SICP-Problem set 5

Jinfeng Lin

April 12, 2015

1 Homework

1.1 Exercise 4.6

See in the code file.

1.2 Exercise 4.7

It is enough to add the (eval (let*-¿nested-lets exp) env) to eval to let the let* run. Because we have rewrite the let* into let and first call will call on let*? clause, while the second one will come back to let?. So no extra work need to be done to let it run.

1.3 Exercise 4.13

I think it is better to just unbind the current frame. Because other wise this function would be dangerous. Any function have the right to interfere other function's performance. If some function overload basic operator like + and then decide to unbind it, then any function's + will be disabled. So I chose to unbind in the current frame.

1.4 Exercise 4.15

Assuming the try function works, then (try try)'s behaviors depend on the if clause of (halts? p p). If try could halt try then if(halts? p p) will go to true branch which will run-forever; Other wise if try could not halt try then (helts? try try) return false, the result is false. So there don't have such a function halt?

1.5 Exercise Cond vs If

If itself now become a procedure which means that the branches will be evaluated as the prediction does. In the normal scheme, the prediction is evaluated before the two branches. So the factorial function now could never hit the 1 clause any more, it will keep evaluating the recursive branch, so if you run (factorial 3) you will get a error report.

2 Building Evaluator