Software Transaction Roll Back Avoidance

Master Proposal Talk ???

Lasse Folger

November 24, 2016

Motivation

```
type Account = MVar Int

transfer :: Account -> Account -> Int -> IO ()
transfer src dst am = do
  a1 <- takeMVar src
  a2 <- takeMVar dst
  putMVar src (a1 - am)
  putMVar dst (a2 + am)
```

MVar

Thread 1:

transfer acc1 acc2 50

Thread 2:

transfer acc2 acc1 50

MVar

Thread 1:

transfer acc1 acc2 50

Thread 2:

transfer acc2 acc1 50

$\Rightarrow$ Deadlock

Use Transactions

```
type Account = TVar Int

transfer :: Account -> Account -> Int -> STM ()
transfer src dst am = do
  a1 <- readTVar src
  a2 <- readTVar dst
  writeTVar src (a1 - am)
  writeTVar dst (a2 + am)
```

TVar

Thread 1:

atomically $ transfer acc1 acc2

Thread 2:

atomically $ transfer acc2 acc1

TVar

Thread 1:

```
atomically $ transfer acc1 acc2
```

Thread 2:

```
atomically $ transfer acc2 acc1
```

$\Rightarrow$ works fine, because transactions provide ACID properties

Current Implementation (Control.Concurrent.STM)

- *writeTVar*, *readTVar* and *newTVar* modify TVars
- *retry* and *orElse* alter the control flow
- *atomically* executes a transaction
- composition via bind operator (or do)

Validation

- validation is performed before commiting or when the thread is dispatched
- validation: version numbers from TVars are matched with the read version numbers.
  - validation fails $\Rightarrow$ restart transaction
  - validation succeeds $\Rightarrow$ continue transaction
- For validation locking is needed

Write Set

- local workspace
- writesTVars cannot to be published directly
- allow to read written values to provide compositionality
- in the comit phase these values are published

Problems

Thread 1:
a1 $\leftarrow$ readTVar acc1
a2 $\leftarrow$ readTVar acc2
writeTVar acc1 (a1 - 50)
writeTVar acc2 (a2 + 50)

Thread 2:
a1 $\leftarrow$ readTVar acc2
a2 $\leftarrow$ readTVar acc1
writeTVar acc2 (a1 - 50)
writeTVar acc1 (a2 + 50)

Problems

Thread 1:
a1 $\leftarrow$ readTVar acc1
a2 $\leftarrow$ readTVar acc2
writeTVar acc1 (a1 - 50)
writeTVar acc2 (a2 + 50)

Thread 2:
a1 $\leftarrow$ readTVar acc2
a2 $\leftarrow$ readTVar acc1
writeTVar acc2 (a1 - 50)
writeTVar acc1 (a2 + 50)

$\Rightarrow$ either sequential or one transaction is rolled back

Idea

- delay the evaluation of readTVar to the comit phase
- this avoids rollbacks...
- ..but no longer allows to access the value directly

Idea

```
type Account = TVar Int

transfer :: Account -> Account -> Int -> STM ()
transfer src dst am = do
  writeTVar src ((readTVar src) - am)
  writeTVar dst ((readTVar dst) + am)
```