Problem Set 2 Solutions

Reminder: **QUIZ 1** in class on Monday, 7 October.

Problem 1

(1) $\models$ (f `<=` g) does not imply $\models$ ($e[x := f]$ `<=` $e[x := g]$). Consider the case when e is `-x`, f is `3` and g is `4`.

(2) $\models$ (e `<=` g) implies $\models$ ($e[x := f]$ `<=` $g[x := f]$).

Proof: Let V be any valuation, and let V_x be the valuation $V[x \leftarrow [\![f]\!]V]$. The hypothesis $\models$ (e `<=` g) means that $[\![e]\!]V' \leq [\![g]\!]V'$ for every valuation V'. So, in particular, we have that $[\![e]\!]V_x \leq [\![g]\!]V_x$. Combining this inequality with the Lemma given with the problem, we have $[\![e[x := f]]\!]V = [\![e]\!]V_x \leq [\![g]\!]V_x = [\![g[x := f]]\!]V$. Since V was arbitrary, we conclude that for *all* valuations V, $[\![e[x := f]]\!]V \leq [\![g[x := f]]\!]V$, that is, $\models$ ($e[x := f]$ `<=` $g[x := f]$).

■

Problem 2

The rules applied in each step are:

> 1.IG, 2.L, 3.IG, 4.I, 5.L, 6.I, 7.P, 8.C, 9.IG, 10.I, 11.P, 12.I, 13.P, 14.I, 15.L, 16.I, 17.P, 18.C, 19.IG, 20.I, 21.P, 22.IG, 23.P, 24.I, 25.L, 26.I, 27.P, 28.C, 29.IG, 30.I, 31.P, 32.I, 33.P, 34.I, 35.L, 36.I, 37.P, 38.CG, 39.IG

Problem 3

Inspection of the successive steps in the evaluation of `(iter-factorial 3)` shows that the subexpressions of a combination are evaluated from right-to-left. Steps 8-13 for example reveal this order. Go through those steps to see why!

Problem 4(a)

Suppose an expression M can be rewritten in one way by some rule A, and another way by the garbage-collect rule, G.

The redexes (lefthand sides) of the Scheme rewrite rules other than G are expressions. Only G has redexes which must be Scheme bodies beginning with at least one ⟨define⟩.

(Note: Pictures would help clarify the following argument a lot. We would have included them if there was time. Meanwhile, readers are encouraged to draw their own diagrams as the read what follows.)

Consider the case that rule A is not G. Then there are three possibilities in M:

1. The redex of A is disjoint from the G redex.
2. The redex of A is a subexpression of the G redex.
3. The redex of G is a subbody of the A redex.

In the disjoint case, there are two subcases. The first case is when the G-redex in M is not part of the righthand side of the A-rule when the A-rule is applied to M. This must be the case for all rules except I. In this case, M_1 can rewrite once by rule G to the same body to which M_2 can rewrite once by rule A. The second case is when A is I, and the G redex is part of definition of the immediate value which will be instantiated. Then M_1 may have two copies of the M's G-redex: the original one which must have been in the immediate value part of the definition in M of the variable being instantiated, and the one in the instantiated copy of this immediate value. In this case, it will take *two* G rewrites—one for each copy of the redex in M_1—for M_1 to reach the same M_3 that M_2 can reach by instantiating the garbage-collected immediate value part of the definition.

In the second case, there are two also subcases. First, G may have erased the A-redex from M_2, in which case G applied to M_1 can erase the result of rewriting the A redex, so M_3 will in fact equal M_2. The second subcase is when M's A-redex is left in M_2 by G. Then the result in M_1 of the A-rewriting of M will also be left in M_3 by G, and rewriting the A-redex left in M_2 will lead to the same M_3.

Finally, the G-redex may be within the A-redex. This means that A must be a C, L, or R rule, and these rules ensure that there will be at most one copy of the G-redex in M_1, and there will be a garbage-collected version of the A-redex in M_2. So with at most one G-rewriting, M_1 can reach the same M_3 that M_2 can reach by one A-rewriting.

We leave the case that A is G to the reader.

4(b)

The kind of argument which shows that the Strong Diamond property implies Confluence leads to a theorem which informally says that, if two sequences of rewritings of an expression M to bodies M_1 and M_2 differ only in when and where G-rewritings occur, then it is possible to rewrite M_1 and M_2 into the same M_3 *solely by G-rewritings*. However, it takes a lot of technical definitions of rewriting properties to state this theorem precisely. (Once the definitions are formulated, the proof of the theorem is not very hard.) Given this difficulty, we will give everyone credit for this problem (and if you didn't get full credit for it on your graded problem set, send email to 6044-lecturer@theory.lcs.mit.edu indicating how many points were deducted).

Another "uniqueness" theorem which is easy to state precisely is: if M rewrites to M_1, then there is a way to rewrite M to M_1 in which no G-rewriting precedes any non-G-rewriting. In other words, garbage collection can always be postponed till the end of any rewriting computation. This theorem is also not hard to prove using a diamond-like lemma similar to the one established in this problem, but the lemma of this problem is not quite enough to prove this garbage-collect-at-the-end theorem.