Programming Languages (COP 4020/6021) [Spring 2019]

Assignment VI

Objectives

1. To demonstrate an understanding of evaluation contexts.
2. To implement an interpreter for diML+P based on evaluation contexts.

Due Date: Wednesday, April 24, 2019 (at 5pm).

Machine Details: Complete this assignment by yourself, the programming portion on the following CSEE computers: c4lab01, c4lab02, ..., c4lab19. These machines are located in ENB 216. You can connect to the C4 machines from home using SSH. (Example: Host name: c4lab01.csee.usf.edu Login ID and Password: <your NetID username and password>) You are responsible for ensuring that your programs compile and execute properly on these machines.

Assignment Description

First, correct any problems with your implementation of the `sub` function from Assignment III. Then, in a directory containing a copy of *as3.sml*, begin a new file called *as6.sml* with the command `use "as3.sml";`. Then implement the following in *as6.sml*.

(1) Exception and datatype declarations for diML+P evaluation contexts:

```
exception stuck;
datatype ec = Hole | IfEC of ec*expr*expr
  | PlusEC1 of ec*expr | PlusEC2 of int*ec
  | LessEC1 of ec*expr | LessEC2 of int*ec
  | ApplyEC1 of ec*expr | ApplyEC2 of string*typ*typ*((pattern*expr) list)*ec
```

You can just copy-paste this code into your *as6.sml* file.

(2) `isValue : expr -> bool`

This function takes a diML+P expression *e* and returns true iff *e* is a value.

(3) `fill : ec -> expr -> expr`

This function takes a diML+P evaluation context *E* and expression *e* and returns *E[e]*.

(4) `decompose : expr -> ec * expr`

This function takes an expression *e* and returns an *(E,e')* such that *e=E[e']* and *e'* can take a beta step. If no such *(E,e')* exists, then this function raises the *stuck* exception.

(5) `beta : expr -> expr`

This function returns the result of β-stepping its argument; if no β-step is possible, *stuck* is raised.

(6) `smallStep : expr -> expr`

This function returns the result of stepping its argument; if no step is possible, *stuck* is raised. This small-step operation must be defined in terms of evaluation contexts (as discussed in class).

(7) `bigStep : expr -> expr`

This function returns the value resulting from fully evaluating its argument expression *e*. If *e* gets stuck, this function raises *stuck*, and if *e* diverges, so does this function. This big-step operation must be defined in terms of small-step operations.

Notes: Assume that all variable names have been chosen to avoid capture; hence, `sub` from Assignment III can be used to perform substitutions. Also, every one of the 6 functions (numbered (2)-(7)) should be legitimately invoked somewhere in `smallStep` or `bigStep`.

Hints: My *as6.sml* is 53 lines of code (not counting comments and whitespace) and took about an hour to implement and test.

Sample Executions

```
- use "as6.sml";
...
- use "exprs.sml"; (* using http://www.cse.usf.edu/~ligatti/pl-19/as3/exprs.sml *)
...
- val pe = (PlusExpr(IntExpr 4,ApplyExpr(ApplyExpr(mult,IntExpr 2),IntExpr 2)));
val pe =
  PlusExpr
    (IntExpr 4,
     ApplyExpr
       (ApplyExpr
          (FunExpr
             ("mult",Int,Arrow (Int,Int),
              [(VarPattern "n",
                FunExpr
                  ("multN",Int,Int,
                   [(IntPattern 0,IntExpr 0),
                    (VarPattern "m",
                     PlusExpr
                       (VarExpr "n",
                        ApplyExpr
                          (VarExpr "multN",PlusExpr (VarExpr "m",IntExpr ~1))))])])),
           IntExpr 2),IntExpr 2)) : expr
- decompose pe;
val it =
  (PlusEC2 (4,ApplyEC1 (Hole,IntExpr 2)),
   ApplyExpr
     (FunExpr
        ("mult",Int,Arrow (Int,Int),
         [(VarPattern "n",
           FunExpr
             ("multN",Int,Int,
              [(IntPattern 0,IntExpr 0),
               (VarPattern "m",
                PlusExpr
                  (VarExpr "n",
                   ApplyExpr
                     (VarExpr "multN",PlusExpr (VarExpr "m",IntExpr ~1))))])])),
      IntExpr 2)) : ec * expr
- fill (#1 it) (#2 it);
val it =
  PlusExpr
    (IntExpr 4,
     ApplyExpr
       (ApplyExpr
          (FunExpr
             ("mult",Int,Arrow (Int,Int),
              [(VarPattern "n",
                FunExpr
                  ("multN",Int,Int,
                   [(IntPattern 0,IntExpr 0),
                    (VarPattern "m",
                     PlusExpr
                       (VarExpr "n",
                        ApplyExpr
                          (VarExpr "multN",PlusExpr (VarExpr "m",IntExpr ~1))))])])),
           IntExpr 2),IntExpr 2)) : expr
- bigStep pe;
val it = IntExpr 8 : expr
```

```
- smallStep pe;
val it =
  PlusExpr
    (IntExpr 4,
     ApplyExpr
       (FunExpr
          ("multN",Int,Int,
           [(IntPattern 0,IntExpr 0),
            (VarPattern "m",
             PlusExpr
               (IntExpr 2,
                ApplyExpr (VarExpr "multN",PlusExpr (VarExpr "m",IntExpr ~1))))]),
        IntExpr 2)) : expr
- smallStep (smallStep it);
val it =
  PlusExpr
    (IntExpr 4,
     PlusExpr
       (IntExpr 2,
        ApplyExpr
          (FunExpr
             ("multN",Int,Int,
              [(IntPattern 0,IntExpr 0),
               (VarPattern "m",
                PlusExpr
                  (IntExpr 2,
                   ApplyExpr
                     (VarExpr "multN",PlusExpr (VarExpr "m",IntExpr ~1))))]),
           IntExpr 1))) : expr
- smallStep it;
val it =
  PlusExpr
    (IntExpr 4,
     PlusExpr
       (IntExpr 2,
        PlusExpr
          (IntExpr 2,
           ApplyExpr
             (FunExpr
                ("multN",Int,Int,
                 [(IntPattern 0,IntExpr 0),
                  (VarPattern "m",
                   PlusExpr
                     (IntExpr 2,
                      ApplyExpr
                        (VarExpr "multN",PlusExpr (VarExpr "m",IntExpr ~1))))]),
              PlusExpr (IntExpr 1,IntExpr ~1))))) : expr
- smallStep it;
val it =
  PlusExpr
    (IntExpr 4,
     PlusExpr
       (IntExpr 2,
        PlusExpr
          (IntExpr 2,
           ApplyExpr
             (FunExpr
                ("multN",Int,Int,
                 [(IntPattern 0,IntExpr 0),
                  (VarPattern "m",
                   PlusExpr
                     (IntExpr 2,
                      ApplyExpr
                        (VarExpr "multN",PlusExpr (VarExpr "m",IntExpr ~1))))]),
              IntExpr 0)))) : expr
- smallStep it;
val it =
  PlusExpr (IntExpr 4,PlusExpr (IntExpr 2,PlusExpr (IntExpr 2,IntExpr 0)))
  : expr
- smallStep it;
val it = PlusExpr (IntExpr 4,PlusExpr (IntExpr 2,IntExpr 2)) : expr
```

```
- smallStep it;
val it = PlusExpr (IntExpr 4,IntExpr 4) : expr
- smallStep it;
val it = IntExpr 8 : expr
- smallStep it;

uncaught exception stuck
  raised at: as6.sml:40.25-40.30
- bigStep e3;
val it = IntExpr 120 : expr
- decompose e3;
val it =
  (Hole,
   ApplyExpr
     (FunExpr
        ("factorial",Int,Int,
         [(IntPattern 0,IntExpr 1),
          (VarPattern "x",
           ApplyExpr
             (ApplyExpr
                (FunExpr
                   ("mult",Int,Arrow (Int,Int),
                    [(VarPattern "n",
                      FunExpr
                        ("multN",Int,Int,
                         [(IntPattern 0,IntExpr 0),
                          (VarPattern "m",
                           PlusExpr
                             (VarExpr "n",
                              ApplyExpr
                                (VarExpr "multN",
                                 PlusExpr (VarExpr "m",IntExpr ~1))))])])]),
                 VarExpr "x"),
              ApplyExpr
                (VarExpr "factorial",PlusExpr (VarExpr "x",IntExpr ~1))])]),
      IntExpr 5)) : ec * expr
- e3 = (fill (#1 it) (#2 it));
val it = true : bool
```

Grading

For full credit, your implementation must:

- be commented and formatted appropriately (as on previous assignments).
- use ML features like pattern matching when appropriate.
- compile on the C4 machines with no errors or warnings.
- not use any ML features that cause *side effects* to occur (e.g., I/O or references/pointers).
- not use built-in/library functions.
- not define extra top-level values.
- not be significantly more complicated than is necessary.
- be reasonably efficient

As always, we will test submissions on inputs not shown in the sample executions above.

Submission Notes

The submission process is the same as for other programming assignments, except here you'll submit as6.sml in Canvas. Please remember to include the pledge as an initial comment; not doing so will lower your grade 50%. As usual, you may submit this assignment up to 48 hours late with a 15% penalty.