Problem Set 4

Due: 18 October, 1996.

Problem 1. The procedure for "flattening" arithmetic expressions suggested in Handout 3 can be described by rewrite rules, the main ones being "directed" distributivity:

$$\texttt{(}e\ \texttt{*}\ \texttt{(}f\ \texttt{+}\ g\texttt{))} \longrightarrow \texttt{((}e\ \texttt{*}\ f\texttt{)}\ \texttt{+}\ \texttt{(}e\ \texttt{*}\ g\texttt{))},$$
$$\texttt{((}f\ \texttt{+}\ g\texttt{)}\ \texttt{*}\ e\texttt{)} \longrightarrow \texttt{((}f\ \texttt{*}\ e\texttt{)}\ \texttt{+}\ \texttt{(}g\ \texttt{*}\ e\texttt{))}.$$

1(a) Suppose h is an arithmetic expression whose only operators are `+` and `*`—no `-` operator. Starting from h and successively applying the directed distributivity rules to "innermost" subexpressions, i.e., subexpressions which do not have sub-subexpressions to which the rule is also applicable, will eventually rewrite h to a "flattened" expression, namely, an expression with no subexpressions to which the rule applies. Give a simple justification for this claim.

1(b) The directed distributivity rules are actually terminating: starting with h, *no matter where the rule is successively applied*, a flattened expression will be reached. This fact is not obvious because if e is a "large" subexpression, then the righthand side of the rule with two occurrences of e may be larger, have more redexes, etc., than the lefthand side with only one e.

There is an ingenious, simple way to verify this termination claim. Define the measure, $\mu(h)$, of an arithmetic expression h, by induction:

- $\mu(h) = 2$ if h is `0`, `1`, or a variable.
- $\mu(\texttt{(}e\ \texttt{+}\ f\texttt{)}) = \mu(e) + \mu(f) + 1$.
- $\mu(\texttt{(}e\ \texttt{*}\ f\texttt{)}) = \mu(e) \times \mu(f)$.

Then termination follows immediately from the observation that if $h \underset{\text{dir-dist}}{\longrightarrow} h'$, then $\mu(h') < \mu(h)$. Prove this fact. *Hint*: You should of course verify that $\mu((e * f) + (e * g)) < \mu(e * (f + g))$, but termination does not follow *solely* from this fact, since the subexpression $e * (f + g)$ which gets rewritten may not the whole of h.

1(c) Now we extend the directed distributivity rule to handle arithmetic expressions with the unary negation operator, `-`, as well:

$$\begin{aligned} -(-e) &\longrightarrow e \\ -(f+g) &\longrightarrow (-f)+(-g) \\ -(f*g) &\longrightarrow (-f)*g \end{aligned}$$

Verify that the rewrite system consisting of the directed distributivity rules and the three rules above is terminating on all arithmetic expressions.

Problem 2. In Handout 2, we treated Scheme's `let` as a derived construct, i.e., "syntactic sugar." There are some technical benefits to treating `let` instead as a legitimate "functional kernel" construct, with specific rewrite rules defining its computational behavior. Describe such a set of rewrite rules for `let`.

Problem 3.

Definition. The *printable values* are defined inductively to be: booleans, numbers, the empty list `()`, and `cons`'s of printable values.

So lambda expressions are *not* printable values, and neither is a list any of whose members are not printable.

Definition. Two Scheme bodies are *observationally distinguishable* iff one rewrites to a printable value, and the other does not evaluate to exactly that printable value.

A *context* is a Scheme body with a single occurrence of a free variable, $[\,]$, called the *hole*. If C is a context and M is an expression, then $C[M]$ denotes the result of literally replacing—without any renaming—the hole in C by M.

Two bodies M, N are *observationally congruent*, in symbols, $M \cong N$ iff there is *no* Scheme context, C, such that $C[M]$ and $C[N]$ are observationally distinguishable.

3(a) Say that two two bodies are *return-distinguishable* iff *exactly one* of them can rewrite to a value (cf. Handout 2, § A.3).

Two bodies M, N are *return congruent*, in symbols, $M \underset{\text{ret}}{\cong} N$ iff there is *no* Scheme context, C, such that $C[M]$ and $C[N]$ are return-distinguishable.

Prove that $M \underset{\text{ret}}{\cong} N$ iff $M \cong N$.

3(b) Carefully prove that

$$\texttt{(+}\ M\ \ N\texttt{)}\cong\texttt{(+}\ N\ \ M\texttt{)}.$$

Hint: A handwaving argument that this follows immediately because addition is commutative is not acceptable. For example, you will have to appeal to facts specific to the functional kernel of Scheme we have considered thus far in the course, since this observational congruence no longer holds when we extend Scheme to include `set!` or `error` expressions.