— Step-indexing continued

Appel-McAllester (2001) proposed the original idea of step-indexing; originally called indexing. They dealt with recursive and polymorphic types. (The paper is on the web page.) It was originally part of their foundational proof-carrying code project: They wanted to semantically model types as sets of machine programs. Step-indexing let them semantically model recursive types.

Even though they were talking about applying high-level types to low-level code, they wanted to avoid a syntactic typing discipline. (For reasons Derek has never completely understood. Mainly to make things more “foundational”.)

They came up with a good idea: How to handle general recursive types. The method they developed ended up having nothing to do with machine programs. (The fact that it works for that level is quite interesting.)

They didn't want to get involved in the more denotational approaches to modelling recursive types.

They weren't doing relational reasoning. They weren't reasoning about equivalences. They just wanted to define a (unary) model of types such that if a machine program was in that model, then it satisfied some safety properties.

Amal (2006) showed how to use (basically) the same approach to reason about equivalences of high-level programs. She dealt with recursive and polymorphic types. Arguably, it's a completely different application, using the same sort of model, except binary instead of unary. This binary approach was conjectured as useful in Appel-McAllester.

Interestingly, she did this after her 2005 thesis work, a unary step-indexed model for higher-order state with recursive and polymorphic types. (Just safety.) To Derek, Amal's thesis was more interesting technically. On a conceptual level, her 2006 work was an equally important result.

It wasn't until Ahmed-Dreyer-Rossberg (2009) dealt with relational reasoning about recursive, polymorphic, and general reference types.

What was suprising at the time of Ahmed's 2006 paper: People who

originally saw the Appel-McAllester model viewed it as a “nice hack”. Since you're counting steps, this couldn't possible be used to reason about observational equivalence and extensional reasoning. In 2006, Ahmed handled several examples (possibly all of the examples) from Sumii-Pierce (2005), a paper on using bisimulations to reason about recursive and polymorphic types. She basically showed you could use an “inductive” method like logical relations to mirror such work. Moreover, she proved wrong the hemming and hawing about the “intensional” technique of step-indexing.

We want to define the relation on terms. There are several very different ways. We'll follow Ahmed 2006. It seems a bit annoying: It looks like you have to count and add up steps. It looks a bit hacky. That doesn't have to be the case. She just happened to have set it up that way.

Once we combine step-indexing with biorthogonality, we'll see it actually simplifies certain things.

We'll work with her approach. We'll see that certain reasoning about equivalences that change the control flow are possible in her original model but not in the model with biorthogonality.

— Logical approximation

Recall from last time:

$$\text{Cand} = \{ R \in \text{Sub}(\mathbb{N} \times \text{CVal} \times \text{CVal}) \mid \forall (n,v_1,v_2) \in R.\ \forall j<n.\ (j,v_1,v_2) \in R \}$$

$$V[\sigma \to \tau]\rho := \{ (n,\lambda x.e_1,\lambda x.e_2) \mid \forall j \leq n.\ \forall v_1,v_2.\ (j,v_1,v_2) \in V[\sigma]\rho \implies (j,e_1[v_1/x],e_2[v_2/x]) \in E[\tau]\rho \}$$

$$V[\mu\alpha.\tau]\rho := \{ (n,\text{fold } v_1,\text{fold } v_2) \mid \forall j<n.\ (j,v_1,v_2) \in V[\tau[\mu\alpha.\tau/\alpha]]\rho \}$$

Here's what we'll do for terms:

Recall the rough (so rough its wrong) idea:

> Two programs are related if you drop them in any context and run that program for n steps of computation, then you won't end up in a situation where one terminates and the other will diverge.

[In class, we kicked around some ideas.]

The following idea is almost right.

If e_1 goes to a value v_1 in under n steps,
then e_2 goes to a value v_2
and the values are related.
(Plus the symmetric thing.)

Derek: You need to define the logical relation in an asymmetric way. We define logical approximation asymmetrically. We'll define equivalence later.

$$E[\tau]\rho := \{ (n,e_1,e_2) \mid \forall k<n,\ e'_1.\ e_1 \downarrow k\ e'_1 \implies \exists v_2.\ e_2 \downarrow v_2 \wedge (n-k, e'_1, v_2) \in V[\tau]\rho \}$$

Here's a key point: We can't write $e_1 \downarrow k\ v_1$ here. That leaves the relation too weak. We have to show that e'_1 ends up being a value. We want to rule out stuck terms. We want to build a relation that includes within it the notion that the terms are well-behaved. We don't want a stuck e_1 to be trivially related to any e_2.

Ahmed used $e_1 \mapsto k\ e'_1 \wedge \text{irred}(e_1)$ where $\text{irred}(e_1)$ means "not stuck".

We define

$$e \downarrow k\ e' :\Longleftrightarrow e_1 \mapsto k\ e'_1 \wedge \text{irred}(e_1)$$

Aside: The idea of defining the relation all at once—rather than via logical approximation—did not work when Amal and Derek tried to make it work one afternoon. They couldn't build a sound model.

Aside: Derek thinks we might be able to put stuck terms in the model (and not suffer as a result).

Observe, that if e_1 doesn't terminate, it approximates everything:

$$\forall n,e.\ (n,\bot,e) \in E[\tau]\rho.$$

Aside: We set up our base case with values in canonical form (e.g., $(n,\lambda x.e_1, \lambda x.e_2) \in V[\sigma\to\tau]$). They correspond to k=0 in $E[\sigma\to\tau]$.

To reason about logical equivalence, we'll use logical approximation in both directions.

Let's lift our logical approximation to open terms.

$$D[\Delta] := \{ \rho \in \Delta \to \text{Cand} \}$$
$$G[\Gamma]\rho := \{ (n,\gamma_1,\gamma_2) \mid \forall (x{:}\tau) \in \Gamma.\ (n,\gamma_1 x,\gamma_2 x) \in V[\tau]\rho \}$$

$\Delta; \Gamma \vdash e_1 \precsim e_2 : \tau :\Longleftrightarrow$

$\Delta; \Gamma \vdash e_1 : \tau \wedge \Delta; \Gamma \vdash e_2 : \tau \wedge$
$\forall \rho \in D[\Delta]. \forall (n, \gamma_1, \gamma_2) \in G[\Gamma]\rho.\ \forall \delta_1, \delta_2 \in \Delta \to \mathrm{CTyp}.$
$(n, \delta_1\gamma_1 e_1, \delta_2\gamma_2 e_2) \in E[\tau]\rho.$

What's changed from our earlier models:
We're defining the relation asymmetrically.
We're carrying around step indices.

— Metatheory

What we want to show is that our logical approximation is contained in contextual approximation. Contextual approximation is just the one-sided version of contextual equivalence. See Ahmed 2006.

Consider the Pitts-style definition of contextual equivalence we used. Contextual approximation no longer satisfies symmetry. It still must satisfy all the compatibility lemmas. The difference is in Adequacy. It will look like:
$\vdash e_1 \precsim e_2 : \tau \wedge e_1{\downarrow} \Longrightarrow e_2{\downarrow}$.
That is, it's half of adequacy. Later, we will see how this plays out in a Pitts-style typed LR.

We want to show $\precsim$ is adequate and compatible. We'll do a few cases. (Ahmed handles them all in her technical report.)

NB we will need downard closure for $V[\tau]\rho$. It's easy to prove. We quantified over smaller indices when we defined $V[\sigma \to \tau]$, so downard closure there is trivial and does not need downard closure of $E[\tau]$. We happened to define $V[\forall \alpha.\tau]$ in terms of $E[\tau]$ at the same step-index. (We could bake it into every part of $V[\tau]\rho$, but that makes proving stuff using the LR more annoying. It's cleaner to bake in just enough.)

Aside from Deepak: In more complicated models, we do not necessarily know that $E[\tau]\rho$ is monotone.

Lemma (Adequacy):
If $\vdash e_1 \precsim e_2 : \tau$
and $e_1{\downarrow}$,
then $e_2{\downarrow}$.
Pf:

Suppose $\vdash e_1 \precsim e_2 : \tau$
$\Rightarrow$ $\forall n.(n,e_1,e_2) \in E[\tau]$.

Suppose $\vdash e_1 \downarrow$
$\Rightarrow$ $\exists k.\ e_1 \downarrow k\ e'_1$.

(Aside: We wrote e'_1 for fun. We know its a value by syntactic means: e_1 is well-typed.)

TS: $e_2\downarrow$.

Pick $n>k$
$\Rightarrow$ $(n,e_1,e_2) \in E[\tau]$
$\Rightarrow (k<n)$
$e_2\downarrow$.

(Aside: Its important that they are related for all steps. We needed to pick $n>k$.)
Q.E.D.

Lemma (Compatibility for functions):
$\Delta; \Gamma, x{:}\sigma \vdash e_1 \precsim e_2 : \tau$
—
$\Delta; \Gamma \vdash \lambda x.e_1 \precsim \lambda x.e_2 : \sigma \to \tau$.

Proof:
Let $\rho \in D[\Delta]$ and $(n,\gamma_1,\gamma_2) \in G[\Gamma]\rho$ and $\delta_1,\delta_2 \in \Delta \to CTyp$.
TS: $(n,\delta_1\gamma_1(\lambda x.e_1),\delta_2\gamma_2(\lambda x.e_2)) = (n,\lambda x.\delta_1\gamma_1 e_1,\lambda x.\delta_2\gamma_2 e_2) \in V[\sigma\to\tau]\rho$.
Let $k \le n$. Suppose $(k,v_1,v_2) \in V[\sigma]\rho$.
TS: $(k,\delta_1\gamma_1 e_1[v_1/x],\delta_2\gamma_2 e_2[v_2/x]) \in E[\tau]\rho$.
Set $\gamma'\_i := \gamma\_i, x \mapsto v\_i$.
STS: $(k,\delta_1\gamma'_1 e_1,\delta_2\gamma'_2 e_2) \in E[\tau]\rho$.
We want to instantiate the premise with γ'_1,γ'_2.
To do that, we must show $(k,\gamma'_1,\gamma'_2) \in G[\Gamma,x{:}\sigma]\rho$.
This follows from downward closure: $(n,\gamma_1,\gamma_2) \Rightarrow (k,\gamma_1,\gamma_2)$.
Q.E.D.

Many things remain easy. For example, LR inclusion.

Before going on, we want the bind lemma. That means we have to figure out how to state it in this setting.

Recall what we did in symmetric models:
If $(e_1,e_2) \in E[\tau]\rho$

and $\forall(v_1,v_2) \in V[\tau]\rho.\ (K_1[v_1],K_2[v_2]) \in E[\tau']\rho'$,
then $(K_1[e_1],K_2[e_2]) \in E[\tau']\rho'$.

Lemma (Bind for closed terms):
If $(n,e_1,e_2) \in E[\tau]\rho$
and $\forall j \le n.\ \forall v_1,v_2.\ (j,v_1,v_2) \in V[\tau]\rho \implies (j,K_1[v_1],K_2[v_2]) \in E[\tau']\rho'$,
then $(n,K_1[e_1],K_2[e_2]) \in E[\tau']\rho'$.
Proof: HW for Tuesday.

Question: Why didn't we have to be strict with the step-indices?
Answer:
Derek prefers to use $<$ only where its absolutely necessary
(for us, that means in $V[\mu\alpha.\tau]$).

Lemma (Bind for open terms):
$\Delta; \Gamma \vdash e_1 \precsim e_2 : \sigma$
$\Delta; \Gamma, x{:}\sigma \vdash K_1[x] \precsim K_2[x] : \tau$
$x \notin fv(K_1,K_2)$
—
$\Delta; \Gamma \vdash K_1[e_1] \precsim K_2[e_2] : \tau$
Proof: HW for Tuesday.
Discussion: You'll be given some n (your goal). It reduces by the bind lemma for closed terms to showing
$\forall j \le n.\ \forall v_1,v_2.\ (j,v_1,v_2) \in V[\tau]\rho \implies (j,K_1[v_1],K_2[v_2]) \in E[\tau']\rho'$.

To do that, you need to use downward closure so you can extend the context. (This part of the argument is quite similar to compatibility for $V[\sigma\to\tau]\rho$.)

Aside: The logic Derek, Amal, and Lars developed effectively merges these two versions of the bind lemma together. You can reason about specific ρ's with inference rules. We'll discuss this next week.

Lemma (Compatibility for applications):
$\Delta; \Gamma \vdash e_1 \precsim e_2 : \sigma \to \tau$
$\Delta; \Gamma \vdash e'_1 \precsim e'_2 : \sigma$
—
$\Delta; \Gamma \vdash e_1\ e'_1 \precsim e_2\ e'_2 : \tau$
Proof:
Using the open bind lemma, this reduces (exactly as before) to showing

$\Delta; \Gamma, x{:}\sigma\to\tau, x'{:}\sigma \vdash x\ x' \precsim x\ x' : \tau$

$\Leftarrow$

$\forall n, v_1, v_2, v'_1, v'_2.$
$(n, v_1, v_2) \in V[\sigma\to\tau]\rho \wedge$
$(n, v'_1, v'_2) \in V[\sigma]\rho$
$\Rightarrow (n,\ v_1\ v'_1,\ v_2\ v'_2) \in E[\tau]\rho.$

Unrolling $E[\tau]\rho$, let $j \le n$ and assume there exists e_1 satisfying
$v_1\ v'_1 \downarrow j\ e_1$.
STS: $v_2\ v'_2 \downarrow e_2 \wedge (n-j, e_1, e_2) \in V[\tau]\rho$.

By the definition of $V[\sigma\to\tau]\rho$,
WK: $\exists e'_1, e'_2.\ v_1 = \lambda x.e'_1 \wedge v_2 = \lambda x.e'_2 \wedge$
1. $(n, e'_1[v'_1/x], e'_2[v'_2/x]) \in E[\tau]\rho$.

WK: $v_1\ v'_1 \mapsto e'_1[v'_1/x] \downarrow\{j-1\}\ e_1 \wedge$
$v_2\ v'_2 \mapsto e'_2[v'_2/x] \wedge$
$j-1 < n$.
$\Rightarrow$(1)
$v_2\ v'_2 \mapsto e'_2[v'_2/x] \downarrow e_2 \wedge$
$(n-j+1, e_1, e_2) \in V[\tau]\rho$
$\Rightarrow$ $v_2\ v'_2 \downarrow e_2 \wedge (n-j, e_1, e_2) \in V[\tau]\rho$.
Q.E.D.

Aside: The only steps you really need to count in the E relation are the unfold/fold steps. (We'll see this when we discuss the logic paper.)

Using the bind lemma, we can reduce compatibility for fold to
$\Delta; \Gamma, x{:}\ \tau[\mu\alpha.\tau/\alpha] \vdash \text{fold}\ x \precsim \text{fold}\ x : \mu\alpha.\tau$.

Lemma (Compatibility for fold):
Suppose $(n, v_1, v_2) \in V[\tau[\mu\alpha.\tau/\alpha]]\rho$.
Show $(n, \text{fold}\ v_1, \text{fold}\ v_2) \in V[\mu\alpha.\tau]\rho \subseteq E[\mu\alpha.\tau\rho]$.
Proof:
STS: $\forall j<n.\ (j, v_1, v_2) \in V[\tau[\mu\alpha.\tau/\alpha]]\rho$.
This follows from downard closure.
(Derek expected to do more work. In Amal's proofs, she didn't use the Bind lemma. She baked it into each of her compatibility proofs.)
Q.E.D.

Lemma (Compatibility for unfold):
Suppose $(n, v_1, v_2) \in V[\tau[\mu\alpha.\tau/\alpha]]\rho$.

Show $(n,\text{unfold } v_1,\text{unfold } v_2) \in E[\tau[\mu\alpha.\tau/\alpha]]\rho$.

Proof:

Let $j<n$ and assume there exists e_1 s.t.

$\text{unfold } v_1 \downarrow j\ e_1$.

TS: $\exists e_2.\ \text{unfold } v_2 \downarrow e_2 \wedge$

$(n-j,e_1,e_2) \in V[\tau[\mu\alpha.\tau/\alpha]]\rho$.

The first assumption implies

$v_1 = \text{fold } v'_1 \wedge v_2 = \text{fold } v'_2 \wedge$

$\forall j'<n.\ (j',v'_1,v'_2) \in V[\tau[\mu\alpha.\tau/\alpha]]\rho$.

There's only one step of reduction to go from these folds to those unfolds.

WK: $e_1 = v'_1 \wedge e_2 = v'_2 \wedge j = 1$.

STS: $(n-1,v'_1,v'_2) \in \tau[\mu\alpha.\tau/\alpha]]\rho$.

Pick $j' := n-1 < n$ and we're done.

Q.E.D.

The rest of the compatibility lemmas will be very similar.

— Admissibility

We never talked about admissibility. We needed it to prove recursive types related. But once we have recursive types and functions, we can encode fixed points in the language. A coinductive proof technique for fixed points fall out from the step-indexed logical relation.

We've taken these finite approximations, limited to the admissibility lemma, and pervaded them through the entire model. What we've built is “something like” a relation on the nth finite approximation.

One of the complaints we can make about this model: It gets annoying to prove things with it because we must deal with these step indices even when we don't want to.

Admissibility isn't baked into the model. What's baked in is that we're always relating finite approximations. We never prove that things are related “in the limit” and that suffices to prove contextual approximation.