Transactional Memory with Finalizers

Michael Schröder

Vienna University of Technology

Abstract

Software Transactional Memory (STM) immensely simplifies concurrent programming by allowing memory operations to be grouped together into atomic blocks. Like database transactions, STM transactions provide atomicity, consistency and isolation. Unlike databases, they do not provide durability.

As part of my master's thesis, I have extended the STM implementation of Haskell with a mechanism to add finalizers to atomic blocks. The new operation `atomicallyWithIO` allows the programmer to safely execute arbitrary I/O during the commit-phase of a transaction. This can be used to make STM durable, but it turns out to have even more applications. For example, a finalizer could ask the user to approve pending results, enabling interactive transactions. A finalizer could also communicate with remote actors, providing the foundation for a distributed STM system.

In this talk I will give a brief overview of STM in Haskell, motivate the need for finalizers, provide examples of their use, and discuss design decisions and potential shortcomings.