Kelley Lynch

Introduction

For my final project developed a continuized gesture grammar in Haskell. The gesture grammar consists of Locations, Blocks, and Gestures. Currently the gesture available is Move'. The Move' gesture supports two forms of input. Input is either a start location and an end location or a block and an end location. If a block and an end location are provided, the first step is to convert the block into a start location. Next I check to see if the (start, end) location pair is valid. The validity of the movement is checked using Continuation Passing Style semantic types. Blocks are converted to Locations before the validity of a command containing a block is checked. Locations are entities which are raised to type (e->t)->t. The Move' gesture takes two entities so the non-continuized type of Move' is e->e->t. The continuized form has type (e->e->t)->t and the computation form has type ((e->e->t)->t)->t. The types of the gesture move' and the entity location cannot be combined, so instead I used the cps-function application procedure (van Eijck and Unger).

```
cpsApply m n = \ k -> n (\ b -> m (\ a -> k (a b)))
```

When a command is evaluated, after the types have been raised, the gesture is applied to the current state of the block simulation. The function for the gesture then generates a list of tuples representing valid parameters to the function. For the gesture Move', this is done by zipping the list of all occupied grid locations and all unoccupied

grid locations. The gesture is then combined with the parameters and if the command is valid, the command is executed.

Executing a command involves using the state monad to manipulate the state of the current blocks.

Block Simulation

To start a block simulation, run the main function from the CPSS.hs module.The current state of the blocks is modeled by a list of tuples containing a block and that blocks location.

The initial configuration of the blocks is hard coded.

[((Block 1), (A', 2)), ((Block 2), (B', 3)), ((Block 3), (D', 5))]

This list represents Block 1 being at location A2, Block 2 being at location B3 and Block 3 being at location D5.

The main function will first display the current configuration of blocks, then the user can enter a command. Commands must have the form “Location Move Location” or “Block # Move Location”.

For example, from the initial configuration, the command “Block 1 Move A1” results in the block configuration.

[((Block 1), (A', 1)), ((Block 2), (B', 3)), ((Block 3), (D', 5))]

Following that command, it is now possible to move a block into the position that was previously occupied by Block 1. Next, issuing the command “B3 Move A2” will result in the following configuration, with Block 2 in the position previously occupied by Block 1.

[((Block 1), (A', 1)), ((Block 2), (B', 2)), ((Block 3), (D', 5))]