operational logical relations and contextual equivalence for $\lambda 2$

dan frumin

last updated: May 11, 2017

Recall that we have a relational model of $\lambda 2$ with the following properties.

Theorem 1 (Fundamental property of logical relations)**.** $\Delta; \Gamma \vdash e : \tau$ *implies* $\Delta; \Gamma \models e \approx_{log} e : \tau$

which we prove using the following compatibility lemmas:

Lemma 2 (Compatibility lemmas)**.**

$$\text{\textit{log\_var}}\ \frac{\Gamma(x) = \tau}{\Delta; \Gamma \models x \approx_{log} x : \tau}$$

$$\text{\textit{log\_true}}\ \frac{}{\Delta; \Gamma \models \mathsf{true} \approx_{log} \mathsf{true} : \mathsf{bool}}$$

$$\text{\textit{log\_false}}\ \frac{}{\Delta; \Gamma \models \mathsf{false} \approx_{log} \mathsf{false} : \mathsf{bool}}$$

$$\text{\textit{log\_if}}\ \frac{\Delta; \Gamma \models e_0 \approx_{log} e_0' : \mathsf{bool} \qquad \Delta; \Gamma \models e_1 \approx_{log} e_1' : \tau \qquad \Delta; \Gamma \models e_2 \approx_{log} e_2' : \tau}{\Delta; \Gamma \models \mathsf{if}\ e_0\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2 \approx_{log} \mathsf{if}\ e_0'\ \mathsf{then}\ e_1'\ \mathsf{else}\ e_2' : \tau}$$

$$\text{\textit{log\_app}}\ \frac{\Delta; \Gamma \models e_0 \approx_{log} e_0' : \tau \to \tau' \qquad \Delta; \Gamma \models e_1 \approx_{log} e_1' : \tau}{\Delta; \Gamma \models e_0\ e_1 \approx_{log} e_0'\ e_1' : \tau'}$$

$$\text{\textit{log\_lam}}\ \frac{\Delta; (x : \sigma), \Gamma \models e \approx_{log} e' : \tau}{\Delta; \Gamma \models \lambda x.e \approx_{log} \lambda x.e' : \sigma \to \tau}$$

$$\text{\textit{log\_tapp}}\ \frac{\Delta; \Gamma \models e \approx_{log} e' : \forall \alpha.\tau(\alpha) \qquad \Delta \vdash \sigma}{\Delta; \Gamma \models e[\sigma] \approx_{log} e'[\sigma] : \tau(\sigma)}$$

$$\text{\textit{log\_tlam}}\ \frac{\alpha, \Delta; \Gamma \models e \approx_{log} e' : \tau(\alpha)}{\Delta; \Gamma \models \Lambda\alpha.e \approx_{log} \Lambda\alpha.e' : \forall \alpha.\tau(\alpha)}$$

We wish to show that when two terms are logically related, they are equivalent as programs. We formalise this with the help of *contextual equivalence*.

Definition 3 (Program contexts)**.** *Program contex are given as the following inductive definition/grammar where* e, x, τ *are types/grammars for expressions, variables, and types, respectively.*

$$\mathcal{C} := \Box \mid \mathsf{if}\ \mathcal{C}\ \mathsf{then}\ e\ \mathsf{else}\ e \mid \mathsf{if}\ e\ \mathsf{then}\ \mathcal{C}\ \mathsf{else}\ e \mid \mathsf{if}\ e\ \mathsf{then}\ e\ \mathsf{else}\ \mathcal{C} \mid \lambda x : \tau.\mathcal{C} \mid \mathcal{C}\ e \mid e\ \mathcal{C} \mid \Lambda\alpha.\mathcal{C} \mid \mathcal{C}[\tau]$$

If $\mathcal{C}$ is a program context and e is an expression, we write $\mathcal{C}[e]$ for the (variable-binding) substitution of e for $\Box$ in $\mathcal{C}$.

We will only consider *typed contexts*. We write $\mathcal{C} : (\Delta; \Gamma \vdash \tau \Rightarrow \Delta'; \Gamma' \vdash \tau')$ if whenever $\Delta; \Gamma \vdash e : \tau$, it is the case that $\Delta'; \Gamma' \vdash \mathcal{C}[e] : \tau'$.

Using typed program contexts we can define the notion of contextual equivalence, which formalizes the notions of program equivalence.

Definition 4 (Context equivalence)**.** *We say that two (possibly open) expressions are contextually equivalent (denoted as* $\Delta; \Gamma \vdash e \approx_{ctx} e' : \tau$*), if they have the same "observable behavior" under any program context; that is*

$$\begin{aligned}
&\Delta; \Gamma \vdash e \approx_{ctx} e' : \tau \\
&\iff \\
&\forall(\mathcal{C} : \Delta; \Gamma \vdash \tau \Rightarrow \cdot; \cdot \vdash \mathsf{bool}).(\forall v.\mathcal{C}[e] \Downarrow v \iff \mathcal{C}[e'] \Downarrow v)
\end{aligned}$$

Note that we only quantify over the typed context with the return type bool. In general, we would like to quantify over all the typed contexts $\mathcal{C} : (\Delta; \Gamma \vdash \tau \Rightarrow \cdot; \cdot \vdash \tau')$ where τ' is a *base type* (e.g. integer, boolean, unit type ...), but the only base type in our system is bool. If we allow $\mathcal{C}$ to be quantified over arbitrary program contexts, then the notion of contexutal equivalence will be too fine. Consider, for instance, a context $\mathcal{C} = \lambda x : \mathsf{bool}.\Box$. This context has a type $\mathcal{C} : (\cdot; (x : \mathsf{bool}) \vdash \tau \Rightarrow \cdot; \cdot \vdash \mathsf{bool} \to \tau)$. If we were to allow contexts of such type in definition 4, then the notion of contextual equivalence will collapse to syntactic equality: $\forall v.(\lambda x : \mathsf{bool}.\Box)[e] \Downarrow v \iff (\lambda x : \mathsf{bool}.\Box)[e'] \Downarrow v$ holds iff $(\lambda x : \mathsf{bool}.e) = (\lambda x : \mathsf{bool}.e')$ iff $e = e'$.

We want to prove the following theorem:

Theorem 5. $\Delta; \Gamma \models e \approx_{log} e' : \tau$ *implies* $\Delta; \Gamma \vdash e \approx_{ctx} e' : \tau$

Proving this theorem by induction on the structure of the context won't work. We will need an auxiliary lemma.

Lemma 6. *Let* $\mathcal{C} : (\Delta; \Gamma \vdash \tau \Rightarrow \Delta'; \Gamma' \vdash \tau')$ *and* $\Delta; \Gamma \models e \approx_{log} e' : \tau$*. Then* $\Delta'; \Gamma' \models \mathcal{C}[e] \approx_{log} \mathcal{C}[e'] : \tau'$*.*

To see that Lemma 6 implies theorem 5, let $\Delta; \Gamma \models e \approx_{log} e' : \tau$ and let $\mathcal{C} : (\Delta; \Gamma \vdash \tau \Rightarrow \cdot; \cdot \vdash \mathsf{bool})$. From lemma 6 you get $\cdot; \cdot \models \mathcal{C}[e] \approx_{log} \mathcal{C}[e'] : \mathsf{bool}$. By picking empty substitutions, one can deduce that both $\mathcal{C}[e]$ and $\mathcal{C}[e']$ terminate to the same value.

Proof of Lemma 6. We prove this by structural induction on $\mathcal{C} : (\Delta; \Gamma \vdash \tau \Rightarrow \Delta'; \Gamma' \vdash \tau')$.

Case (1): $\mathcal{C} = \Box$. This is trivial.

Case (2): $\mathcal{C} = \mathsf{if}\ \mathcal{C}'\ \mathsf{then}\ p\ \mathsf{else}\ p'$. Since $\mathcal{C} : (\Delta; \Gamma \vdash \tau \Rightarrow \Delta'; \Gamma' \vdash \tau')$, it must be the case that $\mathcal{C}' : (\Delta; \Gamma \vdash \tau \Rightarrow \Delta'; \Gamma' \vdash \mathsf{bool})$ and $\Delta'; \Gamma' \vdash p, p' : \tau'$. By the induction hypothesis we have $\Delta'; \Gamma' \models \mathcal{C}'[e] \approx_{log} \mathcal{C}'[e'] : \mathsf{bool}$. Then we use the fundamental property of logical relations for p and p', and the compatibility lemma log_if:

$$\text{log\_if}\frac{\Delta'; \Gamma' \models \mathcal{C}'[e] \approx_{log} \mathcal{C}'[e'] : \mathsf{bool} \qquad \text{fund}\frac{}{\Delta'; \Gamma' \models p \approx_{log} p : \tau'} \qquad \text{fund}\frac{}{\Delta'; \Gamma' \models p' \approx_{log} p' : \tau'}}{\Delta'; \Gamma' \models \mathsf{if}\ \mathcal{C}'[e]\ \mathsf{then}\ p\ \mathsf{else}\ p' \approx_{log} \mathsf{if}\ \mathcal{C}'[e']\ \mathsf{then}\ p\ \mathsf{else}\ p' : \tau}$$

Case (3): $\mathcal{C} = \mathsf{if}\ cthen\mathcal{C}'elsep$. It then must be the case that $\Delta'; \Gamma' \vdash c : \mathsf{bool}$, and $\Delta'; \Gamma' \vdash p : \tau'$, and $\mathcal{C}' : (\Delta; \Gamma \vdash \tau \Rightarrow \Delta', \Gamma' \vdash \tau')$. By the induction hypothesis we have $\Delta'; \Gamma' \models \mathcal{C}[e] \approx_{log} \mathcal{C}[e'] : \tau'$. We get the desired result by using the log_if compatibility lemma and the fundamental property.

Case (4): $\mathcal{C} = \mathsf{if}\ c\ \mathsf{then}\ p\ \mathsf{else}\ \mathcal{C}'$. Similar to case (3).

Case (5): $\mathcal{C} = \lambda x : \sigma.\mathcal{C}'$. Because $\mathcal{C} : (\Delta; \Gamma \vdash \tau \Rightarrow \Delta; \Gamma \vdash \tau')$, it must be the case that $\tau' = \sigma \to \sigma'$ and $\mathcal{C}' : (\Delta; \Gamma \vdash \tau \Rightarrow \Delta'; (x : \sigma), \Gamma' \vdash \sigma')$. Then, by the induction hypothesis, $\Delta'; (x : \sigma), \Gamma' \models \mathcal{C}'[e] \approx_{log} \mathcal{C}'[e'] : \sigma'$. We get the desired result by the compatibility lemma.

$$\text{log\_lam}\frac{\Delta'; (x : \sigma), \Gamma' \models \mathcal{C}'[e] \approx_{log} \mathcal{C}'[e'] : \sigma'}{\Delta'; \Gamma' \models \lambda x.\mathcal{C}'[e] \approx_{log} \lambda x.\mathcal{C}[e'] : \sigma \to \sigma' = \tau'}$$

Case (6): $\mathcal{C} = \mathcal{C}'\ t$. In that case $\mathcal{C}' : (\Delta; \Gamma \vdash \tau \Rightarrow \Delta'; \Gamma' \vdash \sigma \to \tau')$ and $\Delta'; \Gamma' \vdash t : \sigma$ for some type σ. By the induction hypothesis we have $\Delta'; \Gamma' \models \mathcal{C}'[e] \approx_{log} \mathcal{C}'[e'] : \sigma \to \tau'$. Then we use the comptibility lemma

$$\text{log\_app}\frac{\Delta'; \Gamma' \models \mathcal{C}'[e] \approx_{log} \mathcal{C}'[e'] : \sigma \to \tau' \qquad \Delta'; \Gamma' \models t \approx_{log} t : \sigma}{\Delta'; \Gamma' \models \mathcal{C}'[e]\ t \approx_{log} \mathcal{C}'[e']\ t : \tau'}$$

Case (7): $\mathcal{C} = t\ \mathcal{C}'$. Similar to Case (6).

Case (8): $\mathcal{C} = \Lambda\alpha.\mathcal{C}'$. Then $\tau' = \forall\alpha.\sigma$ for some type σ and $\mathcal{C}' : (\Delta; \Gamma \vdash \alpha, \Delta'; \Gamma' \vdash \sigma)$. By the induction hypothesis it is the case $\alpha, \Delta'; \Gamma' \models \mathcal{C}'[e] \approx_{log} \mathcal{C}'[e'] : \sigma$. We obtain the necessary result by the compatibility lemma

$$\text{log\_tlam}\frac{\alpha, \Delta'; \Gamma' \models \mathcal{C}'[e] \approx_{log} \mathcal{C}[e'] : \sigma}{\Delta'; \Gamma' \models \Lambda\alpha.\mathcal{C}'[e] \approx_{log} \Lambda\alpha.\mathcal{C}'[e'] : \forall\alpha.\sigma = \tau'}$$

Case (9): $\mathcal{C} = \mathcal{C}'[\sigma]$. Then $\tau' = \phi(\sigma)$ and $\mathcal{C}' : (\Delta; \Gamma \vdash \tau \Rightarrow \Delta'; \Gamma' \vdash \forall\alpha.\phi(\alpha))$. By the induction hypothesis $\Delta', \Gamma' \models \mathcal{C}'[e] \approx_{log} \mathcal{C}'[e'] : \forall\alpha.\phi(\alpha))$. We obtain the result by applying the log_tapp compatibility lemma. $\Box$