CMSC 336: Type Systems for Programming Languages (Winter 2008)

Homework 5

Due: Thursday, March 13, 2008

1 Parametricity

Recall from class the following *logical relation* for System F (where we assume a call-by-value operational semantics for System F, as we did in class and in Homework 4).

Below, the metavariable R ranges over relations on values (i.e., R denotes a set of pairs of values (v, v')). The metavariable η ranges over finite maps from type variables α to triples (σ, σ', R), where σ and σ' are closed types and R is a value relation. Finally, the metavariable γ ranges over finite maps from term variables x to closed values v (i.e., values with no free type or term variables).

$$
\begin{array}{lll}
R : \tau \leftrightarrow \tau' & \text{iff} & \forall (v, v') \in R.\ \vdash v : \tau \ \wedge\ \vdash v' : \tau' \\[1em]
v \approx v' : \alpha \mid \eta & \text{iff} & \eta(\alpha) = (\tau, \tau', R) \ \wedge\ (v, v') \in R \\
\lambda x{:}\eta^1(\tau_1).\, e \approx \lambda x{:}\eta^2(\tau_1).\, e' : \tau_1 \to \tau_2 \mid \eta & \text{iff} & \forall v, v'.\ v \approx v' : \tau_1 \mid \eta \implies \\
 & & \qquad e[v/x] \sim e'[v'/x] : \tau_2 \mid \eta \\
\Lambda \alpha.\, e \approx \Lambda \alpha.\, e' : \forall \alpha.\, \tau \mid \eta & \text{iff} & \forall \sigma, \sigma', R.\ R : \sigma \leftrightarrow \sigma' \implies \\
 & & \qquad e[\sigma/\alpha] \sim e'[\sigma'/\alpha] : \tau \mid \eta[\alpha \mapsto (\sigma, \sigma', R)] \\[1em]
e \sim e' : \tau \mid \eta & \text{iff} & \exists v, v'.\ e \to^* v \ \wedge\ e' \to^* v' \ \wedge\ v \approx v' : \tau \mid \eta
\end{array}
$$

If $\eta = \{\alpha_1 \mapsto (\sigma_1, \sigma'_1, R_1), \ldots, \alpha_n \mapsto (\sigma_n, \sigma'_n, R_n)\}$ then
$\eta^1(e) = e[\sigma_1/\alpha_1, \ldots, \sigma_n/\alpha_n]$
and $\eta^2(e) = e[\sigma'_1/\alpha_1, \ldots, \sigma'_n/\alpha_n]$.

$$
\begin{array}{lll}
\eta \models \Delta & \text{iff} & \forall \alpha \in \Delta.\ \alpha \in \mathrm{dom}(\eta) \ \wedge \\
 & & \qquad (\eta(\alpha) = (\sigma, \sigma', R) \implies R : \sigma \leftrightarrow \sigma') \\[1em]
\gamma \approx \gamma' : \Gamma \mid \eta & \text{iff} & \mathrm{dom}(\gamma) = \mathrm{dom}(\gamma') = \mathrm{dom}(\Gamma) \ \wedge \\
 & & \forall x \in \mathrm{dom}(\Gamma).\ \gamma(x) \approx \gamma'(x) : \Gamma(x) \mid \eta \\[1em]
\Delta; \Gamma \vdash e \sim e' : \tau & \text{iff} & \forall \eta, \gamma, \gamma'.\ \eta \models \Delta \ \wedge\ \gamma \approx \gamma' : \Gamma \mid \eta \implies \\
 & & \qquad \gamma(\eta^1(e)) \sim \gamma'(\eta^2(e')) : \tau \mid \eta
\end{array}
$$

Theorem (Parametricity)**.** If $\Delta; \Gamma \vdash e : \tau$ then $\Delta; \Gamma \vdash e \sim e : \tau$.

The proof is by induction on the typing derivation $\Delta; \Gamma \vdash e : \tau$.

- **Case:** $\dfrac{\Delta; \Gamma, x : \tau_1 \vdash e : \tau_2}{\Delta; \Gamma \vdash \lambda x{:}\tau_1.\, e : \tau_1 \to \tau_2}$ (Abs)

  Proof

  We are required to show that $\Delta; \Gamma \vdash \lambda x{:}\tau_1.\, e \sim \lambda x{:}\tau_1.\, e : \tau_1 \to \tau_2$.
  Consider arbitrary η, γ, and γ' such that

– $\eta \models \Delta$, and
– $\gamma \approx \gamma' : \Gamma \mid \eta$.

We are required to show that $\gamma(\eta^1(\lambda x{:}\tau_1.\, e)) \sim \gamma'(\eta^2(\lambda x{:}\tau_1.\, e)) : \tau_1 \to \tau_2 \mid \eta$
$\equiv \lambda x{:}\eta^1(\tau_1).\, \gamma(\eta^1(e)) \sim \lambda x{:}\eta^2(\tau_1).\, \gamma'(\eta^2(e)) : \tau_1 \to \tau_2 \mid \eta$.
Take $v = \lambda x{:}\eta^1(\tau_1).\, \gamma(\eta^1(e))$.
Take $v' = \lambda x{:}\eta^2(\tau_1).\, \gamma'(\eta^2(e))$.
We are required to show that

– $\lambda x{:}\eta^1(\tau_1).\, \gamma(\eta^1(e)) \to^* \lambda x{:}\eta^1(\tau_1).\, \gamma(\eta^1(e))$,
which is immediate since $\lambda x{:}\eta^1(\tau_1).\, \gamma(\eta^1(e))$ is a value,
– $\lambda x{:}\eta^2(\tau_1).\, \gamma'(\eta^2(e)) \to^* \lambda x{:}\eta^2(\tau_1).\, \gamma'(\eta^2(e))$,
which is immediate since $\lambda x{:}\eta^2(\tau_1).\, \gamma'(\eta^2(e))$ is a value, and
– $\lambda x{:}\eta^1(\tau_1).\, \gamma(\eta^1(e)) \approx \lambda x{:}\eta^2(\tau_1).\, \gamma'(\eta^2(e)) : \tau_1 \to \tau_2 \mid \eta$,
which we conclude as follows:

Consider arbitrary v_1 and v_1' such that
* $v_1 \approx v_1' : \tau_1 \mid \eta$.

We are required to show $\gamma(\eta^1(e))[v_1/x] \sim \gamma'(\eta^2(e))[v_1'/x] : \tau_2 \mid \eta$.
Applying the induction hypothesis to $\Delta; \Gamma, x : \tau_1 \vdash e : \tau_2$, we have that $\Delta; \Gamma, x : \tau_1 \vdash e \sim e : \tau_2$.
Instantiate the latter with η, $\gamma[x \mapsto v_1]$, and $\gamma'[x \mapsto v_1']$. Note that
* $\eta \models \Delta$,
which follows from above, and
* $\gamma[x \mapsto v_1] \approx \gamma'[x \mapsto v_1'] : \Gamma, x : \tau_1 \mid \eta$,
which follows from
· $\gamma \approx \gamma' : \Gamma \mid \eta$ (follows from above), and
· $v_1 \approx v_1' : \tau_1 \mid \eta$ (follows from above).

Hence, $\gamma[x \mapsto v_1](\eta^1(e)) \sim \gamma'[x \mapsto v_1'](\eta^2(e)) : \tau_2 \mid \eta$.
Thus, $\gamma(\eta^1(e))[v_1/x] \sim \gamma'(\eta^2(e))[v_1'/x] : \tau_2 \mid \eta$, as we were required to show.

□

- **Case:** $\dfrac{\Delta; \Gamma \vdash e_1 : \tau_1 \to \tau_2 \qquad \Delta; \Gamma \vdash e_2 : \tau_1}{\Delta; \Gamma \vdash e_1\ e_2 : \tau_2}$ (App)

Proof

We are required to show that $\Delta; \Gamma \vdash e_1\ e_2 \sim e_1\ e_2 : \tau_2$.
Consider arbitrary η, γ, and γ' such that

– $\eta \models \Delta$, and
– $\gamma \approx \gamma' : \Gamma \mid \eta$.

We are required to show that $\gamma(\eta^1(e_1\ e_2)) \sim \gamma'(\eta^2(e_1\ e_2)) : \tau_2 \mid \eta$
$\equiv \gamma(\eta^1(e_1))\ \gamma(\eta^1(e_2)) \sim \gamma'(\eta^2(e_1))\ \gamma'(\eta^2(e_2)) : \tau_2 \mid \eta$.
Applying the induction hypothesis to $\Delta; \Gamma \vdash e_1 : \tau_1 \to \tau_2$, we have that $\Delta; \Gamma \vdash e_1 \sim e_1 : \tau_1 \to \tau_2$.
Instantiate the latter with η, γ, and γ'. Note that

– $\eta \models \Delta$, and
– $\gamma \approx \gamma' : \Gamma \mid \eta$.

Hence, $\gamma(\eta^1(e_1)) \sim \gamma'(\eta^2(e_1)) : \tau_1 \to \tau_2 \mid \eta$.
From the latter, there exist v_1 and v_1' such that

– $\gamma(\eta^1(e_1)) \to^* v_1$,
– $\gamma'(\eta^2(e_1)) \to^* v_1'$, and

– $v_1 \approx v_1' : \tau_1 \to \tau_2 \mid \eta$.

From the latter it follows that $v_1 = \lambda x{:}\eta^1(\tau_1).\, e_{11}$ and $v_1' = \lambda x{:}\eta^2(\tau_1).\, e_{11}'$.

Applying the induction hypothesis to $\Delta; \Gamma \vdash e_2 : \tau_1$, we have that $\Delta; \Gamma \vdash e_2 \sim e_2 : \tau_1$.
Instantiate the latter with η, γ, and γ'. Note that

– $\eta \models \Delta$, and
– $\gamma \approx \gamma' : \Gamma \mid \eta$.

Hence, $\gamma(\eta^1(e_2)) \sim \gamma'(\eta^2(e_2)) : \tau_1 \mid \eta$.
From the latter, there exist v_2 and v_2' such that

– $\gamma(\eta^1(e_2)) \to^* v_2$,
– $\gamma'(\eta^2(e_2)) \to^* v_2'$, and
– $v_2 \approx v_2' : \tau_1 \mid \eta$.

Instantiate $\lambda x{:}\eta^1(\tau_1).\, e_{11} \approx \lambda x{:}\eta^2(\tau_1).\, e_{11}' : \tau_1 \to \tau_2 \mid \eta$ (from above) with v_2 and v_2'. Note that

– $v_2 \approx v_2' : \tau_1 \mid \eta$.

Hence, $e_{11}[v_2/x] \sim e_{11}'[v_1'/x] : \tau_2 \mid \eta$.
From the latter, there exist v and v' such that

– $e_{11}[v_2/x] \to^* v$,
– $e_{11}'[v_2'/x] \to^* v'$, and
– $v \approx v' : \tau_2 \mid \eta$.

Thus, there exist v and v' such that

– $\gamma(\eta^1(e_1))\ \gamma(\eta^1(e_2)) \to^* v$ (from above by operational semantics),
– $\gamma'(\eta^2(e_1))\ \gamma'(\eta^2(e_2)) \to^* v'$ (from above by operational semantics), and
– $v \approx v' : \tau_2 \mid \eta$, which follows from above.

□

Exercise 1. Do the remaining cases of the proof. You may make use of the following lemmas:

Lemma 1. If $\eta \models \Delta$ and $\Delta \vdash \sigma$ and $R = \{\ (v, v') \mid\ v \approx v' : \sigma \mid \eta\ \}$,
then $v_1 \approx v_2 : \tau[\sigma/\alpha] \mid \eta$ iff $v_1 \approx v_2 : \tau \mid \eta[\alpha \mapsto (\eta^1(\sigma), \eta^2(\sigma), R)]$.

Lemma 2. If $v \approx v' : \tau \mid \eta$, then $\vdash v : \eta^1(\tau)$ and $\vdash v' : \eta^2(\tau)$.

(a) **Case:** $\dfrac{\Gamma(x) = \tau}{\Delta; \Gamma \vdash x : \tau}$ (Var)

(b) **Case:** $\dfrac{\Delta, \alpha; \Gamma \vdash e : \tau}{\Delta; \Gamma \vdash \Lambda\alpha.\, e : \forall\alpha.\, \tau}$ (Tabs)

(c) **Case:** $\dfrac{\Delta; \Gamma \vdash e : \forall\alpha.\, \tau \qquad \Delta \vdash \sigma}{\Delta; \Gamma \vdash e\,[\sigma] : \tau[\sigma/\alpha]}$ (Tapp)

Let us add product types $(\tau_1 \times \tau_2)$ to System F. The logical relation for the extended language is exactly as above, except that we must also define when values of product type are related:

$$\langle v_1, v_2 \rangle \approx \langle v_1', v_2' \rangle : \tau_1 \times \tau_2 \mid \eta \qquad \text{iff} \qquad v_1 \approx v_1' : \tau_1 \mid \eta \ \ \wedge \ \ v_2 \approx v_2' : \tau_2 \mid \eta$$

To complete the parametricity proof for the extended language, we need only show the following 3 cases of the parametricity proof pertaining to the intro and elim forms for products:

- **Case:** $\dfrac{\Delta; \Gamma \vdash e_1 : \tau_1 \qquad \Delta; \Gamma \vdash e_2 : \tau_2}{\Delta; \Gamma \vdash \langle e_1, e_2 \rangle : \tau_1 \times \tau_2}$ (Pair)
- **Case:** $\dfrac{\Delta; \Gamma \vdash e : \tau_1 \times \tau_2}{\Delta; \Gamma \vdash \mathsf{fst}\, e : \tau_1}$ (Fst)
- **Case:** $\dfrac{\Delta; \Gamma \vdash e : \tau_1 \times \tau_2}{\Delta; \Gamma \vdash \mathsf{snd}\, e : \tau_2}$ (Snd)

Exercise 2. For this exercise, we extend System F with sum types $\tau_1 + \tau_2$ and existential types $\exists\alpha.\, \tau$.

(a) Extend the logical relation defined above to handle sum types $\tau_1 + \tau_2$ and existential types $\exists\alpha.\, \tau$ (i.e., define when values of type $\tau_1 + \tau_2$ are related, and when values of type $\exists\alpha.\, \tau$ are related).

Complete the following cases of the parametricity proof:

(b) **Case:** $\dfrac{\Delta; \Gamma \vdash e : \tau_1}{\Delta; \Gamma \vdash \mathsf{inl}\, e : \tau_1 + \tau_2}$ (Inl)

(c) **Case:** $\dfrac{\Delta; \Gamma \vdash e : \tau_1 + \tau_2 \qquad \Delta; \Gamma, x : \tau_1 \vdash e_1 : \tau \qquad \Delta; \Gamma, y : \tau_2 \vdash e_2 : \tau}{\Delta; \Gamma \vdash \mathsf{case}\, e\, \mathsf{of}\, \mathsf{inl}\, x \Rightarrow\ e_1 \mid \mathsf{inr}\, y \Rightarrow\ e_2 : \tau}$ (Case)

(d) **Case:** $\dfrac{\Delta \vdash \sigma \qquad \Delta; \Gamma \vdash e : \tau[\sigma/\alpha]}{\Delta; \Gamma \vdash \mathsf{pack}\, \sigma, e\, \mathsf{as}\, \exists\alpha.\, \tau : \exists\alpha.\, \tau}$ (Pack)

(e) **Case:** $\dfrac{\Delta; \Gamma \vdash e_1 : \exists\alpha..\tau \qquad \Delta \vdash \tau_2 \Delta, \alpha; \Gamma, x : \tau \vdash e_2 : \tau_2}{\Delta; \Gamma \vdash \mathsf{unpack}\, e_1\, \mathsf{as}\, \alpha, x\, \mathsf{in}\, e_2 : \tau_2}$ (Unpack)

2 Representation Independence, Bisimilarity

Parametricity is the essence of *representation independence* for abstract types.

A relation $R : \tau_1 \leftrightarrow \tau_2$ is a *bisimulation* between two packages of type $\exists\alpha.\,\tau$: $\mathsf{pack}\,\tau_1, v_1\ \mathsf{as}\ \exists\alpha.\,\tau$ and $\mathsf{pack}\,\tau_2, v_2\ \mathsf{as}\ \exists\alpha.\,\tau$ if and only if:

$$v_1 \approx v_2 : \tau \mid \emptyset[\alpha \mapsto (\tau_1, \tau_2, R)]$$

When two packages e_1 and e_2 are *bisimilar*, they are indistinguishable by any client — that is, the execution of the client cannot depend on whether e_1 or e_2 is the provided package.

In other words, if $\alpha; x : \tau \vdash e_c : \tau_c$, then

$$\mathsf{unpack}\,e_1\ \mathsf{as}\ \alpha, x\ \mathsf{in}\ e_c \sim \mathsf{unpack}\,e_2\ \mathsf{as}\ \alpha, x\ \mathsf{in}\ e_c : \tau_c \mid \emptyset$$

Exercise 3. Show that the following two implementations of counters are bisimilar.

$$\begin{aligned}
\mathsf{Counter} &= \exists\alpha.\,\{\mathsf{new} : \alpha,\\
&\qquad\quad \mathsf{inc} : \alpha \to \alpha,\\
&\qquad\quad \mathsf{get} : \alpha \to \mathsf{int}\}
\end{aligned}$$

$$\begin{aligned}
\mathsf{c1} &= \{\mathsf{new} = 0,\\
&\quad\ \mathsf{inc} = \lambda x : \mathsf{int}.\ x + 1,\\
&\quad\ \mathsf{get} = \lambda x : \mathsf{int}.\ x\}\\
\mathsf{cntr1} &= \mathsf{pack\ int, c1\ as\ Counter}
\end{aligned}$$

$$\begin{aligned}
\mathsf{c2} &= \{\mathsf{new} = 0,\\
&\quad\ \mathsf{inc} = \lambda x : \mathsf{int}.\ x - 1,\\
&\quad\ \mathsf{get} = \lambda x : \mathsf{int}.\ 0 - x\}\\
\mathsf{cntr2} &= \mathsf{pack\ int, c2\ as\ Counter}
\end{aligned}$$

To show that cntr1 and cntr2 are bisimilar, it suffices to give a relation $R : \mathsf{int} \leftrightarrow \mathsf{int}$ and show that R is such that the implementations of new, inc and get in both packages are equivalent relative to R. That is, show:

(a) $\mathsf{c1.new} \sim \mathsf{c2.new} : \alpha \mid \emptyset[\alpha \mapsto (\mathsf{int}, \mathsf{int}, R)]$

(b) $\mathsf{c1.inc} \sim \mathsf{c2.inc} : \alpha \to \alpha \mid \emptyset[\alpha \mapsto (\mathsf{int}, \mathsf{int}, R)]$

(c) $\mathsf{c1.get} \sim \mathsf{c2.get} : \alpha \to \mathsf{int} \mid \emptyset[\alpha \mapsto (\mathsf{int}, \mathsf{int}, R)]$

Exercise 4. Given the relation $R = \{(l, (b, f)) \mid l = \texttt{append}\ b(\texttt{reverse}\ f)\}$ (where $R : \texttt{intlist} \leftrightarrow (\texttt{intlist} * \texttt{intlist})$), show that the following two implementations of queues are bisimilar. As for the last exercise, you can show this fact by showing that the implementations of all the operations are equivalent relative to R. Note: two lists are related if all of their elements are related.

```
signature Queue =
sig
  type queue
  empty : queue
  insert : int -> queue -> queue
  remove : queue -> (int * queue)
end

structure Q1
struct
  type queue = int list
  val empty = nil
  fun insert (x,xs) = x::xs
  fun remove xs =
    let val (x,xs) = rev xs in (x, rev xs) end
end

structure Q2
struct
  type queue = int list * int list
  val empty = (nil, nil)
  fun insert (x, (bs,fs)) = (x::bs, fs)
  fun remove (bs, nil) = remove (nil, rev bs)
    | remove (bs, f::fs) = (f, (bs,fs))
end
```

Note that the SML signature for queues above may be written as follows:

$$\begin{aligned}\mathsf{Queue} \quad = \quad \exists\alpha. \{&\mathsf{empty} : \alpha,\\ &\mathsf{insert} : \mathsf{nat} \to \alpha \to \alpha,\\ &\mathsf{remove} : \alpha \to (\mathsf{int} \times \alpha)\}\end{aligned}$$

3 Subtyping

Read TAPL Section 15.1, 15.2, and pages 196–198 (Variants, Lists, References).

Exercise 5. For each of the following questions, answer Yes or No. If the answer is Yes, show the subtyping derivation. If the answer is No, give either a *term* that demonstrates how type safety breaks if we allow the two types in the subtype relation, or a *short explanation* of why type safety is preserved even if we allow the two types in the subtype relation.

(a) Is $\{\texttt{x} : \texttt{Top} \to \texttt{Ref Top}\}$ a subtype of $\{\texttt{x} : \texttt{Top} \to \texttt{Top}\}$?

(b) Is $\{\texttt{x} : \texttt{Top} \to \texttt{Ref Top}\}$ a subtype of $\{\texttt{x} : \texttt{Ref Top} \to \texttt{Ref}\ \{\texttt{y} : \texttt{Top}\}\}$?

(c) Is $\{\texttt{x} : \texttt{Ref}\ \{\texttt{y} : \texttt{Top}\}\}$ a subtype of $\{\texttt{x} : \texttt{Ref Top}\}$?

(d) Is $\{\texttt{x} : \texttt{Top}\}$ a subtype of $\{\texttt{x} : \{\,\}\}$?