Department of Information and Computer Science
Utrecht University

INFOB3CC: STM

Trevor L. McDonell

December 5, 2021

Introduction

Use these tasks to practice the topics of the lectures. You may have to do some research to read up on terms or topics not (yet) covered in the lectures.

Questions

1. Properties of software transactional memory.
   (a) What is the purpose of the function `atomically :: STM a -> IO a`?
   (b) Explain why operations in software transactional memory are defined using the type `STM`.
   (c) Which of these properties hold for the atomic blocks of software transactional memory? Explain why:
   - Mutual exclusion
   - Deadlock freedom
   - No starvation

2. Software transactional memory compared to locks.
   (a) What is an advantage of using atomic blocks (software transactional memory) over programming with explicit locks?
   (b) Give a simple example program which is easier to implement using software transactional memory compared to using explicit locks.
   (c) Give an example program which could be easier to implement using explicit locks compared to software transactional memory.

3. Implementation of software transactional memory.
   (a) Describe briefly how the `atomically` operation works.
   (b) What are the performance considerations for `writeTVar`?
   (c) What are the performance considerations for `readTVar`?

4. In Haskell the `MVar` lock implements fairness by queuing up threads blocked on the `MVar` in first-in-first-out order. A thread trying to read a value from an empty `MVar` will be queued waiting for a corresponding `putMVar`, and a `putMVar` on a full `MVar` will be queue waiting for a corresponding `takeMVar`.

   Suppose we want to implement fairness in `STM` in the same way for `TMVar`. Consider the following implementation of `TMVar`, which explicitly keeps track of the blocked `takeTMvar` and `putTMVar` operations:

```
data TMVar a = TMVar
       (TVar (Maybe a))          -- as before, the TMVar may be empty or full
       (TVar [TVar (Maybe a)]) -- list threads blocked waiting to put or take a value
```

   Will this implementation work? Consider the cases for how `putTMVar` should be implemented.

5. Characteristics of software transactional memory.
   (a) When writing code using STM, you cannot perform side effects in `IO`, such as reading or writing files. Why is this restriction needed?
   (b) How does STM guarantee the property of mutual exclusion?
   (c) Does STM guarantee the absence of starvation? Explain why or why not.
   (d) In some situations STM transactions can be slow. Give an example where this can occur, and explain why this happens.
6. Disgruntled with the limitations of software transactional memory, Felix makes his own version which includes the possibility to read and write files on disk. Writing to files is directly executed, and if the transaction fails, the operation is reverted by rewriting the original contents of the file back. Will this approach work? If so motivate your answer, or if not explain why or describe a problem which can be encountered.
7. Follow the tutorial *Beautiful Concurrency*. If you have any questions, ask the TAs in the working group session.

   `https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/beautiful.pdf`