Diplomarbeit

Static Detection of Higher Level Synchronization Conflicts
(Race Conditions)

Background

The Bauhaus Project develops methods and tools to support and ease program understanding in the process of software maintenance. Since the founding in 1996, an extensive infrastructure was built that allows the execution of program analyses with low and high levels of abstraction.

The IML (InterMediateLanguage) is a detailed, programming language independent representation for programs in the form of an enhanced syntax tree. Manifold program analyses are performed on the basis of this representation, e.g., control flow and data flow analyses. The IML-library is implemented in Ada95 and provides operations for the generation, manipulation and traversal of the IML.

Task

Data races are a common problem in multi-threaded programming. The classical notion (low-level) of data races can only detect and report unprotected accesses to a particular shared variable. However, this notion is not powerful enough to capture certain types of inconsistencies occurring in practice, where some system properties must be achieved through a combined access on several shared variables. Such situations require a new notion (high-level) of data races on a higher abstraction level. This enables detection of inconsistent uses of shared variables, even if no classical (low-level) race condition occurs. For example, a data structure representing a coordinate pair may have to be treated atomically. By lifting the meaning of a data race to a higher level, such problems can be covered and detected.

This thesis will investigate high-level data races in parallel programs through static analysis, and describe what kind of errors can be detected with this new definition. Further goals are to find the limitations of high-level notion of data races and give a formal guideline for using data structures in a multi-threaded environment.

Those are the tasks that must be accomplished:

- A literature search for existing algorithms for high-level race conditions detection must be done.
- Implementation of method described in [Athro] to statically detect combined accesses to the fields of shared ADT, further, a strategy should be devised on the basis of the available SSA implementation in Bauhaus to argue about atomicity violations for shared data structures.
• Upon detection of an error, diagnostics should inform how actual execution can reach the potential error situations.

• A high-level May Happen in Parallel (MHP) analysis shall be implemented to supported the validation of detected error scenarios.

• A theoretical evaluation of runtime behaviour of the algorithms must be performed.

• The algorithm must be implemented using the provided tools and APIs.

• Extensive tests must be performed to show the correctness of the implementation. The analysis must be applied to several large programs. The tutor will provide some of these programs, but the student is also obligated to search for his own test programs. Further, the student will receive prepared test programs where races are hidden. The analysis should be able to detect all of these race conditions. The student is also encouraged to implement his own suite of test cases for the complete test of all features of the analysis.

• The precision of the analysis must be evaluated. Further strategies to improve the results should be investigated.

The requirements for the quality of the implementation and reasonable code documentation are high. The rules of the Bauhaus Coding style for Ada programs must be observed.

Available tools and programming language

Bauhaus tool suite, APIs and libraries of Bauhaus, GNAT Ada95 compiler

Tutor

Aoun Raza (1.218)
phone: +49-711-7816-399
aoun.raza@informatik.uni-stuttgart.de