**Comp 317: Semantics of Programming Languages**

# Sixth Tutorial Sheet

1. Recall the module `NUMERAL-EXPRESSION` from the Fifth Problem Sheet:

```
fmod NUMERAL-EXPRESSION is

  sorts Digit NumeralExp .
  subsort Digit < NumeralExp .

  ops 0 1 : -> Digit .

  op _ _ : NumeralExp Digit -> NumeralExp .

  op _+_ : NumeralExp NumeralExp -> NumeralExp .

endfm
```

   Add equations to this module that define addition (cf. Exercise 5 from Problem Sheet 1).

2. Play Spot the Model with these equations. You might like to use this model:

   - $B_{\texttt{Digit}} = \{true, false\}$
   - $B_{\texttt{NumeralExp}} = \{true, false\}$
   - $B_{\texttt{0}} = false$
   - $B_{\texttt{1}} = true$
   - $B_{\texttt{\_\_}}(x, y) = x \text{ and } y$
   - $B_{\texttt{+}}(x, y) = x \text{ or } y$

   from Problem Sheet 5, but make up your own model as well.

Continued...

3. Simplify the following (use Maude to check your answers):

   (a) `[[ 'x + 1 ]](initial)`

   (b) `[[ 2 * 'x ]](initial)`

   (c) `[[ 2 * 'x ]]([[ 'x := 1 ]](initial))`

   (d) `[[ 2 * 'x ]]([[ 'y := 1 ]](initial))`

   (e) `[[ 2 * 'x ]]([[ 'x := 'x + 1 ]](initial))`

4. Simplify the following, where `s` is an arbitrary store:

   (a) `[[ 'x + 1 ]](s)`

   (b) `[[ 2 * 'x ]]([[ 'x := 1 ]](s))`

   (c) `[[ 2 * 'x ]]([[ 'y := 1 ; 'x := 'y + 1 ]] (s))`

   (d) `[[ 'x ]]([[ 'x := 'x + 'y ; 'y := 'x - 'y ; 'x := 'x - 'y ]](s))`

   You can check your answers in Maude by adding an arbitrary store as follows:

```
th  SOME-STORE  is

  including STORE .

  op  s  : -> Store .

endth
```

   and then doing the reductions.