*Software Component Technology Group*

# Master Project

# Counterexample Execution

Jürg Billeter
Supervisors: Prof. Peter Müller, Joseph N. Ruskiewicz

February 2008

**Introduction** Boogie is a static program verifier that can verify programs written in Spec#[2], an extension to C# to include non-null types, method contracts, and object invariants. An automatic theorem prover proves the correctness of the program or finds potential errors in it.

Understanding the verification errors that Boogie reports can be very complex. However, developers can understand errors very well when they can use a debugger interface.

Z3[3] is an efficient SMT theorem prover that has been integrated with Spec# and Boogie. It uses a modern DPLL-based SAT solver, which makes it possible to return concrete counterexamples.

**Goal of this master's project** is to make it easier to understand and locate errors reported by Boogie. Instead of relaying only on static verification, we create a runtime state from the counterexample returned by the Z3 theorem prover. The developer will be able to run the method in the debugger step-by-step from the beginning to the failing condition.

This will provide several benefits to developers. When Boogie reports a verification error, the developer no longer has to manually write test code to ensure that it's a real error. The exact values from the error model in the counterexample are used to construct a runtime state of relevant objects. Running the method in the familiar debugger interface makes it easy to understand the error reported by Boogie.

Check 'n' Crash[1] is a similar project that uses static checking and automatic testing to find possible runtime exceptions in Java, as for example invalid casts, null dereferences, and divisions by zero. We want to take this further and verify the specified method pre- and postconditions.

Spec# uses contracts to verify method calls. Instead of executing real objects, we use mock objects that behave according to the contracts.

**The main parts** of this project are:

1. Design and implement a system to create mock objects with specific behavior
2. Use the Z3 error model of counterexamples to create mock objects
3. Support runtime debugging of verification errors as returned by Boogie

**Possible extensions** include:

1. Extend the Spec# compiler to report errors instead of warnings for counterexamples that have been verified by execution
2. Automatically generate test cases from counterexamples
3. Feed spurious counterexamples back to prover to improve verification

### Timeline

| | |
|---|---|
| 2 weeks | Get to know Spec# and Boogie |
| 2 weeks | Implement prototype for very simple methods |
| 4 weeks | Design and implement library to inject specific behavior into existing methods |
| 4 weeks | Use the Z3 error model to specify behavior |
| 4 weeks | Generate valid values for behavior not specified in the model |
| 4 weeks | Support debugging of verification errors |
| 6 weeks | Write project report |

# References


[1] Christoph Csallner and Yannis Smaragdakis. Check 'n' Crash: combining static checking and testing. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 422–431, New York, NY, USA, 2005. ACM.

[2] Microsoft Research. Spec# programming system. http://research.microsoft.com/specsharp/.

[3] Microsoft Research. Z3: Theorem prover. http://research.microsoft.com/projects/z3/.