relations, called AG, which stands for *affine generators*, was also introduced. Furthermore, we present an experimental study comparing the precision and performance of analyses with the KS and MOS domains.

- **Abstraction Framework for Affine Transformers:** In this work [7], I defined the Affine-Transformers Abstraction Framework, which represents a new family of numerical abstract domains. This framework is parameterized by a base numerical abstract domain, and allows one to represent a set of affine transformers (or, alternatively, certain disjunctions of transition formulas). Specifically, this framework is a generalization of the MOS domain. The choice of the base abstract domain allows the client to have some control over the performance/precision trade-off.

- **Bit-Vector Precise Inequality Abstract Domains**
  - **An Abstract Domain for Bit-vector Inequalities:** This work [9] describes the design and implementation of a new abstract domain, called the *Bit-Vector Inequality* domain, which is capable of capturing certain inequalities over bit-vector-valued variables (which represent a program's registers and/or its memory variables). This domain tracks properties of the values of selected registers and portions of memory via *views*, and provides automatic heuristics to gather equality and inequality views from the program. Furthermore, experiments are provided to show the usefulness of the Bit-Vector Inequality domain.
  - **Sound Bit-Precise Numerical Domains Framework for Inequalities:** This work [8] introduces a class of abstract domains, parametrized on a base domain, that is sound with respect to bitvectors whenever the base domain is sound with respect to mathematical integers. The base domain can be any numerical abstract domain. We also describe how to create abstract transformers for this framework that incorporate lazy wrap-around to achieve more precision, without sacrificing soundness with respect to bitvectors. We use a finite number of disjunctions of base-domain elements to help retain precision. Furthermore, we present experiments to empirically demonstrate the usefulness of the framework.

Other Contributions

Consistency Without Ordering: This work [1] introduced the No-Order File System (NoFS), a simple, lightweight file system. NoFS uses backpointer-

based consistency to provide crash consistency without ordering writes as they go to disk. A formal model was utilized to prove that NoFS provides data consistency in the event of system crashes. Experiments showed that NoFS is robust to such crashes, and delivers excellent performance across a range of workloads. Backpointer-based consistency thus allows NoFS to provide crash consistency without resorting to the heavyweight machinery of traditional approaches.

Speeding up machine-code synthesis: This work [10] improved the performance of a machine-code synthesizer McSynth. McSynth brought down synthesis time from days to minutes, but it was still not fast enough: McSynth times out for several larger Quantifier-Free Bit-Vector (QFBV) formulas; even for smaller formulas, McSynth takes several minutes to find an implementation. Consequently, if a binary-rewriter client supplies a formula as input to McSynth, the client has to wait several minutes before McSynth finds an implementation. This delay might not be tolerable for a client that has to invoke the synthesizer multiple times to rewrite an entire binary (e.g., a machine-code partial evaluator). This work introduced McSynth++, that provided several improvements to the synthesis algorithm used in McSynth. In addition to a novel pruning heuristic, the improvements incorporate a number of ideas known from the literature, which were adapted in novel ways for the purpose of speeding up machine-code synthesis. Experiments for IA-32 showed that the improvements enable synthesis of code for 12 out of 14 formulas on which McSynth times out, speeding up the synthesis time by at least 1981X, and for the remaining formulas, speeds up synthesis by 3X.

Future Directions

Bit-Vector Precise Equality Abstract Domains: One obvious future work is to provide an implementation of the Affine-Transformers Abstraction framework, and compare the performance and precision of this framework for different base domains. A second direction is to extend this generic framework so that it can be applied for sound program analysis for floating-point numbers. There have been a lot of work for tracking floating-point arithmetic soundly in an abstract domain. Once the generic abstract-transformation-abstraction framework is implemented for floating-point numbers, these base domains can be used to provide new sound abstract domains capable of expressing a certain class of disjunction of affine transformers over floating-point numbers.

Bit-Vector Precise Inequality Abstract Domains: One future research direction is improving the lazy wrap-around method to improve preci-

sion. Intuitively, wrap-around operation performs join operation that leads to loss in precision. Lazy wrap-around aims to improve the precision by only calling wrap-around procedure at guard and cast statements. While calling wrap-around at these points is essential for correctness, the result of a wrap-around can be modified to a value that requires less disjunctions to express by using a dual of wrap-around operation, thus leading to improved precision. The challenge is in implementing the dual operation efficiently.

Symbolic abstraction is an automatic methods to create abstract transformers. However, this method is not applicable to the polyhedra domain because the polyhedral domain has both infinite ascending and descending chains. Symbolic abstraction method is desirable because the concrete semantics of bit-twiddling operations can be precisely described in Quantifier-Free Bit-Vector (QFBV) logic. Recent improvements in the SMT optimization techniques provides insights in designing a symbolic abstraction for the polyhedra domain. Morever, with the help of the dual of wrap-around operation, sound and most abstract transformers can be created for bit-vector-sound inequality abstract domains.

References

[1] V. Chidambaram, T. Sharma, A. Arpaci-Dusseau, and R. Arpaci-Dusseau. Consistency without ordering. In *FAST*, 2012.

[2] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction of approximation of fixed points. In *POPL*, pages 238–252, 1977.

[3] M. Elder, J. Lim, T. Sharma, T. Andersen, and T. Reps. Abstract domains of affine relations. In *SAS*, 2011.

[4] M. Elder, J. Lim, T. Sharma, T. Andersen, and T. Reps. Abstract domains of affine relations. *TOPLAS*, 2014.

[5] A. King and H. Søndergaard. Automatic abstraction for congruences. In *VMCAI*, 2010.

[6] M. Müller-Olm and H. Seidl. Analysis of modular arithmetic. *TOPLAS*, 2007.

[7] T. Sharma and T. Reps. A new abstraction framework for affine transformers. In *SAS*, 2017.

[8] T. Sharma and T. Reps. Sound bit-precise numerical domains. In *VMCAI*, 2017.

[9] T. Sharma, A. Thakur, and T. Reps. An abstract domain for bit-vector inequalities. Tech. Rep. TR-1789, Comp. Sci. Dept., Univ. of Wisconsin, Madison, WI, April 2013.

[10] V. Srinivasan, T. Sharma, and T. Reps. Speeding up machine-code synthesis. In *OOPSLA*, 2016.