Christian-Albrechts-Universität zu Kiel
Institut für Informatik

Priv.-Doz. Dr. Frank Huch, Marius Rasch

Institut
für Informatik

9. Übung zur Vorlesung „Concurrent and Distributed Programming“

Abgabe am Monday, 10. June 2019 - 18:00

Aufgabe 1 - Chan implementation in Haskell STM

1 Punkt

In an early lecture, we implemented an unbounded channel in Python. This implementation had a bug with respect to `isEmpty` and `unGet`.

1. Also Concurrent Haskell (without STM) provides such Channels. Check, whether the predefined `Chan` implementation in Haskell also has this bug.
2. Transfer the `Chan` implementation from Python to Haskell STM, by reusing the STM version of the `MVar`s from the lecture. Does your implementation still contain the bugs for `isEmpty` and `unGet`? Explain, why.

Aufgabe 2 - Testing STM

1 Punkt

Implement an Erlang application, which can be used to test different STM implementations. The number of threads, number of TVars and the number of read and write accesses (and other usefull parameters you might think of) should be set by using parameters. Think of a way to check the correctness of an execution (e.g. by checking the sum of all TVar values at the end).

Extend the STM implementation from the lecture by a monitoring, which is able to protocol the number and order of the transitions, as well as all rollbacks and retrys that take place.

Try to force an interessting behaviour with your application and watch the result.

Aufgabe 3 - Shorten lock times

1 Punkt

In the STM implementation from the lecture we locked all TVars in the validating/commit phase and freed them afterwards. Think of a way to shorten the time, in which all TVars ar locked.

Modifiy the implementation from the lecture by using your idea and compare your implemenation with the original one by using tests. Do you notice a difference in the number of rollback happening? Can you measure a difference in the lock times?