T-501-FMAL Programming languages, Practice class 10
Spring 2021

1. Consider the simple imperative language from Imp.fs.

   Extend the language of commands with a *repeat-until* command that checks the guard at the end of each iteration (rather than at the beginning of each iteration like *while-do*).

   Accordingly extend the interpreter.

2. Change the interpreter to account for an alternative meaning for the *for-do* command where the stop expression is evaluated at the beginning of each iteration rather than only once at the very beginning of the loop.

   Construct some programs on which the two versions of the *for-do* command behave differently.

   (You can also consider a meaning where the loop control variable is incremented at the end of each loop iteration wrt. the value it has acquired by that moment. In the version implemented, the loop control variable is incremented as compared to the value it had at the beginning of the iteration; the updates to it during the iteration are disregarded.)

3. Consider the imperative language from MicroC.fs.

   Add a C-style *for-do* command form, permitting for instance

   ```
   for (i=0; i<100; i=i+1) sum = sum+i
   ```

   The general form of such a loop construct should be

   ```
   for (e1; e2; e3) stm
   ```

   That command should be equivalent to

   ```
   e1; while (e2) {stm; e3}
   ```

   Extend the interpreter accordingly.

4. Extend the language with an expression form `e?e1:e2` for an *if-then-else* construct to choose between two expressions rather than commands. Extend the interpreter accordingly. `e?e1:e2` must first evaluate `e` and then `e1` or `e2` depending on the value of `e`. Differently from the command `if e then e1 else e2`, which does not return anything, the expression `e?e1:e2` must return the value of `e1` or `e2`.

5. Add also an accessor form `e?a1:a2` where `a1`, `a2` are accessors. (You need to invent a different tag for it, since there is no tag overloading in F#). Extend the interpreter accordingly.

   The idea is that, eg, `(x<y)?z:w = 4` should become a valid expression (and command).

6. Add expression forms `++a` ("pre-increment", i.e., increment, then return the value) and `a++` ("post-increment", return the value and only then increment) where `a` is an accessor, and extend the interpreter.