# Design document

## Implementation

My implementation revolvs around the functions:

- interpret – Takes a grid and a robot and returns final position of robot
- evalStmt – Takes a state and a statement and returns state after statement has been evaluated
- evalExp – Takes a buffer and an expression, returns an int

The “state” type is a type made to keep track of the position and variables of the robot, state consists of a tuple (grid * position * buffer). The buffer is the type that keeps track of what variables have been assigned the robot, the buffer consists of a list of variables which in turn consists of a tuple (string * exp). The buffer has two methods which are used to set and get variables to/from it:

- buf_set – Takes a buffer and a variable and returns a new buffer with the variable added to the front of the old buffer, where the old buffer has been filtered for any duplicate variables
- buf_get – Takes a buffer and a key and returns the expression that the key is associated with.

In the code I have included the four test-programs that where given as examples in the oblig-text. To run my oblig simply type “sml robol.sml” and to try it out, type “interpret testX;” with 'X' being replaced by a number from 1-4.