Advanced Type Systems
Homework #4

Instructor: Derek Dreyer
Assigned: February 15, 2006
Due: February 27, 2006

1 Termination in the Presence of Existentials

Prove that call-by-value System F is still terminating even when extended with existential types. This result should not be surprising, since it is well-known that existential types are encodable in System F in terms of universal types (see Section 11.3.5 in Girard’s *Proofs and Types*).

Here are the appropriate extensions:

<table>
<thead>
<tr>
<th>Types</th>
<th>τ ::= ...</th>
<th>τ′</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terms</td>
<td>e, f ::= ...</td>
<td>pack [σ, e] as τ</td>
</tr>
<tr>
<td>Values</td>
<td>v, w ::= ...</td>
<td>pack [σ, v] as τ</td>
</tr>
</tbody>
</table>

\[\Delta \vdash σ \quad \Delta; Γ \vdash e : τ[σ/α] \]

\[\Delta; Γ \vdash \text{pack} [σ, e] \text{ as } τ : α \]

\[\Delta; Γ \vdash e : τ \quad \Delta, α; Γ, x : τ \vdash e′ : τ′ \quad α \not\in \text{FV}(τ′) \]

\[\Delta; Γ \vdash \text{let } [α, x] = \text{unpack } e \text{ in } e′ : τ′ \]

\[\Delta; Γ \vdash e \sim e′ \]

\[\text{pack} [σ, e] \text{ as } τ \sim \text{pack} [σ, e′] \text{ as } τ \]

\[\text{let } [α, x] = \text{unpack } e \text{ in } f \sim \text{let } [α, x] = \text{unpack } e′ \text{ in } f \]

\[\text{let } [α, x] = \text{unpack} (\text{pack} [σ, v] \text{ as } τ) \text{ in } e \sim e[σ/α][v/x] \]

2 A Simple Parametricity Result

We are working in plain call-by-value System F. Suppose that we are given a closed value \(f\) of type \(\forall α. (α \to α) \to (α \to α)\), a closed type \(τ\), and two closed values \(v_s\) and \(v_z\) of type \(τ \to τ\) and \(τ\), respectively.

Your task is to define a set \(S\) that precisely characterizes the possible values that \(f[τ](v_s)(v_z)\) might compute to, but whose definition does NOT mention \(f\). That is, for any value \(v\), if it is possible that \(f[τ](v_s)(v_z) \sim^* v\), then \(v \in S\), and vice versa. (If I left out “and vice versa”, then you could just pick \(S\) to be the universal set.) Prove that your definition of \(S\) is correct.
3 Termination in the Presence of Constants of Arbitrary Type

Prove that call-by-value System F is still terminating even when extended with constants at every type (à la Girard). For the purpose of this problem, you can assume that the language has no base type $T$ and no other constants beside 0 (see below). This illustrates that the existence of a value of the “false” type $\forall \alpha. \alpha$ does not break termination per se.

Here are the appropriate extensions:

$$
\begin{align*}
\text{Terms} & \quad e,f ::= \cdots \mid 0 \\
\text{Values} & \quad v,w ::= \cdots \mid 0 \mid 0[\tau] \\
\Delta; \Gamma \vdash 0 : \forall \alpha. \alpha \\
0[\sigma \rightarrow \tau](v) & \leadsto 0[\tau] \\
0[\forall \alpha. \tau][\sigma] & \leadsto 0[\tau[\sigma/\alpha]]
\end{align*}
$$

4 Using Girard’s $J$ to Implement Recursion

For this problem, we are working in System F with full reduction, extended with Girard’s 0 and $J$ operators. Recall the semantics of Girard’s $J$ operator:

$$
\Delta; \Gamma \vdash J : \forall \alpha. \forall \beta. \alpha \rightarrow \beta
$$

$$
\begin{align*}
\sigma = \tau & \quad & \sigma \neq \tau & \quad \sigma \text{ and } \tau \text{ are closed} \\
J[\sigma][\tau](e) & \leadsto e & J[\sigma][\tau](e) & \leadsto 0[\tau]
\end{align*}
$$

Let us say that a closed term $Y$ “encodes the fixed-point combinator at type $\tau$” if: (1) $Y$ has type $(\tau \rightarrow \tau) \rightarrow \tau$, and (2) for any closed term $f$ of type $\tau \rightarrow \tau$, there exists a term $e$ such that $Y(f) \leadsto e$ and $e \leadsto f(e)$.

Your task is to use Girard’s $J$ operator to define a closed term $\text{fix}$ such that for all closed types $\tau$, it is the case that $\text{fix}[\tau]$ encodes the fixed-point combinator at type $\tau$.

**Hint:** $Y = \lambda f. (\lambda x. f(x x))(\lambda x. f(x x))$ is the fixed-point combinator in the classical untyped $\lambda$-calculus. At least in my solution to this problem, the untyped erasure of my $\text{fix}$ is precisely the classical $Y$ combinator.

5 Extra Credit: The $DJ$ Is a Real Smooth Operator

Let us say that an operator $Op$ of type $\sigma$ “satisfies the Harper-Mitchell criterion” if: (1) there exist closed terms of type $\sigma$ in pure System F, and (2) adding $Op$ to System F causes strong normalization to fail. Furthermore, let us say that $Op$ is a “real smooth operator” if: (1) it satisfies the Harper-Mitchell criterion, and (2) for all operators $\text{Lame}$ that satisfy the Harper-Mitchell criterion, for all closed types $\tau$, if $\text{Lame}$ can be used to construct a term that encodes the fixed-point operator at type $\tau$, then so can $Op$. In other words, $Op$ is real smooth if it is the most powerful operator (in terms of encoding recursion) that one can define without violating the Harper-Mitchell criterion.

Prove that there exists a real smooth operator by defining one and proving that it is real smooth. (Mine is called $DJ$, short for “Dreyer’s $J$”.)