Fehlersuche beim Debuggen nebenläufiger Programme

Frank Huch
Institut für Informatik und Praktische Mathematik
Christian-Albrechts-Universität zu Kiel
fhu@informatik.uni-kiel.de

Nebenläufige Systemen können neue, bei sequentiellen Systemen nicht auftretende Fehler enthalten, wie z.B. Deadlocks, Lifelocks oder die Nichteinhaltung von wechselseitigem Ausschluß. Im Gegensatz zu klassischen Fehlern sequentieller Programme, hängen diese Fehler vom Prozeß-Scheduling ab und treten möglicherweise nur in bestimmten Systemläufen auf.

Zum Finden von Fehlern werden in der Regel Debugger verwendet, welche den Programmablauf visualisieren und so ein besseres Verständnis der Fehlersituation ermöglichen sollen. Wir haben einen Debugger für Concurrent Haskell (eine nebenläufige Erweiterung der lazy, funktionalen Programmiersprache Haskell) entwickelt. Dieser ermöglicht es nebenläufige Haskell Programme auszuführen und manuell unterschiedliche Prozeß-Scheduls auf Fehler zu überprüfen.

Ein Nachteil dieses Ansatzes ist, daß sich ein System beispielsweise kurz vor eine Deadlock befinden kann, der Benutzer aber ein falsches Scheduling wählt und so den Programmfehler nicht findet. Als Lösung für dieses Problem, soll der Debugger (alle) möglichen Schedules bis zu einer vorgegebenen Tiefe simulieren und im Fehlerfall den Benutzer in den Fehler leiten können. Da nicht nur Deadlocks mögliche Fehler sind, soll die Linearzeit Logik (LTL) zur Spezifikation allgemeiner Fehlerzustände verwendet werden.

Im Concurrent Haskell Debugger läuft das nebenläufige System im Hintergrund ab und wird zusätzlich visualisiert. Zur Implementierung der Fehlersuche müssen wir es ermöglichen, daß durchgeführte Aktionen von Concurrent Haskell rückgängig gemacht werden und so ein alter Systemzustand wieder hergestellt wird. Dies ist natürlich nur begrenzt möglich, da z.B. das Schreiben in Dateien kaum rückgängig gemacht werden kann. Für umkehrbare Aktionen kann aber das Durchsuchen der Folgezustände mittels Backtracking implementiert werden und Fehlerzustände können mit Hilfe eines LTL Checkers automatisch gefunden werden.

Das System befindet sich noch in der Entwicklung. In diesem Vortrag präsentieren wir den aktuellen Entwicklungsstand des Projekts und gehen insbesondere darauf ein, wie das Backtracking für den Systemablauf implementiert werden kann.