Lecture Notes 4

Dynamic Semantics

- Dynamic Semantics – Meaning of expressions and statements of programming language
- Operational Semantics – Describes meaning by specifying effects of running it on a machine.
  - Natural Operational Semantics – Final result of running complete program.
  - Structural Operational Semantics – Complete sequence of state changes that occur when program is executed.
  - Intermediate Language – Lower level languages (above machine level) that describes operation of language constructs
- Denotational Semantics – Formal Method for describing meaning of programs
  - Syntactic Domain – Domain of the denotational semantic functions
  - Semantic Domain – Range of the denotational semantic functions
  - Binary Example

    ```
    <bin_num> →'0' | '1'| <bin_num> '0' | <bin_num> '1'

    M_bin('0') = 0
    M_bin('1') = 1
    M_bin(<bin_num> '0') = 2 * M_bin(<bin_num>)
    M_bin(<bin_num> '1') = 2 * M_bin(<bin_num>) + 1
    ```
  - Program State Example

    ```
    s = {<i_1,v_1>, <i2,v_2>, ..., <i_n,v_n>}
    VARMAP(i_j,s) = v_j
    ```
    Where `s` is the state of the program, the `i`'s are variable names, and `v`'s are values (including **`undef`** meaning that variable name is undefined)
  - Expression Example

    ```
    <expr> → <dec_num> | <var> | <bin_expr>
    <bin_expr> → <l_expr> <op> <r_expr>
    <l_expr> → <dec_num> | <var>
    <r_expr> → <dec_num> | <var>
    <op> → + | *

    M_e(<expr>,s)Δ =
          case <expr> of
            <dec_num> => M_dec(<dec_num>,s)
            <var> => if VARMAP(<var>,s) == undef
                         then error
                         else VARMAP(<var>,s)
            <bin_expr> =>
              if(M_e(<bin_expr>.<l_expr>,s) == undef OR
                 M_e(<bin_expr>.<r_expr>,s) == undef)
    ```

```
                    then error
                    else if(<bin_expr>.<op> == '+')
                      then M_e(<bin_expr>.<l_expr>,s) +
                           M_e(<bin_expr>.<r_expr>,s)
                      else M_e(<bin_expr>.<l_expr>,s) *
                           M_e(<bin_expr>.<r_expr>,s)
```

- Assignment Example

```
M_a(x=E,s)Δ =
    if M_e(E,s) == error
       then error
       else s' = {<i_1,v_1'>, ..., <i_n,v_n'>} where
          for j = 1, ..., n
             if i_j == x
                then v_j' = M_e(E,s)
                else v_j' = VARMAP(i_j,s)
```

- While Loop Example – Assumes M_b and M_{sl} already defined

```
M_w(while B do L,s)Δ =
    if M_b(B,s) == undef
        then error
        else if Mb(B,s) == false
           then s
           else if M_sl(L,s) == error
              then error
              else M_w(while B do L,M_sl(L,s))
```

- Axiomatic Semantics – Predicate calculus that specifies the constraints of each statement
- Predicate (or assertion) – Boolean statement that expected to be true for a correct program
- Precondition – Assertion that is expected to be true prior to the statement for correct program
- Postcondition – Assertion that is expected to be true after the statement for correct program
- Weakest Precondition – The least restrictive precondition to guarantee the post condition is true
- Inference Rule – Method of inferring truth of an assertion on the basis of other assertions
  $\frac{S_1, S_2, \ldots, S_n}{S}$ If S_1, S_2, …, S_n (the antecedent) are true then S (the consequent) is true
- Axiom – Logical statement assumed to be true
- Assignment Statement Axiom – $P = Q_{x \to E}$ where P is the weakest precondition of assignment x = E and Q is the post condition of assignment.
  - Example determining precondition with `x > 8` as postcondition:

```
x = y * 2 - 4 {x > 8}
{y * 2 - 4 > 8}
{y * 2 > 12}
{y > 6}
```

```
{y > 6} x = y * 2 - 4 {x > 8}
```

- Rule of Consequences – Essentially states that preconditions can always be strengthen or postconditions can be weakened
  $$\frac{\{P\}S\{Q\}, P' => P, Q => Q'}{\{P'\}S\{Q'\}}$$
- Sequence Inference Rule – P_1 is precondition of statement S_1, P_2 is the postcondition of S_1 and precondition of S_2, and P_3 is the post condition of S_2.
  $$\frac{\{P_1\}S_1\{P_2\}, \{P_2\}S_2\{P_3\}}{\{P_1\}S_1, S_2\{P_3\}}$$
- Selection Inference Rule – Assume form **if** B **then** S_1 **else** S_2 with P precondition and Q postcondition.
  $$\frac{\{B \text{ and } P\}S_1\{Q\}, \{(\text{not } B) \text{ and } P\}S_2\{Q\}}{\{P\}\textbf{if } B \textbf{ then } S_1 \textbf{ else } S_2\{Q\}}$$
- Pretest Loop Inference Rule – Assume form **while** B **do** S **end** with P precondition and Q postcondition, and I being the loop invariant.
  $$\frac{\{I \text{ and } B\}S\{I\}}{\{I\}\textbf{while } B \textbf{ do } S \textbf{ end}\{I \text{ and } (\text{not } B)\}}$$

```
{P} while B do S end {Q}
P => I
{I and B} S {I}
(I and (not B)) => Q
```

  - Total Correctness – Loop termination can be proved
  - Partial Correctness – Conditions are met, but termination is not guaranteed
  - Predicate Transformer – Function that returns the weakest precondition given a statement and postcondition
  - Example for loop with postcondition `y = x`:

```
while y <> x do y = y - 1 end {y = x}
wp(y = y - 1,{y = x}) = {y - 1 = x} or {y = x + 1}
wp(y = y - 1,{y = x + 1}) = {y = x + 2}
wp(y = y - 1,{y = x + 2}) = {y = x + 3}
{y > x},{not y <> x}
{y > x},{y = x}
{y >= x}
```