Problem Set 6

Due: Friday, 13 November.

Problem 1. Let E_1 be a kernel Scheme expression satisfying the Variable Convention (VC) defined in `sm.scm`. Suppose E_1 rewrites by one Substitution Model rule application into E_2. For most of the rewrite rules, E_2 will still satisfy the VC, but for a few of the rules (you can find them listed in `sm.scm`), it will not. For each of these rules. give a simple example of E_1 satisfying the VC which rewrites to E_2 which does not satisfy the VC.

Problem 2. The definition of `crules:eq?` in the file `sm.scm` has a regrettable property: it allows the substitution model to return `#t` as the value of

```
(eq? (lambda (x) x) (lambda (x_0) x_0))
```

but `#f` as the value of

```
(eq? (lambda (x) x) (lambda (y) y))
```

This violates an important "black box" principal of Scheme, that printable values of Scheme expressions are invariant under renaming of bound variables to "fresh" variables.

Define a procedure

$$\texttt{canonical-bound-vars} : \langle\text{Scheme expression}\rangle \longrightarrow \langle\text{Scheme expression}\rangle$$

so that this problem with `eq?` is repaired simply by editing `crules:eq?` to call `canonical-bound-vars` instead of the `clean-suffixes` procedure it currrently calls.

Problem 3. Scheme expressions E and F are said to be *renaming equivalent*[1], written as,

$$E =_{\text{fresh}} F,$$

if they can be tranformed into the same expression by some fresh renaming of their bound variables. (That is, the same expression is returned by `(canonical-bound-vars 'E)` and `(canonical-bound-vars 'F)`.)

Suppose E_1 is a kernel Scheme expression satisfying the VC, and E_1 rewrites by one Substitution Model rule application (with the revised `crules:eq?`) into E_2. Then, if $E_1 =_{\text{fresh}} F_1$ and F_1 also satisfies the VC, it turns out that that F_1 must rewrite by one substitution model rule application into some F_2 such that $E_2 =_{\text{fresh}} F_2$.

3(a) Carefully explain why this property – that renaming equivalence of expressions satisfying the VC is preserved by a Substitution Model rule application – holds in the particular case that E_2 is obtained from E_1 by `rule:instantiate`.

3(b) Give a counterexample of E_1, F_1 showing that even if E_1 satisfies the VC, the hypothesis used above that F_1 also satisfies the VC cannot be relaxed.

[1] In the programming theory literature, "renaming equivalence" is often called "α-equivalence," written $=_\alpha$.

3(c) Conclude that if $E_1 =_{\text{fresh}} F_1$, and evaluating E_1 returns a printable value, then evaluating F_1 (starting in the same global environment) returns the same printable value.

Problem 4. A set $\mathcal{R} \subseteq \mathbb{Z}^n$ of n-tuples of integers is said to be a *Diophantine root set* iff there a Diophantine polynomial, p, of n variables such that $\mathcal{R}$ is the set of roots of p. A set $\mathcal{D} \subseteq \mathbb{Z}^m$ is said to be a Diophantine set if it is the projection on the first m coordinates of some Diophantine root set. (That is, $\mathcal{D} = \{(k_1, \ldots, k_m) \mid (k_1, \ldots, k_m, \ldots) \in \mathcal{R}\}$ for some Diophantine root set $\mathcal{R}$.)

4(a) Show that the range of any Diophantine polynomial is a Diophantine set.

4(b) Let $p(x_1, \ldots, x_n)$ be a Diophantine polynomial and $\mathcal{D}$ the Diophantine set which is the projection on the first coordinate of the root set of p. Define

$$f(x_1, \ldots, x_n) ::= (x_1 + 1)(1 - p(x_1, \ldots, x_n)^2) - 1.$$

Prove that $\mathcal{D} \cap \mathbf{N} = \text{range } f \cap \mathbf{N}$.