CSE 461: Programming Languages Concepts

Prof. G. Tan
Spring 2018

Homework 6: Memoization in Scheme
Due on Apr 13th before class (12:20pm) in Canvas.

Submission: Please submit a DrRacket file via Canvas.

1. (5 points) In this task, we define a function called `fac_mem`, which is the memoized version of the factorial function; it takes an integer n as input and returns the factorial of n.

   - Define the factorial function `fac` as usual.
   - Define the `bind` and `lookup` functions for association lists, as we discussed in class. Recall that an association list in Scheme is just a list of pairs and each pair contains a key and a value.
     - `(bind k v al)` returns a new association list, which is the result of adding a new entry `(k,v)` to the beginning of association list `al`.
     - `(lookup k al)` returns the value for key `k` in `al` if there is an entry for `k` and returns `#f` otherwise.
   - Define a global variable `al` for the association list used in `fac_mem`.

     ```
     (define al '())
     ```

   - Finally, define `fac_mem`. When given `n`, it checks whether there is an entry for `n` in `al`. If there is, it returns the value in the entry; if not, it invokes `(fac n)`, adds the entry `(n, (fac n))` in the association list, and returns `(fac n)`.
     Notes:
     - To distinguish the two cases in `fac_mem`, add the following `display` command for the case when the input `n` is in the current association list. It displays the string on screen.

       ```
       (display ``memoization hit \n'')
       ```

- To add an entry to `al`, you will have to use `set!` to modify the global variable `al`. This has the side effect of modifying `al` so that it is visible to the next invocation of `fac_mem`.
- You will also need to use the sequencing construct in Scheme. In particular, `(begin e1 e2)` evaluates `e1` (which usually has some side effect) and then evaluates `e2`; the value of `e2` becomes the value of `(begin e1 e2)`. For example,

```
(begin
   (display ‘‘memoization hit \n’’)
   (+ 1 2))
```

  The example displays the message and returns 3.

2. (**Automatic memoization**, 3 points). In this task, we write a function `build_mem` that takes an input function and returns a function that is the memoized version of the input function. That is `(build_mem fac)` should return a function behaving the same as `fac_mem`. For simplicity, you can assume the input function takes exactly one parameter. As before, the returned function should display the message "memoization hit" when given an input that is already in the association list.

   Hint: the function returned by `build_mem` should have its own association list, which cannot be defined as a global variable. Introduce a local variable instead. That is, the function `build_mem` should be defined in the following way:

```
(define (build_mem f)
   (let ((al '()))
      (lambda (n) ... )))
```