Programming Languages (COP 4020/CIS 6930) [Fall 2014]

Assignment VI

Objectives

1. To gain experience programming with recursively defined data types in ML.
2. To demonstrate an understanding of diML+P static semantics by implementing a type checker.
3. To demonstrate an understanding of diML+P dynamic semantics by implementing an interpreter.

Due Date: Sunday, October 26, 2014, at 11:59pm.

Machine Details: Complete this assignment by yourself on the following CSEE network computers: c4lab01, c4lab02, ..., c4lab20. These machines are physically located in the Center 4 lab (ENB 220). Do not use any server machines like grad, babbage, sunblast, etc. 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 IV. Then, in a directory containing a copy of *as4.sml*, begin a new file called *as6.sml* with the command `use "as4.sml";`. Then implement the following values in *as6.sml*.

(1) `tc : expr -> typ option`

This function takes a diML+P expression *e* and returns NONE iff *e* is an ill-typed program and SOME *t* iff *e* is a well-typed program having type *t*. Please note that in diML+P, a function having non-exhaustive pattern(s) is ill typed, so `tc` must check exhaustiveness of function patterns. Also, functions having no body (i.e., `FunExpr(f,t1,t2,[])`), and functions having incompatible body patterns (e.g., `FunExpr(f,t1,t2,[(TruePattern,_), (FalsePattern,_), (IntPattern i,_)])`), are ill typed.

(2) `exception stuck;`

All you need to do for this step is to declare the exception "stuck". Please read Section 5.2 of the *Elements of ML Programming* textbook for details on using exceptions in ML.

(3) `eval : expr -> expr`

This function takes a diML+P expression *e* and evaluates *e* for as many steps as possible. If evaluation of *e* converges to a value *v*, then *eval(e)* returns *v*; if *e* diverges then so does *eval(e)*. Function *eval* must raise exception *stuck* at any point that evaluation gets "stuck" without a value being produced (but note that because diML+P is type safe, only ill-typed expressions can get stuck before becoming values).

Throughout this assignment, you may assume that all variable names in expressions being type checked and evaluated are unique, so you never have to alpha-convert expressions.

Hints: My *as6.sml* is 79 lines of code (not counting comments and whitespace) and took about 3 hours to implement and test.

Sample Executions

```
- use "as6.sml";
...
- use "exprs.sml"; (* using http://www.cse.usf.edu/~ligatti/pl-14/as4/exprs.sml *)
...
- (tc e1, tc e2, tc e2bad);
val it = (SOME Bool,SOME (Arrow (Int,Arrow (Int,Arrow (Int,Int)))),NONE)
  : typ option * typ option * typ option
- tc (PlusExpr(IntExpr(4),ApplyExpr(ApplyExpr(mult,IntExpr 5),IntExpr(6))));
val it = SOME Int : typ option
- eval (PlusExpr(IntExpr 4,ApplyExpr(ApplyExpr(mult,IntExpr 5),IntExpr 6))); (* 4+5*6 *)
val it = IntExpr 34 : expr
- tc (ApplyExpr(ApplyExpr(e2bad,IntExpr 6), IntExpr 7));
val it = NONE : typ option
- eval (ApplyExpr(ApplyExpr(e2bad,IntExpr 6), IntExpr 7)); (* ill typed=>may get stuck *)

uncaught exception stuck
  raised at: as6.sml:62.31-62.36
- eval e3;  (* e3 computes 5 factorial *)
val it = IntExpr 120 : expr
- let (* test pattern compatibility and exhaustive coverage *)
    val f1 = FunExpr("f",Bool,Int,[])
    val f2 = FunExpr("f",Bool,Int,[(VarPattern "x",IntExpr 4),(TruePattern,IntExpr 3)])
    val f3 = FunExpr("f",Int,Int,[(VarPattern "x",IntExpr 4),(TruePattern,IntExpr 3)])
    val f4 = FunExpr("f",Bool,Int,[(VarPattern "x",IntExpr 4),(TruePattern,IntExpr 3),
                                    (IntPattern 5,IntExpr 6)])
    val f5 = FunExpr("f",Int,Int,[(IntPattern 4,IntExpr 5),(WildcardPattern,IntExpr 8),
                                   (FalsePattern,IntExpr 6)])
    val f6 = FunExpr("f",Int,Int,[(IntPattern 4,IntExpr 5),(WildcardPattern,IntExpr 8),
                                   (IntPattern 4,IntExpr 6)])
    val f7 = FunExpr("f",Int,Int,[(IntPattern 4,IntExpr 5),(IntPattern 6,IntExpr 7),
                                   (IntPattern 8,IntExpr 9)])
    val f8 = FunExpr("f",Int,Int,[(WildcardPattern,IntExpr 5),
                                   (WildcardPattern,IntExpr 8),(IntPattern 4,IntExpr 6)])
    val f9 = FunExpr("f",Bool,Int,[(FalsePattern,IntExpr 4),(TruePattern,IntExpr 3)])
  in (tc f1, tc f2, tc f3, tc f4, tc f5, tc f6, tc f7, tc f8, tc f9,
    eval(ApplyExpr(f6,IntExpr 4)), eval(ApplyExpr(f6,IntExpr 5)))
  end;
val it =
  (NONE,SOME (Arrow (Bool,Int)),NONE,NONE,NONE,SOME (Arrow (Int,Int)),NONE,
   SOME (Arrow (Int,Int)),SOME (Arrow (Bool,Int)),IntExpr 5,IntExpr 8)
  : typ option * typ option * typ option * typ option * typ option *
    typ option * typ option * typ option * typ option * expr * expr
```

Grading

For full credit, your implementation must:

- be commented and formatted appropriately (as on previous assignments).
- use ML features like pattern matching when appropriate.
- not define extra top-level values.
- 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 be significantly more complicated than is necessary.

Please note that we will test submissions on inputs not shown in the sample executions above.

Submission Notes

- Type the following pledge as an initial comment in your *as6.sml* file: “I pledge my Honor that I have not cheated, and will not cheat, on this assignment.” Type your name after the pledge. Not including this pledge will lower your grade 50%.
- Upload and submit your *as6.sml* file in Canvas.
- You may submit your assignment in Canvas as many times as you like; we will grade your latest submission.
- For every day that your assignment is late (up to 3 days), your grade reduces 10%.