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

Assignment IV

Objectives

1. To gain experience programming with recursively defined data types in ML.
2. To demonstrate an understanding of static semantics by implementing a type checker.
3. To demonstrate an understanding of dynamic semantics by implementing an interpreter.
4. To gain experience setting up deductive systems.
5. To formalize static and dynamic semantics for a new programming language.

Due Date: Monday, March 18, 2019 (at 5pm).
Late submission: You may submit any part (or both parts) of this assignment late (i.e., between 5pm on 3/18 and 5pm on 3/20) with a 15% penalty on the whole assignment.

Machine Details: *Complete this assignment by yourself*, the programming portion on the following CSEE network computers: c4lab01, c4lab02, ..., c4lab20. 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

(1) **Programming portion:** 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 *as4.sml* with the command `use "as3.sml";` and then implement the following values.

(a) `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.

(b) `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.

(c) `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 *as4.sml* is 79 lines of code (not counting comments and whitespace) and took about 3 hours to implement and test.

Sample Executions

```
- use "as4.sml";
...
- use "exprs.sml"; (* using http://www.cse.usf.edu/~ligatti/pl-19/as3/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: as4.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 and Submission Notes

For full credit, your implementation must (i) be commented and formatted appropriately (as on previous assignments), (ii) use ML features like pattern matching when appropriate, (iii) not define any extra top-level values, (iv) compile and run on the C4 machines with no errors or warnings (except for stuck exceptions raised at appropriate times), (v) not use any library functions, (vi) not use any ML features that cause side effects to occur (e.g., I/O or references/pointers), (vii) not be significantly more complicated than is necessary, and (viii) be reasonably efficient.

The submission process is the same as for Assignment III, except here you'll submit *as4.sml* in Canvas. Please remember to include the pledge as an initial comment; not doing so will lower your grade 50%.

(2) **Theory portion:**

Recall the following language L from Assignment III:
types τ ::= bool | $\tau_1 \times \tau_2$
exprs e ::= x | true | false | e_1 NOR e_2 | (e_1,e_2) | let val (x_1,x_2) = e_1 in e_2 end

This language contains variables (x), true and false literals, logical-NOR expressions, binary tuples, and let-expressions. Let-expressions in L have the same meaning as in ML, except that L's let-expressions always declare a pair of variables.

Define static and dynamic semantics for L. Assume that all variable names in all expressions under consideration have been made unique through alpha-conversion; hence, you never have to consider contexts containing more than one entry for the same variable. Also, assume that capture-avoiding substitution ([e/x]e') is already defined for L, so you can just use that notation ([e/x]e') without defining it.

As always, avoid making the definitions significantly more complicated than necessary. If you get stuck at any point, please explain in prose whatever you're having trouble formalizing.

Theory-portion Submission Reminders

- Write the following pledge at the end of your submission: "I pledge my Honor that I have not cheated, and will not cheat, on this assignment." Sign your name after the pledge. Not including this pledge will lower your grade 50%.
- For full credit, turn in a hardcopy (handwritten or printed) version of your solutions.
- Late submissions may be emailed or submitted in hardcopy.
- All emailed submissions, even if sent before the deadline, will be graded as if they were submitted late, i.e., with a 15% penalty.
- If you think there's a chance you'll be absent or late for class on the date this assignment is due, you're welcome to submit solutions early by giving them to me or the TA before or after class, or during any of our office hours.