Formal Semantics of Programming Languages
Exercise 3 (January 12)

Wolfgang Schreiner
Wolfgang.Schreiner@risc.uni-linz.ac.at

November 30, 2006

The exercise is to be submitted by **January 12** (hard deadline)

1. as a single PDF file sent to me per email, or
2. as a paper report (cover page with full name and Matrikelnummer, pages stapled) handed out to me in class.

1 A Language with Heap Allocation

Extend the language of Figure 7.2 presented in class to include constructs for the allocation and use of *heap* cells:

C ::= . . . | **alloc** I | ∗I := E
E ::= . . . | ∗I

The command "**alloc** I" allocates a cell on the heap and writes its location (a "pointer") into the variable *I*. The command "∗I = E" assigns a value to the heap cell referenced by I; the expression "∗I" reads the content of this cell.

Please note that the command "I := . . . " essentially retains its original meaning (assigning a value to the variable I) as well as the expression "I" (reading the content of the variable I). Thus, if J is a variable holding a pointer, the statement "I := J" denotes the assignments of this pointer to I.

The heap can be represented as a fixed amount of memory before the bottom of the stack. Thus use the initial memory location passed to the program for reserving some memory for the heap. If the heap is full, the allocation of a new memory cell triggers an error.

Variables can now hold either natural numbers (as in the original language) or pointers, thus the domain *Storable-value* (introduced in Figure 5.5) has to be appropriately extended and also the dynamic type checks have to be adapted.