Automatic Verification of Concurrent Programs
Project Description

Concurrent applications have been in the focus of research since Peterson's mutual exclusion algorithm and Dijkstra's Semaphore abstraction. Most concurrent programs to this day on still use a combination of semaphores and condition variables, which guarantee mutual exclusion. This simple framework can often lead to two of the most common bugs which only happen in a concurrent environment: Data Races and Deadlocks.

In order to eliminate these common bugs, the Chalice Programming language was created. Chalice is a research programming language which is able to statically proof the absence of data races and deadlocks in a concurrent program. Additionally Chalice provides dynamic checking of contracts, similar to these provided in other programming languages such as Eiffel or Spec#.

The main goal of the project is to explore automatic verification of concurrent programs, while exploring existing software technologies and methodologies. The main focus will be on the Chalice programming language, determining which concurrency patterns Chalice is not yet able to successfully verify.

The concurrency patterns will most likely include the Composite pattern and Priority Inheritance Protocol. Both patterns have been proved to be complex in the setting of verification and concurrency. The implementation and verification of these patterns in Chalice will most likely show the current limits of the verification technique and will be a good starting point to explore possible extensions of the Chalice Programming language.

The core of the project includes defining a specification for concurrent version of the Composite Pattern and implementing the specification in Chalice such that the program verifies, proving the correctness, absence of data-races and deadlocks. The core of the project also includes research into multi granularity locking and the application of these in Chalice.

The extensions of the project include formally extending Chalice methodology for multi granularity locking; implementing this methodology into the current Chalice compiler and verifier; Specifying and implementing the Priority Inheritance Protocol in Chalice; Research how the new specification can help in verifying other programming patterns.