CIS 500 — Software Foundations

Homework Assignment 11

Objects in the lambda-calculus

Due: Friday, December 8, 2006, by noon

1 Exercise Write down a careful proof of the Preservation Theorem for FJ (19.5.1 in TAPL). You may use, without proof, the following Substitution Lemma for FJ:

> **Lemma [Term substitution preserves typing]:** If $\Gamma, \overline{\mathtt{x}} : \overline{\mathtt{B}} \vdash \mathtt{t} : \mathtt{D}$ and $\Gamma \vdash \overline{\mathtt{s}} : \overline{\mathtt{A}}$, where $\overline{\mathtt{A}} <: \overline{\mathtt{B}}$, then $\Gamma \vdash [\overline{\mathtt{x}} \mapsto \overline{\mathtt{s}}]\mathtt{t} : \mathtt{C}$ for some $\mathtt{C} <: \mathtt{D}$.

2 Exercise The Progress Theorem for FJ involves a notion of *evaluation contexts*, which are intended to capture the "next subterm to be reduced" by the evaluation relation. This intention can be formalized by the following claims:

Claim 1: If $\mathtt{r} \longrightarrow \mathtt{r}'$ by one of the computation rules E-ProjNew, E-InvkNew, or E-CastNew and E is an arbitrary evaluation context, then $E[\mathtt{r}] \longrightarrow E[\mathtt{r}']$ by the ordinary evaluation relation.

Claim 2: If $\mathtt{t} \longrightarrow \mathtt{t}'$ by the ordinary evaluation relation, then there are unique E, $\mathtt{r}$, and $\mathtt{r}'$ such that

1. $\mathtt{t} = E[\mathtt{r}]$,
2. $\mathtt{t}' = E[\mathtt{r}']$, and
3. $\mathtt{r} \longrightarrow \mathtt{r}'$ by one of the computation rules E-ProjNew, E-InvkNew, or E-CastNew.

Prove these claims.

3 Exercise Although mutable state has been omitted from FJ for simplicity, it is not too hard to modify the definitions to allow it. Carry out this extension as follows (there are **four** things you need to turn in):

1. First, we need to add a new syntactic form for *assignment* to the field of an object:

$\mathtt{t}$	::=	...	*terms*
		$\mathtt{t}_0\mathtt{.f = t}_1\mathtt{; t}_2$	*assignment and sequencing*

   The intended meaning is that we assign the value of $\mathtt{t}_1$ to the field `f` of the object denoted by $\mathtt{t}_0$ and then continue evaluating $\mathtt{t}_2$. (Merging the assignment and sequencing constructs into a single syntactic form avoids having to introduce a `unit` value or in some other way deciding what is to be the "result value" for an assignment expression.)

   Task 1: Write out the typing rule for this new syntactic form.

2. Next, we add a store to the operational semantics of FJ programs:

(a) We add *locations* to the syntax of terms (as we did for references in the simply typed lambda-calculus).

(b) We add to the evaluation relation a store mapping *locations* to *store objects* of the form `new C(`$\bar{l}$`)`, where $\bar{l}$ is a sequence of locations storing the contents of the fields of the object.

(c) We change the definition of FJ *values* so that all values are locations (not `new` expressions).

We now need to add a new computation rule for `new`: since `new` expressions are no longer values, we need a rule showing how a `new` expression (whose arguments have all been reduced to values) evaluates to a fresh location value. This rule needs to

(a) choose a fresh location l for the new object,

(b) update the store with an appropriate binding, and

(c) yield l as its result.

Task 2: Write out this rule.

3. Finally, we need to make appropriate changes to the other evaluation rules. The congruence rules don't change significantly, e.g., E-Cast becomes

$$\frac{\mathtt{t}_0 \mid \mu \longrightarrow \mathtt{t}_0' \mid \mu'}{\mathtt{(C)t}_0 \mid \mu \longrightarrow \mathtt{(C)t}_0' \mid \mu'} \text{ E-Cast}$$

However, each of the computation rules must be reformulated so that, instead of dealing directly with objects (`new` expressions) they deal with locations pointing to store objects. For example, the left-hand side of the field access rule E-ProjNew will be $l\mathtt{.f}_i$ instead of `(new C(`$\overline{\mathtt{v}}$`)).f`$_i$.

Task 3: Write out the new versions of the rules E-ProjNew, E-InvkNew, and E-CastNew.

4. Finally, we need to check that appropriate versions of progress and preservation continue to hold.

Task 4: Write out the *statements* of the progress and preservation properties for the extended language. (You don't need to include proofs. But it's a good idea to think through a few cases of the proof, as a sanity check on your definitions.)

4 Exercise [Required for PhD groups; optional for others.] TAPL exercise 19.4.2. (Your answer should be less than a page long; the purpose is just to think a little about design alternatives, not to actually carry out an alternative design in detail.)

5 Debriefing

1. Approximately how many hours (per person, on average) did you spend on this assignment?
2. Would you rate it as easy, moderate, or difficult?
3. How deeply do you feel you understand the material it covers (0%–100%)?
4. Any other comments?