Advanced Type Systems
Homework #2

Instructor: Derek Dreyer

Assigned: January 23, 2006
Due: January 30, 2006

1 Termination in the Presence of Sums

Consider extending the simply-typed λ-calculus with sum types and the corresponding standard introduction and elimination forms:

$$\begin{array}{lrl} \text{Types} & \tau & ::= \cdots \mid \tau_1 + \tau_2 \\ \text{Terms} & e & ::= \cdots \mid \mathtt{inj}_i(e) \mid \mathtt{case}\ e\ \mathtt{of}\ \mathtt{inj}_1(x) \Rightarrow e_1 \mid \mathtt{inj}_2(x) \Rightarrow e_2 \\ \text{Values} & v & ::= \cdots \mid \mathtt{inj}_i(v) \end{array}$$

The typing and small-step evaluation rules for sums (also standard) are as follows:

$$\frac{\Gamma \vdash e : \tau_i \quad i \in \{1,2\}}{\Gamma \vdash \mathtt{inj}_i(e) : \tau_1 + \tau_2} \qquad \frac{\Gamma \vdash e : \tau_1 + \tau_2 \quad \Gamma, x:\tau_1 \vdash e_1 : \tau \quad \Gamma, x:\tau_2 \vdash e_2 : \tau}{\Gamma \vdash \mathtt{case}\ e\ \mathtt{of}\ \mathtt{inj}_1(x) \Rightarrow e_1 \mid \mathtt{inj}_2(x) \Rightarrow e_2 : \tau}$$

$$\frac{e \leadsto e'}{\mathtt{inj}_i(e) \leadsto \mathtt{inj}_i(e')} \qquad \frac{e \leadsto e'}{\begin{array}{l}\mathtt{case}\ e\ \mathtt{of}\ \mathtt{inj}_1(x) \Rightarrow e_1 \mid \mathtt{inj}_2(x) \Rightarrow e_2 \leadsto \\ \quad \mathtt{case}\ e'\ \mathtt{of}\ \mathtt{inj}_1(x) \Rightarrow e_1 \mid \mathtt{inj}_2(x) \Rightarrow e_2\end{array}}$$

$$\frac{}{\mathtt{case}\ \mathtt{inj}_i(v)\ \mathtt{of}\ \mathtt{inj}_1(x) \Rightarrow e_1 \mid \mathtt{inj}_2(x) \Rightarrow e_2 \leadsto e_i[v/x]}$$

Problem: Extend the logical relations proof that well-typed terms terminate to include the new constructs for sums. Just give the new cases of the proof. You should not need to modify any of the old cases (*i.e.,* the ones I showed in class). For uniformity of solutions, base your proof on the following formulation of the logical relation (you will need to add a definition for $\mathcal{V}[\![\tau_1 + \tau_2]\!]$):

$$\begin{array}{rcl} \mathcal{C}[\![\tau]\!] & \stackrel{\text{def}}{=} & \{e \mid \exists v.\ e \leadsto^* v \wedge v \in \mathcal{V}[\![\tau]\!]\} \\ \\ \mathcal{V}[\![\mathbf{T}]\!] & \stackrel{\text{def}}{=} & \{c\} \\ \mathcal{V}[\![\tau_1 \to \tau_2]\!] & \stackrel{\text{def}}{=} & \{v \mid \forall v' \in \mathcal{V}[\![\tau_1]\!].\ v(v') \in \mathcal{C}[\![\tau_2]\!]\} \end{array}$$

2 Strong Normalization

Recall the logical relation employed in Tait's method for proving strong normalization:

$$\begin{array}{rcl} \mathcal{C}[\![\mathbf{T}]\!] & \stackrel{\text{def}}{=} & \text{SN} \\ \mathcal{C}[\![\tau_1 \to \tau_2]\!] & \stackrel{\text{def}}{=} & \{e \mid \forall e' \in \mathcal{C}[\![\tau_1]\!].\ e(e') \in \mathcal{C}[\![\tau_2]\!]\} \end{array}$$

Part A: Suppose we change the definition to include an explicit requirement that $\mathcal{C}[\![\tau_1 \to \tau_2]\!] \subseteq \text{SN}$:

$$\begin{array}{rcl} \mathcal{C}[\![\mathbf{T}]\!] & \stackrel{\text{def}}{=} & \text{SN} \\ \mathcal{C}[\![\tau_1 \to \tau_2]\!] & \stackrel{\text{def}}{=} & \{e \mid \boxed{e \in \text{SN} \wedge}\ \forall e' \in \mathcal{C}[\![\tau_1]\!].\ e(e') \in \mathcal{C}[\![\tau_2]\!]\} \end{array}$$

This new version of the logical relation avoids the need for the "Main Lemma", but it will require more work to be done at some other point(s) in the proof. Please note all such points, and show any new work that has to be done.

Part B: Suppose we extend the "full reduction" relation with a rule for η-reduction:

$$\frac{x \notin \text{FV}(e)}{\lambda x.(e\,x) \rightsquigarrow e}$$

Prove or disprove: Strong normalization holds under full $\beta\eta$-reduction. For a proof, show any new work that has to be done above and beyond the proof of strong normalization for full β-reduction. For a disproof, give a counterexample.

3 The Return of Uninhabitation

Consider the simply-typed λ-calculus with *no* constants of base type $\mathbf{T}$. In class, we discussed that it is easy to prove *uninhabitation* (*i.e.,* that there are no closed terms of base type) for this language by relying on termination and preservation. We've also seen that it is possible to prove uninhabitation by devising a syntactic characterization of which types are (un-)inhabited.

Problem: Prove uninhabitation using a direct *logical relations* argument. Your proof should NOT mention any reduction or evaluation relation or any "dynamic" property such as termination.