State & StateT Monads (9A)

Young Won Lim
9/20/18

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using LibreOffice.

Young Won Lim
9/20/18

Based on

Haskell in 5 steps

https://wiki.haskell.org/Haskell_in_5_steps

State Monad

execState
:: State s a **state-passing computation to execute**
-> s **initial value**
-> s **final state**

Evaluate a state computation with the given initial state and return the final state, discarding the final value.

execState m s = snd (runState m s)

mapState :: ((a, s) -> (b, s)) -> State s a -> State s b

Map both the return value and final state of a computation using the given function.

runState (mapState f m) = f . runState m

withState :: (s -> s) -> State s a -> State s a

withState f m executes action m on a state modified by applying f.

withState f m = modify f >> m

https://hackage.haskell.org/package/mtl-2.2.2/docs/Control-Monad-State-Lazy.html

StateT Monad Transformer

newtype StateT s (m :: * -> *) a

A state transformer monad parameterized by:

s - The state.
m - The inner monad.

The return function leaves the state unchanged,
while >>= uses the final state of the first computation
as the initial state of the second.

Constructors
StateT (s -> m (a, s))

https://hackage.haskell.org/package/mtl-2.2.2/docs/Control-Monad-State-Lazy.html

StateT Monad Transformer

runStateT :: StateT s m a -> s -> m (a, s)

evalStateT :: Monad m => StateT s m a -> s -> m a

Evaluate a state computation with the given initial state and return the final value, discarding the final state.

```
evalStateT m s = liftM fst (runStateT m s)
```

execStateT :: Monad m => StateT s m a -> s -> m s

Evaluate a state computation with the given initial state and return the final state, discarding the final value.

```
execStateT m s = liftM snd (runStateT m s)
```

https://hackage.haskell.org/package/mtl-2.2.2/docs/Control-Monad-State-Lazy.html

StateT Monad Transformer

mapStateT :: (m (a, s) -> n (b, s)) -> StateT s m a -> StateT s n b

Map both the return value and final state of a computation using the given function.

runStateT (mapStateT f m) = f . runStateT m

withStateT :: (s -> s) -> StateT s m a -> StateT s m a

withStateT f m executes action m on a state modified by applying f.

withStateT f m = modify f >> m

https://hackage.haskell.org/package/mtl-2.2.2/docs/Control-Monad-State-Lazy.html

References

[1] ftp://ftp.geoinfo.tuwien.ac.at/navratil/HaskellTutorial.pdf

[2] https://www.umiacs.umd.edu/~hal/docs/daume02yaht.pdf