STM implementation:

- optimistic execution (so fare)

  record all read and written TVars, when transaction is over, validate read set and commit write set

- pessimistic execution

  directly lock TVars, when they are read (or perhaps also wrote), so no other transaction can read or write them.

readTVar t1 lock t1
readTVar t2 lock t2
writeTVar t3 v lock here or in the end

readTVar t2 lock t2
readTVar t1 lock t1 -> deadlock possible
deadlock can be avoided by releasing all locks, if one lock cannot be acquired.

The implementation can result in a life lock, where two transactions (like the ones above) try to read/write the same resource in differenet order. So it might be the case, that both hold one resource and want to access the ressource held by the other transaction and then rollback both.

Comparison of optimistic and pessimistic implementation:

- lifelock in pessimistic approach can be problematic
- perhaps some more rollbacks in the pessimistic approach?
- usually more concurrency in the optimistic approach

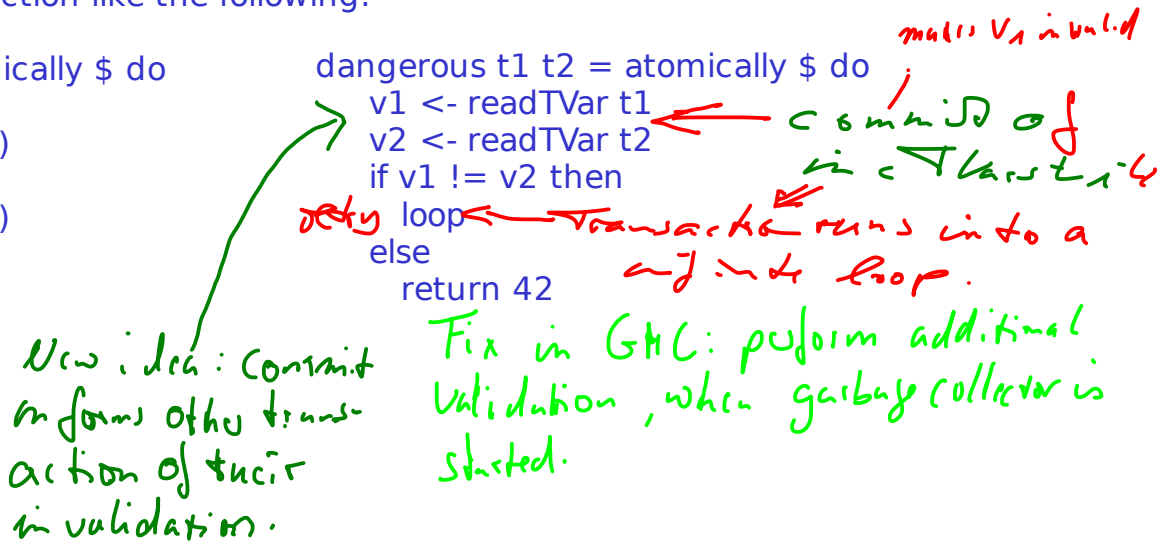
Problem in the optimistic implementation:

Consider a situation in which two TVars t1 and t2 always contain the same values.

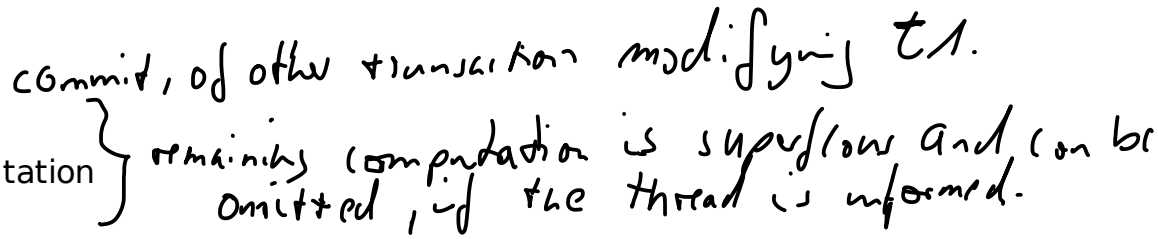
Lets consider a transaction like the following:

```
incTVars t1 t2 = atomically $ do
    v1 <- readTVar t1
    writeTVar t1 (v1+1)
    v2 <- readTVar t2
    writeTVar t2 (v2+1)
```

```
dangerous t1 t2 = atomically $ do
    v1 <- readTVar t1
    v2 <- readTVar t2
    if v1 != v2 then
      loop
    else
      return 42
```




```
readTVar t1
readTVar t2
writeTVar t3
complex computation
writeTVar t4
...
```



Implementation:

- readTVar and writeTVar should always check, if the transaction was invalidated by another commit. The same for atomically, before it performs a commit.
- Validation is not necessary anymore, because a transaction is informed if it gets invalid.
- Inside the TVars, we store information about all Transactions which read this TVar, similar to the RS, but distributed.
- When writing a new value to TVar, all non-committed transaction have to be informed about their invalidation. Be carefull, that an invalidation information registered too late.