Programming Languages (COP 4020/6021) [Spring 2017]

Assignment III

Objectives

1. To become familiar with recursive data types in ML.
2. To understand basic definitions related to variables in programming languages: free variables, capture-avoiding substitution, and alpha equivalence.
3. To gain experience with deductive systems and inference rules.

Due Date: Monday, February 27, 2017 (at 5pm).
Late submission: You may submit any part (or both parts) of this assignment late (i.e., between 5pm on 2/27 and 5pm on 3/1) 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. These machines are physically located in 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

(1) **Programming portion:** Let's consider a language called diMLazy. The datatypes defining diMLazy are:

```
(* diMLazy types *)
datatype typ = Bool | Int | Arrow of typ*typ; (* i.e., Arrow(argType, returnType) *)

(* diMLazy expressions *)
datatype expr = TrueExpr | FalseExpr | IntExpr of int
 | PlusExpr of expr*expr | LessExpr of expr*expr
 | IfExpr of expr*expr*expr (* i.e.: IfExpr(condition, thenBranch, elseBranch) *)
 | VarExpr of string | ApplyExpr of expr*expr
 | FunExpr of string*string*typ*typ*expr;
 (* i.e., FunExpr(functionName, parameterName, parameterType, returnType, body) *)
```

Create a new file called *as3.sml*, and begin that file with the datatypes given above. Then implement the following values in *as3.sml*.

(a) `isFV : expr -> string -> bool`
This function returns true iff the given expression has a free variable with the given name.

(b) `exception Capture`
This exception is raised whenever a substitution is attempted that could result in variable capture. Exceptions are described in Section 5.2 of the textbook.

(c) `sub : expr -> string -> expr -> expr`
This function takes an expression e, variable name x, and expression e', and returns [e/x]e'. Exception Capture must be raised whenever the substitution isn't defined because a variable could be captured.

(d) `uniquifyVars: expr -> expr` [Note: This function is 5% extra credit for undergraduates.]
This function takes an expression e and returns an alpha-equivalent expression e' such that e' never declares two variables having the same name (all variables declared in e' must be uniquely named). You can assume that no function in the argument e has the same name as its parameter.

Hints: My *as4.sml* is 57 lines of code (not counting comments and whitespace) and took about 2-3 hours to implement and test.

Grading and Submission Notes
For full credit, your implementation must (1) compile and execute on the C4 machines with no errors/warnings (though the sub function may raise Capture exceptions), (2) contain no side effects, (3) not define any top-level values beyond the functions and exception described in this handout, (4) not use any library (i.e., built-in) functions other than map, foldl, foldr, and Int.toString, (5) be commented and formatted properly (as part of this restriction, use spaces instead of tabs and limit line widths to 100 characters), (6) use ML constructs like pattern matching and anonymous variables when appropriate, (7) not be significantly more complicated than necessary, and (8) be reasonably efficient.

The submission process is the same as for Assignment II, except here you'll submit *as3.sml* in Canvas. Please remember to include the pledge as an initial comment; not doing so will lower your grade 50%.

(2) **Theory portion:**
Consider the following language L:

expressions e ::= x | n | S(e) | let val x = e in e' end

This language contains variables (x), natural numbers (n), successor expressions, and let-expressions. Let-expressions in L have the same meaning as let-expressions in ML. For example, the program:

let val x = let val x = 4 in S(x) end in let val x = S(x) in S(x) end end

is equivalent to the following (alpha-converted) program.

let val x = let val y = 4 in S(y) end in let val z = S(x) in S(z) end end

Both of these example programs evaluate to 7.

Provide definitions for: (a) free variables in expressions in L, (b) capture-avoiding substitution in L, and (c) alpha-equivalence of expressions in L.

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.