Modified: 20121206

Today we'll talk about $\top\top$-closure, update the model, and actually prove what we sketched last time.

Let's motivate one thing that came up in the definition of $\top\top$-closure but wasn't motivated.

In the setting of System F, adequacy said that if two things are logically related, then they're observably equivalent. With recursion, we can get by with a somewhat simpler notion of observation that relies only on termination.

Definition (Cotermination):
$e_1 \downarrow\downarrow e_2 :\Longleftrightarrow (e_1\downarrow \wedge e_2\downarrow) \vee (e_1\uparrow \wedge e_2\uparrow)$

Typically, cotermination is the notion of adequacy we'll use in this course (and researchers use when working with such models).

Adequacy:
If $\vdash e_1 \approx e_2 : \tau$,
then $e_1 \downarrow\downarrow e_2$.

Why is looking at cotermination sufficient? It covers all of the distinguishing contexts you might care about. Suppose there existed some context C s.t. $C[e_1]\downarrow$true while $C[e_2]\downarrow$false. Clearly, they both terminate. But we can transform C into a context that distinguishes e_1 and e_2. Construct

C' := if C then () else $\bot$

where $\bot$:= (fix f(x).f(x))(). Then $C'[e_1]\downarrow$ but $C'[e_2]\uparrow$.

This is nice. We can observe equivalence at all types by just looking at termination. (Compare to our previous notion of adequacy.)

— Motivating $\top\top$-closure

Recall what we said last time:

• Two terms are related if they behave the same in related continuations.

• Two continuations are related if they behave the same when applied

to related values.

• Two values are related in the usual way (but now referring to our new term relation).

$$
\begin{aligned}
&E[\tau]\rho := \{ (e_1,e_2) \mid \forall (K_1,K_2)\in K[\tau]\rho.\ K_1[e_1] \downarrow\downarrow K_2[e_2] \} \\
&K[\tau]\rho := \{ (K_1,K_2) \mid \forall (v_1,v_2)\in V[\tau]\rho.\ K_1[v_1] \downarrow\downarrow K_2[v_2] \} \\
&V[\alpha]\rho := \rho(\alpha) \\
&V[\sigma\to\tau]\rho := \{ (v_1,v_2) \mid \forall (v'_1,v'_2) \in V[\sigma]\rho.\ (v_1\ v'_1,\ v_2\ v'_2) \in E[\tau]\rho \} \\
&V[\forall\alpha.\tau]\rho := \{ (v_1,v_2) \mid \forall \sigma_1,\sigma_2 \in \mathrm{CTyp}.\ \forall R \in \mathrm{Cand}. \\
&\qquad (v_1\ \sigma_1,\ v_2\ \sigma_2) \in E[\tau](\rho,\alpha\mapsto R) \} \\
&V[\tau'\times\tau'']\rho := \{ ((v'_1,v''_1), (v'_2,v''_2)) \mid \\
&\qquad (v'_1,v'_2) \in V[\tau']\rho \wedge (v''_1,v''_2) \in V[\tau'']\rho \} \\
&V[\tau'+\tau'']\rho := \{ (\mathrm{inl}\ v_1,\mathrm{inl}\ v_2) \mid (v_1,v_2) \in V[\tau']\rho \} \\
&\qquad \cup \{ (\mathrm{inr}\ v_1,\mathrm{inr}\ v_2) \mid (v_1,v_2) \in V[\tau'']\rho \} \\
&V[\exists\alpha.\tau]\rho := \{ (\mathrm{pack}\ [\sigma_1,v_1]\ \mathrm{as}\ \exists\alpha.\tau_1,\ \mathrm{pack}\ [\sigma_2,v_2]\ \mathrm{as}\ \exists\alpha.\tau_2) \mid \\
&\qquad \exists R\in\mathrm{Cand}.\ (v_1,v_2) \in V[\tau]\rho,\alpha\mapsto R \}
\end{aligned}
$$

It's not terribly obvious why this definition gives you an admissible term relation. Why does it help with recursion?

Intuition: It really makes sense in languages where you cannot understand the behavior of terms separate from their contexts.

> Consider a langauge with first-class continuations or sections. You don't necessarily have a way of doing that.
>
> Consider a language with low-level features: We can relate high- and low-level (assembler) languages using logical relations. You need some notion of the whole machine to understand what's going on at the low-level.

So the idea is to first complete a term to a whole program by plugging it into a continuation. You can then talk about the program's observable properties.

From Neel: When you take the union of admissible sets, you don't get an admissible set. OTOH, $\top\top$-closed sets are closed under union.

> (Derek and Neel will talk about this point and present it for general consumption. Had we tried to prove the relational actions of our relations all preserved admissibility, we would have done fine for $\forall$ and $\to$, but we'd have failed.)

— Adapting our metatheory

Some of our lemmas remain trivial:

Type substitution ($V[\tau[\sigma/\alpha]]\rho = V[\tau]\rho,\alpha\mapsto V[\sigma]\rho$).
Irrelevance ($V[\tau]\rho = V[\tau]\rho'$ if ρ, ρ' match on $ftv(\tau)$).
Validity ($V[\tau]\rho \in Cand$).

Others stop being completely obvious.

Lemma (Value inclusion):

$V[\tau]\rho \subseteq E[\tau]\rho$.

Proof:

Let $(v_1,v_2)\in V[\tau]\rho$.
Let $(K_1,K_2)\in K[\tau]\rho$.
TS: $K_1[v_1]\downarrow\downarrow K_2[v_2]$.
Follows by unrolling the definition of $K[\tau]\rho$.

Q.E.D.

Lemma (Divergence):

If $e_1\uparrow$
and $e_2\uparrow$,
then $(e_1,e_2) \in E[\tau]\rho$.

Proof:

Let $(K_1,K_2)\in K[\tau]\rho$.
TS: $K_1[e_1]\downarrow\downarrow K_2[e_2]$.
By a case analysis on K_1,K_2
STS: $e_1\uparrow \wedge e_2\uparrow$.
The point is both e_1 and e_2 are in the respective evaluation positions, so both programs diverge.

Q.E.D.

Aside: Derek will follow with a comparison with Pitts' setup. Derek thinks its over-complicated.

Lemma (Closure under expansion):

If $(e_1,e_2) \in E[\tau]\rho$
and $e'_1 \mapsto^* e_1$
and $e'_2 \mapsto^* e_2$,
then $(e'_1,e'_2) \in E[\tau]\rho$.

Proof:

Let Let $(K_1,K_2)\in K[\tau]\rho$.
TS: $K_1[e'_1]\downarrow\downarrow K_2[e'_2]$.

Then $K_1[e'_1] \mapsto^* K_1[e_1]$

and $K_2[e'_2] \mapsto^* K_2[e_2]$.

STS: $K_1[e_1] \downarrow\downarrow K_2[e_2]$
$\Leftarrow$ $(e_1,e_2) \in E[\tau]\rho$.
Q.E.D.

Question: We're assuming deterministic reduction. Does this setup scale to non-deterministic reduction?

Answer: Our observation is cotermination. It's fine if you mix cotermination with the possibility that either term diverges. (You've less to prove.) A good question: You now have a notion "may be related".

Lemma (Adequacy):
If $\vdash e_1 \approx e_2 : \tau$,
then $e_1 \downarrow\downarrow e_2$.
Proof:
WK: $\vdash e_1 \approx e_2 : \tau$
$\Rightarrow$ $(e_1,e_2) \in E[\tau]\emptyset$
$\Leftrightarrow$ $\forall (K_1,K_2) \in K[\tau]\emptyset.\ K_1[e_1] \downarrow\downarrow K_2[e_2]$ }.

STS: $(\cdot,\cdot) \in K[\tau]\emptyset$
$\Rightarrow$ $\cdot[e_1] \downarrow\downarrow \cdot[e_2]$ }
$\Leftrightarrow$ $e_1 \downarrow\downarrow e_2$.

Let $(v_1,v_2) \in V[\tau]\emptyset$.
TS: $\cdot[v_1] \downarrow\downarrow \cdot[v_2]$
$\Leftrightarrow$ $v_1 \downarrow\downarrow v_2$.
Q.E.D.

We'll have another useful lemma for doing compatibility proofs. These should suffice for proving admissibility.

Notation (Unrollings):
For any $F = \text{fix } f(x).e$
satisfying $\vdash F : \sigma \to \tau$,
write
$F_0 := \text{fix } f(x).f(x)$
$F_{n+1} := \text{fix } \_(x).e[F\_n/f]$.

Theorem (Admissibility):
If $\forall n.\ (e[F\_n/f],e'[F'\_n/f]) \in E[\tau]\rho$,
then $(e[F/f],e'[F'/f]) \in E[\tau]\rho$.

Proof:

Let $(K,K') \in K[\tau]\rho$.

TS: $K[e[F/f]] \downarrow\downarrow K'[e'[F'/f]]$.

WK: $\forall n.\ K[e[F\_n/f]] \downarrow\downarrow K'[e'[F'\_n/f]]$.

Consider one direction.

STS: $K[e[F/f]]\downarrow \Rightarrow K'[e'[F'/f]]\downarrow$.

By the unwinding theorem with K[e],

$\exists n.\ K[e[F\_n/f]]\downarrow$.

By our assumption,

$K'[e'[F'\_n/f]]\downarrow$.

By the unwinding theorem with K'[e'],

$K'[e'[F'/f]]\downarrow$.

The other direction is analogous.

Q.E.D.

Theorem (Unwinding):

$e[F/f]\downarrow$ iff $\exists n.\ e[F\_n/f]\downarrow$.

Put another way:

$\exists n.\ e[F/f]\downarrow\downarrow e[F\_n/f]$.

Proof: HW for next Tuesday.

See Exercise 7.4.5 in Pitts' ATTPL chapter.

— Compatibility for fix

We're now ready to prove

Lemma (Compatibility for fix):

If $\Delta; \Gamma, f : \sigma \to \tau, x{:}\sigma \vdash e_1 \approx e_2 : \tau$,

then $\Delta; \Gamma \vdash \text{fix } f(x)e_1 \approx \text{fix } f(x).e_2 : \sigma \to \tau$.

Proof:

Let $\rho \in D[\Delta]$, $(\gamma_1,\gamma_2) \in G[\Gamma]\rho$, $\delta_1,\delta_2 \in \Delta \to \text{CType}$.

Set $F^1 := \text{fix } f(x).\delta_1\gamma_1 e_1$ and $F^2 := \text{fix } f(x).\delta_2\gamma_2 e_2$.

TS: $(\delta_1\gamma_1(\text{fix } f(x).e_1),\delta_2\gamma_2(\text{fix } f(x).e_2)) \in E[\sigma\to\tau]\rho$

$\iff$ $(\text{fix } f(x).\delta_1\gamma_1 e_1, \text{fix } f(x).\delta_2\gamma_2 e_2) \in E[\sigma\to\tau]\rho$

$\Leftarrow$ (Admissibility instantiated with e = f = e' and $F=F^1$ and $F'=F^2$ and $\tau=\sigma\to\tau$.)

$\forall n.(f[F^1\_n/f], f[F^2\_n/f]) = (F^1\_n,F^2\_n) \in E[\sigma\to\tau]\rho$

$\Leftarrow$ (Slight strengthening, at least to simplify the induction but possibly to make it go through.)

$\forall n.(F^1\_n,F^2\_n) \in V[\sigma\to\tau]\rho \subseteq E[\sigma\to\tau]\rho$.

By induction on n.

Case n = 0:

TS: $(\text{fix } f(x).f(x), \text{fix } f(x).f(x)) \in V[\sigma \to \tau]\rho$

$\Leftarrow$ $\forall (v_1, v_2) \in V[\sigma]\rho.$

$((\text{fix } f(x).f(x))\ v_1, (\text{fix } f(x).f(x))\ v_2) \in E[\tau]\rho.$

This follows immediately from the divergence lemma.

("Just by looking at them, ···" Heh.)

Case:

TS: $(F^1\_\{n+1\}, F^2\_\{n+2\}) \in V[\sigma \to \tau]\rho$

$\Leftarrow$ $\forall (v_1, v_2) \in V[\sigma]\rho.$

$((F^1\_\{n+1\})\ v_1, (F^2\_\{n+2\})\ v_2) \in E[\tau]\rho.$

But $F^1\_\{n+1\} \mapsto \delta_1\gamma_1 e_1[F^1\_n/f]$

and $F^2\_\{n+1\} \mapsto \delta_2\gamma_2 e_2[F^2\_n/f]$.

STS: $(\delta_1\gamma_1 e_1[F^1\_n/f], \delta_2\gamma_2 e_2[F^2\_n/f])) \in E[\tau]\rho.$

By IH,

WK: $(F^1\_n, F^2\_n) \in V[\sigma \to \tau]\rho.$

Define $\gamma'\_i := \gamma_1, x \mapsto v\_i, f \mapsto F\hat{}i\_n.$

WK: $(\gamma'_1, \gamma'_2)\ \mu o\ G(\Gamma, f{:}\sigma \to \tau, x{:}\sigma]\rho.$

Instantiating the premise we're done.

Q.E.D.

We may not have to strengthen our induction hypotheses in the previous proof. Suppose we had defined

$$G[\Gamma]\rho := \{ (\gamma_1, \gamma_2) \in (\text{dom}(\Gamma) \to \text{CVal})^2 \mid \forall (x{:}\tau) \in \Gamma.\ (\gamma_1 x,\ \gamma_2 x) \in E[\tau]\rho \}$$

Rather than using $V[\tau]\rho$. Then it might have worked out. This seems more intuitive.

— Compatibility for application and the Bind Lemma

Consider the following proof.

Lemma (Compatibility for application):

If $\Delta; \Gamma \vdash e_1 \equiv e_2 : \sigma \to \tau$

and $\Delta; \Gamma \vdash e'_1 \equiv e'_2 : \sigma$,

then $\Delta; \Gamma \vdash e_1\ e'_1 \equiv e_2\ e'_2 : \tau.$

Proof:

This will look a little painful.

Once we do it, we'll see a way to abstract the pain

so it's less painful in the future.

Let ··· intros ···
TS: $(\delta_1\gamma_1(e_1\ e'_1),\delta_2\gamma_2(e_2\ e'_2)) \in E[\tau]\rho$.
By IH,
1. $(\delta_1\gamma_1 e_1,\delta_2\gamma_2 e_2) \in E[\sigma\to\tau]\rho \wedge$
2. $(\delta_1\gamma_1 e'_1,\delta_2\gamma_2 e'_2) \in E[\sigma]\rho$.
Unfolding $E[\tau]\rho$, let $(K_1,K_2)\in K[\tau]\rho$.
TS: $K_1[\delta_1\gamma_1(e_1\ e'_1)] \downarrow\downarrow K_2[\delta_2\gamma_2(e_2\ e'_2)]$.
$\iff$ $K_1[\delta_1\gamma_1 e_1\ \delta_1\gamma_1 e'_1] \downarrow\downarrow K_2[\delta_2\gamma_2 e_2\ \delta_2\gamma_2 e'_2]$.
$\iff$ $K'_1[\delta_1\gamma_1 e_1] \downarrow\downarrow K'_2[\delta_2\gamma_2 e_2]$
where K'_i = K_i[• (δ_iγ_ie'_i)].

Instantiating 1,
STS: $(K'_1, K'_2) \in K[\sigma\to\tau]\rho$.
Let $(v_1,v_2) \in V[\sigma\to\tau]\rho$.
TS: $K'_1[v_1] \downarrow\downarrow K'_2[v_2]$
$\iff$ $K_1[v_1\ (\delta_1\gamma_1 e'_1)] \downarrow\downarrow K_2[v_2\ (\delta_2\gamma_2 e'_2)]$.

Set K''_i := K_i[v_i •].

STS: $K''_1[\delta_1\gamma_1 e'_1] \downarrow\downarrow K''_2[\delta_2\gamma_2 e'_2]$.
Instantiating 2,
STS: $(K''_1,K''_2) \in K[\sigma]\rho$.
Let $(v'_1,v'_2) \in V[\sigma]\rho$.
TS: $K''_1[v'_1] \downarrow\downarrow K''_2[v'_2]$
$\iff$ $K_1[v_1\ v'_1] \downarrow\downarrow K_2[v_2\ v'_2]$.

Since $(K_1,K_2) \in K[\tau]\rho$,
STS: $(v_1\ v'_1,\ v_2\ v'_2) \in E[\tau]\rho$
$\impliedby$ $(v_1,v_2) \in V[\sigma\to\tau]\rho \wedge$
$(v'_1,v'_2) \in V[\sigma]\rho$.
Q.E.D.

Claim (for next time): If you just go through your compatibility proofs naively, you'll have to jump through such hoops (reasoning about each subexpression, values, etc).

There's a simple lemma, called the bind lemma, that does this kind of reasoning once and for all.

By closing up the term relation so it's admissible, we no longer know as much about it as we once did. We don't obviously know from just looking at
$(e_1,e_2) \in E[\tau]\rho$

and the definitions, that e_1 and e_2 go to values related by the value relation. We end up doing this kind of back-and-forth reasoning.

Lemma (Bind lemma):
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 $(K_1[e_1],K_2[e_2]) \in E[\tau']\rho'$.

This is like a monadic bind/let: To prove $(K_1[e_1],K_2[e_2])$, you quantify over any values e_1 and e_2 could have produced.

Intuition: In proving the compatibility lemma for applications, we had to show (ignoring the δ,γ crap):

$(e_1\ e'_1,e_2\ e'_2) \in E[\tau]\rho$.

Observe this is of the form $(K'_1[e_1], K'_2[e_2]) \in E[\tau]\rho$.
So it reduces to showing $\forall(v_1,v_2) \in V[\sigma\rightarrow\tau]\rho.\ (K'_1[v_1],K'_1[v_2]) \in E[\tau]\rho$.
And similarly this reduces to showing $\forall(v'_1,v'_2) \in V[\sigma]\rho$.
$(K''_1[v'_1],K''_2[v'_2]) \in E[\tau]\rho$.
Then STS $(v_1\ v'_1,\ v_2\ v'_2) \in E[\tau]\rho$.

The bind lemma lets you reduce proving the compatibility lemma to proving it when you zap all the e's to v's.

— Next time.

We'll prove the bind lemma, go through some more compatibility lemmas, discuss how Pitts does things, and discuss how we apply the method to proving contextual equivalence.

We may try to sketch a variant of what we've done that's a bit closer to Pitts. Pitts uses a typed model (built up from syntactically typed terms). He proves completeness: Contextual equivalence implies logical equivalence. He gets some extensionality principles out of this approach. (Overall, it winds up being more complicated for the kinds of goals Derek is interested in.)

— Aside from Dave: Don't get stuck

We want the definition of cotermination to rule out stuck states in untyped models. Otherwise, we can't prove things we'd like to prove.

In class, Derek used the alternative

$e_1 \downarrow\downarrow' e_2 := e_1\downarrow \iff e2\downarrow$

instead of

$e_1 \downarrow\downarrow e_2 :\iff (e_1\downarrow \wedge e_2\downarrow) \vee (e_1\uparrow \wedge e_2\uparrow)$.

(We never defined $e\uparrow$. When asked, both Derek and Deepak read "$e_1\uparrow$" as ruling out stuck states.)

Either definition works in typed models: Progress and preservation ensure that the e_i don't get "stuck". In untyped models, the former says two e_i coteriminate if they are stuck.