The Simple Programming Language

A Simple Language

- The Simple Language is based upon the *Unbounded Register Machine* of Shepherdson and Sturgis (JACM 10, 1963)
- It is a very simple machine, but retains features we recognize from modern traditional languages
- It is constructed to be Turing Complete
- We can construct “macros” that encode more sophisticated features that help us picture how complicated algorithms can be expressed.

Features

- Variables that take on the Natural numbers (0,1,2,3,…) as values.
- Simple assignment statements
  - x := 0, x:= succ(y),   x := pred(y)
  - Assign values to variables but also perform only the most primitive kind of computation.
- Control
  - Sequencing { s1; s2; s3 }
  - While loop

Macros

- More complicated statements can be built from combining the simple statements in a algorithmic way.
- The idea is to build up a library of “macros” that allow us to write high level programs that “macro-expand” intoi the simple language.

Assignment of one variable to another

- y : = y

{x := succ(y);
x := pred(x)}

Assignment of a constant

- X := 4

{x := 0;
 x := succ(x);
 x := succ(x);
 x := succ(x);
 x := succ(x)}

Addition

- X := X + Y

```
{i_0 := succ(y);
 i_0 := pred(i_0);
 while i_0 =/= 0
    {x := succ(x);
     i_0 := pred(i_0)}}
```

Note the temporary, or local, variable i_0

Multiplication

- X := X * Y

```
{x_0 := succ(x);
 x_0 := pred(x_0);
 y_1 := succ(y);
 y_1 := pred(y_1);
 ans_2 := 0;
 while x_0 =/= 0
    {i_3 := succ(y_1);
     i_3 := pred(i_3);
     while i_3 =/= 0
        {ans_2 := succ(ans_2);
         i_3 := pred(i_3)};
     x_0 := pred(x_0)};
 x := succ(ans_2);
 x := pred(x)}
```

```
mult x y =
  do { i <- gensym "x“
     ; j <- gensym "y“
     ; ans <- gensym "ans"
      ; seq' [varAssign i x
              , varAssign j y
              , assign ans Zero
              , while i  (seq' [add ans j
                                , assign i (Pred i)])
             , varAssign x ans]}
```

Factorial

```
{count_0 := succ(n);
 count_0 := pred(count_0);
 ans_1 := 0;
 ans_1 := succ(ans_1);
 while count_0 =/= 0
    {x_2 := succ(ans_1);
     x_2 := pred(x_2);
     y_3 := succ(count_0);
     y_3 := pred(y_3);
     ans_4 := 0;
     while x_2 =/= 0
        {i_5 := succ(y_3);
         i_5 := pred(i_5);
         while i_5 =/= 0
            {ans_4 := succ(ans_4);
             i_5 := pred(i_5)};
         x_2 := pred(x_2)};
     ans_1 := succ(ans_4);
     ans_1 := pred(ans_1);
     count_0 := pred(count_0)};
 n := succ(ans_1);
 n := pred(n)}
```

Notes

- Large things can be constructed from many small things.
- The introduction of new “local” variables is crucial, but is possible since the model does not limit how many variables there are.
- The new variables must be “fresh”
- While the results are larg (and perhaps slow) we are only interested in what we can express.