Author:John Self

Background and Problems

The topic: Understanding Concurrent Software

The goal: Understanding concurrent program executions and structure.

The problem: How to map from the series of low level events that constitute a concurrent program execution to a high level understanding or program structure and behavior.

My answer: Static Concurrency Analysis (SCA) as an aid to Dynamic Concurrent Program Understanding.

When debugging a concurrent program the common approach is to record concurrency events for later examination. While this allow the userto determine the behavior of a program on a particular execution, it does little to go a deeper understanding of the program as a whole.

SCA examines the static structure of a concurrent program and determines whether certain classes of faults could occur. Thus it provides information about the program as a whole. Unfortunately, complete SCA is intractable,and often impossible for large programs that use dynamic features.Thus it fails when it is most needed.

What is needed is a technique that combines the applicability to large problems found in dynamic debugging with the high level understanding provided by SCA.

Title: Static Concurrency Analysis as an Aid to Concurrent Program Understanding

Hypothesis and Insights

Insight: SCA often produces spurious error reports, necessitating manual inspection of the reports. This activity often provides experienced SCA users with insight into the program's concurrency structure.

Hypothesis: The Task Interation Concurrency Graph (TICG) generated by static concurrency analysis tools is a useful visualization of programconcurrency structure, independent of its use as an internal rtifact.

Insight: Often when there is one particular series of concurrency actions that lead to a faulty final state, there will be anotherseries of events leading either to the same failed state, or to a similar failed state.

Hypothesis: The ability to generate and examine the portion of the TICG close to a particular failed state may help us discover otherfaults that we might miss given only knowledge about a particular execution.

Approach and Validation

Nutshell: Build a system that allows the user to view the trace collected at runtime as a partial TICG, then interactively expand and examine this partial TICG.

Method: Build a prototype that allows the user to:

•Collect traces describing the execution of a concurrent program and map these to the equivalent partial TICG**.**

•Explore the partial TICG to see what actions are possible at each state, and to incrementally expand the TICG while looking for paths to the same or similar "interesting" states.

Validation:Apply the resulting prototype to several systems that are known to have bugs. Determine how the techniques enabled by this system could be used to debug these and similar faults.

The Chiron-1 UIMS provides a ready source of understood bugs that were very difficult to find using conventional debugging techniques.

Contributions and Schedule

Expected contributions:

1. A technique that combines SCA with traditional trace based debugging.
2. Extending SCA to handle dynamic features that are typically beyond its scope.

Schedule:

Prototype: Partial tools already running, to be completed in February, 1996.

Validation: Try methodology on large examples. Late February, early March 1996.

Write: FSE paper and disseration, March and April 1996.