CSCE 330 Fall 2000

Example of Denotational Semantics for a Program with a While Loop

Wednesday 00/9/13

The main reference for these notes is Section 9.4 of: Ghezzi, Carlo and Mehdi Jazayeri. *Programming Language Concepts, 2nd ed.*, New York: John Wiley and Sons, 1987.

Consider the following program P:

```
read(n); fact := 1; i := 1;
while i <= n do
  fact := fact * i;
  i := i + 1;
od;
write(fact);
```

If the input stream is empty, then
$dsem_{PROG}(P, empty) =$
$out(dsem_S L(read(n); fact := 1; i := 1; \text{while i <= n do } fact := fact * i; i := i + 1; od; write(fact)), init(empty)) =$
$out(dsem_{SL}(fact := 1; i := 1; \text{while i <= n do } fact := fact * i; i := i + 1; od; write(fact),$
$dsem_{RD}(read(n), init(empty)))) =$
$out(dsem_{SL}(fact := 1; i := 1; \text{while i <= n do } fact := fact * i; i := i + 1; od; write(fact), error)) =$
$out(error) = error.$

Now, assume that the input stream is not empty and it consists of the integer z. Then,
$dsem_{PROG}(P, < z >) =$
$out(dsem_{SL}(read(n); fact := 1; i := 1; \text{while i <= n do } fact := fact * i; i := i + 1; od; write(fact)), init(< z >)) =$
$out(dsem_{SL}(fact := 1; i := 1; \text{while i <= n do } fact := fact * i; i := i + 1; od; write(fact)),$
$dsem_{RD}(read(n), init(< z >))) =$
$out(dsem_{SL}(fact := 1; i := 1; \text{while i <= n do}$

$fact := fact * i; i := i + 1; od; write(fact)), < mem1, empty, empty >)$,
where $mem1 = \{< n, z >, < fact, undef >, < i, undef >\}$.

The two assignments change the state in such a way that:

$dsem_{PROG}(P, < z >) =$
$out(dsem_{SL}($while i <= n do $fact := fact * i; i := i + 1; od; write(fact)), < mem2, empty, empty >)$,
where $mem2 = \{< n, z >, < fact, 1 >, < i, 1 >\}$.

Let us now use the semantics of the statement list one more time:

$dsem_{PROG}(P, < z >) =$
$out(dsem_{SL}(write(fact), dsem_{DO}($while i <= n do $fact := fact * i; i := i + 1; od, < mem2, empty, empty >))$.

Now, we concentrate on the *while* loop:

$dsem_{DO}($while i <= n do $fact := fact*i; i := i+1; od, < mem2, empty, empty >) =$
$if dsem_{BOOL}(i <= n, < mem2, empty, empty >) = false$
$then < mem2, empty, empty >$
$else\ dsem_{DO}($while i <= n$do\ fact := fact * i; i := i + 1; od, dsem_{SL}(fact := fact * i; i := i + 1; od; write(fact), < mem2, empty, empty >)) =$
$if dsem_{BOOL}(i <= n, < mem2, empty, empty >) = false$
$then < mem2, empty, empty >$
$else\ dsem_{DO}($while i <= n do $fact := fact*i; i := i+1; od, < mem3, empty, empty >$, where
$mem3 = \{< n, z >, < fact, mem2(fact)*mem2(i) >, < i = mem2(i)+1 >\}$.

The last equation above defines recursively a function $dsem_{DO}$ from S to $S \cup \{error\}$. The result is the *fixpoint* of the function. We do not cover the theory of recursive functions in this course, but it should be evident that in this particular case, when the stopping condition $(i > n)$ holds, $fact = n!$ and therefore

$dsem_{DO}($while i <= n do $fact := fact*i; i := i+1; od, < mem2, empty, empty >) =$
$< mem4, empty, empty >$, where
$mem4 = \{< n = z >, < fact = z! >, < i = z + 1 >\}$.

We therefore conclude that

$dsem_{PROG}(P, < z >) = out(dsem_{WR}(write(fact), < mem4, empty, empty >) =$
$out(< mem4, empty, < z! >) = z!$