

# Homework # 7

## due October 25

### 1 Reading

Please read Chapters 13 and 14 in your textbook.

### 2 Problems

Please do the following problem:

- Exercise 13.1.1, and include the effects of evaluating

```
(lambda x: Nat . {ref x, ref x}) 0
```

Explain your answer!

### 3 Counter Examples

The textbook says (page 167) that we must include a store typing in the requirements for progress and preservation. Give two counter-examples, one for Progress (Theorem 13.5.7) and one for Preservation (Theorem 13.5.3) if they omit any mention of store typing.

### 4 Exceptions

Suppose we wished to implement `raise` in the simply-typed lambda-calculus. As with `cons` in Chapter 11, we provide an annotation giving the type of the value carried, for instance `raise[Nat] 3`. We would also annotate `try` clauses so they would catch only a particular type of errors:

```
try t with x:T => t'
```

Give the evaluation rules, types rules and prove progress (modified to permit `raise` normal forms) and preservation. You need only give the changes in the lemmas and theorems from the simply-typed lambda calculus (no references!).

### 5 Discussion

The book argues for the need of  $\Sigma$ , a store typing. Where does this come from; how does one create a store typing? In particular, for a programmer wanting to know whether their program will execute without getting stuck, how can they use the progress and preservation theorems? The theorems require that we have a store typing, what store typing should one use?