Advanced Programming (I00032)
A shallow embedded DSL for Crane Control

Assignment 12

Goal

The purpose of this assignment is to understand the possibilities and limitations of a shallow embedded DSL. For this purpose we implement the DSL controlling the same physical device as last week.

Crane Control

To load the containers on and off a ship we have a harbour crane controlled by a DSL. For simplicity, we assume there is only a single stack of containers on the ship and a single stack on the quay. These stacks are able to accommodate any number containers. The crane is able to execute the following actions encoded as a datatype.

```
:: Action
  = MoveToShip                  // move the crane to the ship
  | MoveToQuay                  // move the crane to the quay
  | MoveUp                      // moves the crane up
  | MoveDown                    // moves the crane down
  | Lock                        // locks the top container of the stack under the crane
  | Unlock                      // unlocks the container the crane is carrying, put it on the stack
  | Wait                        // do nothing
  | (:.) infixl 1 Action Action // sequence of two actions
  | WhileContainerBelow Action  // repeat action while there is a container at current position
```

The following program loads all containers from the quay on the ship, given that the crane is initially up, does not carry a container and is above the quay.

```
loadShip =
    whileContainerBelow (
        MoveDown:.
        Lock:.
        MoveUp:.
        MoveToShip:.
        Wait:.
        MoveDown:.
        Wait:.
        Unlock:.
        MoveUp:.
        MoveToQuay
    )
```

Most actions should only be applied in specific states. Some violations of this rule are pretty harmless, e.g., `MoveToShip` when the crane is already above the ship, but can be considered as a no-operation. Other actions are very dangerous, like unlocking a container while the crane is `Up`, or moving the crane horizontally when it is `Down`. Finally, there are

actions that are impossible to execute, for instance `Unlock` when the crane is not carrying a container or `Lock` when the crane is up, already carrying a container, or the stack is empty.

Shallow Embedding

Make a shallowly embedded version of the crane control DSL that is able to execute the actions on a state. Try to ensure by the type system the most important safety conditions:

1. the crane can only move to the ship or the quay when it is `High`;
2. the crane can only lock and unlock containers when it is `Down`;
3. the crane can only lock a container when it is `Free`;
4. the crane can only unlock when it is carrying a container: `Container`.

The state from the previous assignment is a useful starting point for the state in this assignment.

```
:: State =
    { onShip      :: [Container]
    , onQuay      :: [Container]
    , craneUp     :: Bool
    , craneOnQuay :: Bool
    , locked      :: Maybe Container
    }
:: Container :== String

initialState =
    { onShip      = []
    , onQuay      = ["apples","beer","camera's"]
    , craneUp     = True
    , craneOnQuay = True
    , locked      = Nothing
    }
```

Review

Note that optimizing a composed action and showing actions without executing them is at least very troublesome. In the deep embedded version of this DSL adding these views of the DSL was just defining other functions over the same datatype `Action`.

On the other hand, adding an action can be done in the shallow embedding without touching any existing code or risks that other code will break. For instance, we can add an instruction `Park` that moves the crane to the quay and down from any other position.

Deadline

The deadline for this assignment is Thursday, June 3 at 13:30h. This is just before the next tutorial. It might be convenient to finish this assignment before the next lecture on Monday May 31.