Assignment 3: μScheme

CS - 456 – Programming Languages

September 25, 2014

General Information

- Due Date: 10/09/2014 by midnight.
- Submission: See the instructions at the end of the document. Multiple submissions are possible before the deadline. Only the last one will be considered.
- All assignments are *individual*.
- Use Piazza for Q&A.

Programming Assignment: 50%

1. Solve the following exercises of Chapter 3:

   (1-2) S-Expr: It is recommended to solve Exercise 1 before solving 2. (pg. 154)

   (5) Recursive functions on Lists.

   (14) Let: add your answer as a comment in the programming solutions file.

2. Take & Drop: Function `(take n xs)` expects a natural number and a list. It returns the longest prefix of `xs` that contains at most `n` elements. Function `(drop n xs)` expects a natural number and a list. Roughly, it removes `n` elements from the front of the list. The exact semantics are given by this algebraic law: for any list `xs` and natural number `n`,

   ```
   (append (take n xs) (drop n xs)) == xs
   ```

   Implement take and drop.

3. Zip and unzip. Function `zip` converts a pair of lists to an association list; `unzip` converts an association list to a pair of lists. If `zip` is given lists of unequal length, its behavior is not specified.

```
-> (zip '(1 2 3) '(a b c))
((1 a) (2 b) (3 c))
-> (unzip '((I Magnin) (U Thant) (E Coli)))
((I U E) (Magnin Thant Coli))
```

Provided lists xs and ys are the same length, zip and unzip satisfy these laws:

```
(zip (car (unzip pairs)) (cadr (unzip pairs))) = pairs
         (unzip (zip xs ys))  = (list2 xs ys)
```

Implement Zip and Unzip.

4. Merging sorted lists. Implement function merge, which expects two sorted lists of numbers and returns a single sorted list containing exactly the same elements as the two argument lists together:

```
-> (merge '(1 2 3) '(4 5 6))
(1 2 3 4 5 6)
-> (merge '(1 3 5) '(2 4 6))
(1 2 3 4 5 6)
```

5. Interleaving lists. Implement function interleave, which expects as arguments two lists xs and ys, and returns a single list obtained by choosing elements alternately, first from xs and then from ys. When either xs or ys runs out, interleave takes the remaining elements from the other list, so that the elements of the result are exactly the elements of the two argument lists taken together.

```
-> (interleave '(1 2 3) '(a b c))
(1 a 2 b 3 c)
-> (interleave '(1 2 3) '(a b c d e f))
(1 a 2 b 3 c d e f)
-> (interleave '(1 2 3 4 5 6) '(a b c))
(1 a 2 b 3 c 4 5 6)
```

Theory Assignment: 50%

1. Calculational proof: Do Exercise 30 on page 164 of Ramsey, proving that reversing a list does not change its length. (Structural Induction works.)

2. Language Design: Do all parts of Exercise 37 on page 165 of Chapter 3. Be sure your answer to part (b) compiles and runs under uscheme.

3. Operational semantics & Algebraic laws: This problem has two parts. a) The operational semantics for μScheme includes rules for `cons`, `car`, and `cdr`. Assuming

that `x` and `xs` are variables and are defined in ρ, use the operational semantics to prove that

```
(cdr (cons x xs)) = xs
```

b) Use the operational semantics to prove or disprove the following conjecture: if `e1` and `e2` are arbitrary expressions, in any context where the evaluation of `e1` terminates and the evaluation of `e2` terminates, the evaluation of `(cdr (cons e1 e2))` terminates, and

```
(cdr (cons e1 e2)) = e2
```

The conjecture says that two independent evaluations, starting from the same initial state, produce the same value as a result.

General Evaluation Criteria

The programming exercise will be subject to the same evaluation criteria as previous assignments. Test your own implementation. Test cases are not provided. You can provide your test cases in your submission.

Submission

- Submit both the programming and theory assignments in a single (zip or tar) file containing a directory with the following naming structure {lastname1}-{lastname2}-programming-hw3.
- Programming Assignment: Name your programming submission file {lastname1}-{lastname2}-programming-hw3.
- Theory assignment
  1. Submit each exercise in the order they appear in this note. Start each exercise in a different page.
  2. Number your pages and add the total number of pages submitted in the first page.
  3. Submit a single pdf file containing all your answers. Your file should be named {lastname}-theory-hw3.pdf.
  4. Use Latex if possible.