2009 11th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing

SMT solvers for Testing, Program Analysis and Verification at Microsoft

Nikolaj Bjørner
Microsoft Research, One Microsoft Way, Redmond, WA, 98074, USA
nbjorner@microsoft.com

***Abstract*—Modern program analysis and model-based tools are increasingly complex and multi-faceted software systems. However, at their core is invariably a component using logic for describing states and transformations between system states. The system Z3, developed at Microsoft Research, is a *Satisfiability Modulo Theories* (SMT) solver. It provides logic inference capabilities that are critical for the functionality of these systems. The tutorial exemplifies a number of applications of Z3 at Microsoft.**

***Keywords*-Theorem Proving, SMT, Static Program Analysis, Program Verification, Dynamic Symbolic Execution.**

I. Introduction

Decision procedures for checking satisfiability of logical formulas are crucial for many testing, program analysis and verification applications. The tutorial is centered around the Microsoft SMT solver, Z3 [1]. We describe how SMT solvers, and Z3 in particular, are used in program analysis and verification. Several concrete applications from program analysis at Microsoft are presented. The applications covered in the tutorial are summarized in the following.

A. Pex: Dynamic Symbolic Execution for .NET

Pex is a (*Program EXploration* tool. The main functionality provided by Pex is to extract concrete test inputs from *parametrized* test cases. For example a program

```
bool Validate(int[] digits)
{
   int sum = 0;
   for (bool alt = true, int i = digits.Length − 1;
        i >= 0; i−−, alt = !alt) {
        if (alt) {
           int temp = digits[i];
           temp *= 2;
           if (temp > 9) temp −= 9;
           sum += temp;
        }
        else {
           sum += digits[i];
        }
   }
   return 0 == sum % 10;
}
```

that takes as input an arbitrary sequence of digits representing a potential credit-card number and checks if the digits pass a check-sum. The program contains internal branches and returns different results depending on the input. Pex can take a program, such as `Validate` and create tests that exercise the different branches and return values. As a side-effect, Pex also checks for boundary cases, such as the case where `digits` is `null`.

B. SDV: The Static Driver Verifier

The Static Driver Verifier is a software model checker that ships with Microsoft's driver development kit. It is specialized to check drivers for absence of common programming errors. SDV uses Z3 as part of the process of building a finite state abstraction from driver code, written in C.

C. PREfix: Bit-precise Scalable Static Analysis

We recently integrated Z3 into the scalable static analysis engine PREfix. PREfix has been used at Microsoft over the last decade. It contains a custom theorem prover, but it is incomplete and does not model the true semantics of machine integers. The integration with Z3's bit-vector theory allowed uncovering a number of bugs related to integer overflows.

D. Spec#: C# + Contracts

Spec# is a programming language that builds upon the syntax and constructs found in C#, but in addition, it provides ways for users to annotate programs with *contracts*. The contracts are in the form of object invariants, pre-, and post-conditions. Type-checking Spec# programs then requires theorem proving, where particularly SMT solvers are suitable as they provide built-in support for the types found in programming languages.

E. VCC: Verifying C Compiler for the Viridian Hyper-Visor

The VCC system is a freely available system for verifying deep properties of low-level C code. It comes with annotation support for the C memory system and concurrency.

F. M3/FORMULA: Model Program Exploration

The M3 and FORMULA systems take as starting point higher-level descriptions of software, or so-called software models. These descriptions are not necessarily executable, or do not correspond to a fully functional software deployment. In the case of FORMULA, executable software deployments can be extracted by specializing the models. Z3 is used for exploring such models, and also for specializing the models.

References

[1] L. de Moura and N. Bjørner, " Z3: An Efficient SMT Solver," in *TACAS 08*, ser. Lecture Notes in Computer Science, C. R. Ramakrishnan and J. Rehof, Eds., vol. 4963. Springer, 2008.