Influence of data dependencies in concurrent software on execution paths

In the multicore era, in which we are currently living, software consists of dependent threads that execute concurrently. Input data and the interaction between concurrent threads determine the result of the concurrent software. Testing of concurrent software is a challenging task. It is necessary to create test cases with a range of input data, in order to test individual software threads. Further, it is necessary to test interaction between threads. However, due to the nature of multicore hardware (e.g. shared cache, shared memory bus); it is hard to predict order of interactions between threads, as they do not progress uniformly. Forcing different orders of interaction requires an intense work and due to the number of possible orderings, is not an efficient approach.

There are two major issues with concurrent threads. Concurrent threads interact over shared memory. If threads do not synchronize their operations on shared memory, the outcome could be invalid or overwritten data. Threads can also influence execution paths of each other, over data dependencies. Some bugs only originate due to execution paths that appear because of mutual dependency between threads, and it is not possible to reproduce these bugs by only changing the input data. There is a gap in the state of the art, regarding the second point.

One way to test concurrent software is to execute software, record execution trace, and apply different algorithms on the execution trace. The main objective of this thesis is to find a relation between data dependencies of concurrent threads, and their influence on the execution paths of concurrent threads. Suggested technique is symbolic execution. Student should design and program a solution that analyses structure of software and execution trace. For given execution traces, the solution should conclude if threads could influence execution path of each other. If yes, the solution should list the points of influence and the propagation of dependencies. It is necessary to make an evaluation, using real world, concurrent software frameworks (e.g. SPLASH-2). Student will work with already existing software framework infrastructure that provides execution traces and structure of software.

Requirements:

- Strong C/C++, JAVA development skills.
- Ability to work alone, self-initiative, self-motivation, research-oriented mind.

Beneficial, but not mandatory:

- Understanding of formal testing methods.
- Understanding of compilers.
- Experience with LLVM compiler infrastructure.

M.Sc Jasmin Jahić
Engineer at Embedded Software Engineering
Fraunhofer IESE
Fraunhofer-Platz 1 | 67663 Kaiserslautern
Tel: +49 631 / 6800-2252 | Fax: +49 631 / 6800-9 2252
Mail: jasmin.jahic@iese.fraunhofer.de