

Assignment 3

CSE 755

Due: November 3, by 11:00 am

1. (10 pts)

Consider the Lisp interpreter discussed in class. Write the full definitions of functions `bound`, `getval`, and `addpairs`. Use Lisp notation — i.e., `(DEFUN BOUND . . .)`. You can assume that the preconditions listed in the lecture notes are valid. The first parameter of `bound` is a literal atom, and the second parameter is a (possibly empty) list of `(x . y)` S-expressions, where `x` is a literal atom. The precondition of `getval` is that `bound` had returned true. The precondition of `addpairs` is that the first two parameters are two lists with the same length. Your code does not have to handle cases when the preconditions are not true.

2. (5 pts)

Consider the following expression in the language defined during the discussion of type systems:

```
if
  (if true then true else false)
then
  (if true then false else true)
else
  (if true then false else false)
```

Show the sequence of derivation steps (based on the small-step operational semantics for this language, as discussed in class) that describe how this expression gets evaluated. Make sure to explicitly show each rule instance being applied, and indicate clearly what are its premises and conclusion.

3. (5 pts)

Consider the example from the previous question. Show how this expression gets typed based on the type system discussed in class. Explicitly show each instance of a type rule being applied, and indicate clearly what are its premises and conclusion.

4. (10 pts)

Consider the following expression from the same language

```
let
  x = (let y = {p=0,q=true} in y.q)
in
  (let z = 0 in {r={s=x,t=z},v={z,x}})
```

Is this expression typable in the type system discussed in class? If not, prove that it is not. Otherwise, show how it gets typed. You *must* show explicitly each instance of a type rule being applied, and indicate clearly what are its premises and conclusion.

5. (10pts)

Consider the following pairs of types from the type system discussed in class. Indicate if the first type is a subtype of the second, the second is a subtype of the first, both, or neither. Explain briefly the reasoning behind your answer.

- | | | |
|-----|----------------------------------|---------------------------|
| (a) | Nat | Bool |
| (b) | Nat | {l:Nat} |
| (c) | {l:Nat} | {l:Nat} |
| (d) | {m:Nat} | {m:Bool} |
| (e) | {a:Nat,b:Bool} -> {a:Bool} | {a:Nat} -> {a:Bool,c:Nat} |
| (f) | {a:Nat,b:Bool} -> {c:Nat,d:Bool} | {a:Nat} -> {c:Nat} |
| (g) | {a:Nat,b:Bool} | {a:Nat} |