Goal: Prove termination for CBV System F using unary logical relations.

We're not proving strong normalization (ie, every reduction sequence is finite). Girard proved SN for full β(η?)-reduction.

— Remarks

Next Tuesday, 10:00-noon in E1.4 024. No class next Thursday.

Forgot to mention last time: We'll use operational semantics as a unifying framework for all the things we review (eg, term models don't come up in Reynolds' paper on parametricity but they do here).
Pitts developed this approach.

— Call-by-value System F

Call-by-value System F.

Types	$\sigma, \tau ::= \alpha \mid \sigma \to \tau \mid \forall\alpha.\tau$
Terms	$e ::= x \mid \lambda x.e \mid e_1\ e_2 \mid \Lambda\alpha.e \mid e\ \sigma$
Values	$v ::= x \mid \lambda x.e \mid \Lambda\alpha.e$

No base types: Many typical base types ($\mathbb{N}$, $\bot$, 1, 2, etc) are definable.

Useful sets:

CTyp := { σ | ftv(σ) = ∅ } (closed types)
CTerm := { e | fv(e) = ∅ } (closed terms may have free type variables)
CVal := { v | fv(v) = ∅ } (closed values)

Static Semantics

Type Ctxts $\Delta ::= \cdot \mid \Delta,\alpha$
Term Ctxts$\Gamma ::= \cdot \mid \Gamma, x{:}\tau$

Judgement: $\Delta; \Gamma \vdash e : \tau$

We won't worry about well-formedness in our rules. We'll maintain the invariant that $\mathrm{ftv}(\Gamma,e,\tau) \subseteq \Delta$; $\mathrm{fv}(e) \subseteq \mathrm{dom}(\Gamma)$.

The typing rules are all standard.

$x{:}\tau \in \Gamma$
—
$\Delta; \Gamma \vdash x : \tau$

$\Delta; \Gamma, x{:}\sigma \vdash e : \tau$
—
$\Delta; \Gamma \vdash \lambda x.e : \sigma \rightarrow \tau$

$\Delta; \Gamma \vdash e_1 : \sigma \rightarrow \tau$
$\Delta; \Gamma \vdash e_2 : \sigma$
—
$\Delta; \Gamma \vdash e_1\ e_2 : \tau$

$\Delta, \alpha; \Gamma \vdash e : \tau$
—
$\Delta; \Gamma \vdash \Lambda\alpha.e : \forall\alpha.\tau$

$\Delta, \Gamma \vdash e : \forall\alpha.\tau$
$\mathrm{ftv}(\sigma) \subseteq \Delta$
—
$\Delta; \Gamma \vdash e\ \sigma : \tau[\sigma/\alpha]$

Dynamic semantics

Derek used one judgement e ↦ e'. Deepak and Viktor pointed out that presentations supporting a simpler metatheory exist. Derek argued they're equivalent. Such details matter most when you formalize your metatheory in a proof assistant. Derek suggested reducing non-determinism by with two judgements e ↦ e' and e ↦_r e'. I left the r's in place here, but ignore them when you read the sequel.

Eval Ctxt   K ::= • | K e | v K | K σ

Judgement:
e ↦ e'
e ↦_r e'

e ↦_r e'
—

$K[e] \mapsto K[e']$

—

$(\lambda x.e)v \mapsto_r e[v/x]$

—

$(\Lambda\alpha.e)\sigma \mapsto_r e[\sigma/\alpha]$

Definition:

We write $e \downarrow v$ if e evaluates to v in some number of steps ($e \mapsto^* v$).

— Direct proofs of termination fail

Goal:

If $\vdash e : \tau$,
then $\exists v.\ e \downarrow v$.

A direct proof by induction on the derivation won't work.

• We would obviously need to generalize to open terms.

• Even strengthened, induction on $D :: \Delta; \Gamma \vdash e : \tau$ fails (as follows) in the case for applications.

Case $e = e_1\ e_2$.

$\vdash e_1 : \sigma \to \tau$
$\vdash e_2 : \sigma$
—
$\vdash e_1\ e_2 : \tau$

By IH, $e_1 \downarrow v_1$ and $e_2 \downarrow v_2$.
(Imagine we know $v_1 = \lambda x.e'$.)
We have $e_1\ e_2 \mapsto^* (\lambda x.e')\ v_2 \mapsto e'[v_2/x]$.

Problem: We have no reason to believe $e'[v_2/x] \downarrow$.

Basic idea: Strengthen the IH in a more involved way, via (unary) logical relations (aka logical predicates).

— Unary logical relations

We begin by writing down the logical relation without the exciting bit that Girard added (candidates of reducibility).

Informally: We're going to define type-indexed families of sets $E[\tau]$ and $V[\tau]$ with (problem-specific) conditions baked in: We want E to pick out those terms that evaluate to values. Put another way, we're going to say how to interpret a type τ as a set of terms $E[\tau]$ that inhabit that type and as a set of values $V[\tau]$ that inhabit that type.

The rough idea (we'll have to improve it later):

$$\begin{array}{ll} \mathrm{CTerm} \supseteq & E[\tau] := \{ e \mid \exists v.\ e \downarrow v \wedge v \in V[\tau] \} \\ \mathrm{CVal} \supseteq & V[\sigma \to \tau] := \{ \lambda x.e \mid \forall v \in V[\sigma].\ e[v/x] \in E[\tau] \} \\ & V[\forall\alpha.\tau] := \{ \Lambda\alpha.e \mid \forall\sigma.\ e[\sigma/\alpha] \in E[\tau[\sigma/\alpha]] \} \end{array}$$

Some notes, before defining $V[\tau]$ at the other types:

• Informally, we've baked the step “$e'[v_2/x] \downarrow$” missing from our proof into the logical relation.

• This isn't a simple recursive definition of E and V: Induction on types is necessary for the thing to well-founded. In $V[\sigma \to \tau]$, we quantify over $V[\sigma]$; that's ok since σ is smaller than $\sigma \to \tau$. (We couldn't define V using “just” V in a negative position.) More concretely, if we erase the types to define E and V and we assert that such a construct exists, then we can prove that the untyped lambda calculus is strongly normalizing.

For the $\forall\alpha.\tau$ case, we can't use the obvious

$$V[\forall\alpha.\tau] := \{ \Lambda\alpha.e \mid \forall\sigma.\ e[\sigma/\alpha] \in E[\tau[\sigma/\alpha]] \}$$

since σ might be $\forall\alpha.\tau$, screwing up our induction on types. (This definition would work with a predicative language.)

This motivates Girard's “candidates of reducibility” trick: Abstract types can represent “arbitrary” sets of values (or terms, depending on the setup) where those sets are drawn from some class Cand of candidates. As with $V[-]$ and $E[-]$, we get to impose problem-specific restrictions on Cand.

For our purposes, we can get by with

$$\text{Cand} := \text{Sub}(\text{CVal}).$$

Informally, a candidate set is any set of closed values. Why do we expect that to work? First, note that in $\Lambda\alpha.e$, the subexpression e can't analyze α; for example, there's no typecase in System F. Second, we're modelling types as sets of values. Thus we should wind up with a candidate set being a set of values (with no conditions attached).

We'll use

$$V[\forall\alpha.\tau]\rho := \{ \Lambda\alpha.e \mid \forall\sigma.\ \forall S \in \text{Cand}.\ e[\sigma/\alpha] \in E[\tau](\rho, \alpha \mapsto S) \}$$

Some notes:

- Informally, σ is the syntactic point of α and we've added S, the semantic point of α.

- We're using $\rho : \text{Tyvar} \longrightarrow \text{Cand}$ to account for the free type variables in τ when writing $V[\tau]$. We denote the extension of ρ by $(\rho, \alpha \mapsto S)$.

- We're quatifying over all terms of form $\Lambda\alpha.e$: We don't require them to be well-typed or even to be closed wrt type variables. The whole point of such models is to separate ourselves from such syntactic considerations.

- The thing is trivially well-founded: τ is clearly smaller than $\forall\alpha.\tau$.

- We will use ρ in the interpretation of type variables. The other cases simply pass ρ along.

The full definition:

$$\text{CTerm} \supseteq \quad E[\tau]\rho := \{ e \mid \exists v.\ e \downarrow v \wedge v \in V[\tau]\rho \}$$

$$\begin{aligned}\text{CVal} \supseteq V[\alpha]\rho &:= \rho(\alpha)\\ V[\sigma\longrightarrow\tau]\rho &:= \{ \lambda x.e \mid \forall v \in V[\sigma]\rho.\ e[v/x] \in E[\tau]\rho \}\\ V[\forall\alpha.\tau]\rho &:= \{ \Lambda\alpha.e \mid \forall\sigma.\ \forall S \in \text{Cand}.\ e[\sigma/\alpha] \in E[\tau](\rho, \alpha \mapsto S) \}\end{aligned}$$

— Fundamental Theorem

We'll want to state a theorem about open terms along the lines:

If $\Delta; \Gamma \vdash e : \tau$,
then $e \in E[\tau]\cdots$.

but our logical relation is defined over closed terms. We'll use a standard trick: Quantify over all "closing substitutions" of the context. (Think of e as a function from its context to its result type.)

Define

$D[\Delta] := \{ \rho \in \text{Tyvar} \rightharpoonup \text{Cand} \mid \Delta \subseteq \text{dom}(\rho) \}$

$G[\Gamma]\rho := \{ \gamma \in \text{Var} \rightharpoonup \text{CVal} \mid \forall(x{:}\tau) \in \Gamma.\ \gamma(x) \in V[\tau]\rho \}$

Informally, $D[\Delta]$ picks out those substitutions ρ supporting Δ (all candidates are interesting) and $G[\Gamma]\rho$ picks out those substitutions γ supporting Γ with "interesting" values.

Aside: Read "$V[\tau]\rho$" as "the value relation at τ interpreted by ρ".

Theorem (fundamental theorem of the logical relation):

If $\Delta; \Gamma \vdash e : \tau$,
then $\forall\rho \in D[\Delta].\ \forall\gamma \in G[\Gamma]\rho.\ \gamma e \in E[\tau]\rho$.

Informally, the theorem says that the model respects the syntax of the language. Once we generalize to relational parametricity, the fundamental theorem is sometimes called the abstraction theorem (following Reynolds).

Corollary:
If $\vdash e : \tau$,
then $e \downarrow v$.

Proof: By the fundamental theorem with ρ, γ the identity substitutions. We have $\gamma e = e \in E[\tau]\rho \Rightarrow e \downarrow v$. Q.E.D.

As Derek worked through the proof, he "discovered" some necessary lemmas as well as a bug. That's how these things go. I leave things as Derek presented them. (Exercise: Fix the buggy proof.)

Proof of the fundamental theorem:
By induction on the derivation $D :: \Delta; \Gamma \vdash e : \tau$.

Case

$$D = \frac{x{:}\tau \in \Gamma}{\Delta; \Gamma \vdash x : \tau}$$

Let $\rho \in D[\Delta]$ and $\gamma \in G[\Gamma]\rho$ be given.

(Aside: From now on, we implicitly introduce parameters when proving $\forall x.\varphi$ and implicitly close with "since x was arbitrary, $\forall x.\varphi$".)

TS: $\gamma x \in E[\tau]\rho$.
(Aside: TS = to show).

By definition of $G[\Gamma]\rho$, we know $\gamma x \in V[\tau]\rho \subseteq E[\tau]\rho$.

(Aside: We implicitly used the "coincidence lemma" $V[\tau]\rho \subseteq E[\tau]\rho$. It's too trivial to state in this setting.)

(Aside: There is an important lemma:

> If $\mathrm{ftv}(\tau) \subseteq \Delta$
> and $\rho \in D[\Delta]$,
> then $V[\tau]\rho \in \mathrm{Cand}$.

It's trivial in this setting. In Girard's proof, Cand has various closure properties that make this lemma nontrivial.)

Case

$$D = \frac{\Delta; \Gamma, x{:}\sigma \vdash e : \tau}{\Delta; \Gamma \vdash \lambda x.e : \sigma \rightarrow \tau}$$

TS: $\gamma(\lambda x.e) = \lambda x.(\gamma e) \in E[\sigma \rightarrow \tau]\rho$.

(Aside: We implicitly assume, without loss of generality, that $x \# \mathrm{dom}(\gamma)$.)

(Aside: We implicitly use substitiution lemmas.)

By coincidence, it suffices to show

$$\lambda x.(\gamma e) \in V[\sigma \rightarrow \tau]\rho$$

$\Longleftrightarrow$ (Definition of $V[\sigma \rightarrow \tau]$.)

$$\forall v \in V[\sigma]\rho.\ (\gamma e)[v/x] \in E[\tau]\rho.$$

Set $\gamma' := (\gamma, x \mapsto v)$. Then $\gamma' \in G[\Gamma, x{:}\sigma]\rho$.
By IH, $(\gamma e)[v/x] = \gamma' e \in E[\tau]\rho$.

Case

$$D = \dfrac{\Delta; \Gamma \vdash e_1 : \sigma \to \tau \quad \Delta; \Gamma \vdash e_2 : \sigma}{\Delta; \Gamma \vdash e_1\ e_2 : \tau}$$

By IH, $\gamma e_1 \in E[\sigma \to \tau]\rho$ and $\gamma e_2 \in E[\sigma]\rho$.
TS: $\gamma(e_1\ e_2) = (\gamma e_1)(\gamma e_2) \in E[\tau]\rho$.

Then we may choose v_1 and v_2 satisfying

$(\gamma e_1) \downarrow v_1 \in V[\sigma{\to}\tau]\rho$
$(\gamma e_2) \downarrow v_2 \in V[\sigma]\rho$

By the definition of $V[\sigma{\to}\tau]$, we know

$v_1 = \lambda x.e$

such that $e[v_2/x] \in E[\tau]\rho$.
The case is done once we apply the following lemma.

(Another trivial here lemma that isn't trivial in Girard's setting.

Lemma (Closure under expansion):

If $e \in E[\tau]\rho$
and $e' \mapsto_* e$,
then $e' \in E[\tau]\rho$.

Proof: Trivial.)

Case

$$D = \dfrac{\Delta, \alpha; \Gamma \vdash e : \tau}{\Delta; \Gamma \vdash \Lambda\alpha.e : \forall\alpha.\tau}$$

TS: $\gamma(\Lambda\alpha.e) = \Lambda\alpha.(\gamma e) \in E[\forall\alpha.\tau]\rho$
$\Leftarrow \gamma(\Lambda\alpha.e) \in V[\forall\alpha.\tau]\rho$.

Let σ, $S \in$ Cand be given.
TS: $(\gamma e)[\sigma/\alpha] \in E[\tau](\rho, \alpha \mapsto S)$.

Set $\rho' := (\rho,\alpha \mapsto S) \in D[\Delta,\alpha]$.
We have $\Gamma \in G[\Gamma]\rho \iff$ (Because $ftv(\Gamma) \subseteq \Delta$.) $G[\Gamma]\rho'$.
(Ie, by assumption Γ does not refer to α.)
By the IH, we have $\gamma e \in E[\tau]\rho'$.

To fix the proof, note that types shouldn't matter.
The difference between γe and $(\gamma e)[\sigma/\alpha]$ shouldn't matter.
We can probably fix this proof by generalizing it:

If $\Delta; \Gamma \vdash e : \tau$,
then $\forall\rho \in D[\Delta].\ \forall\gamma \in G[\Gamma]\rho.\ \forall\delta : \Delta \rightharpoonup \text{Type}.\ \delta(\gamma e) \in E[\tau]\rho$.

In this case, we also extend δ:
$\delta' := (\delta, \alpha \mapsto \sigma)$.
The IH gives us
$\delta'(\gamma e) = (\delta\gamma e)[\sigma/\alpha] \in E[\tau]\rho'$.

Exercise: Go back and add the δ's in to this proof. The only hard case should be the following.

Case

$$D = \frac{\Delta, \Gamma \vdash e : \forall\alpha.\tau \qquad ftv(\sigma) \subseteq \Delta}{\Delta; \Gamma \vdash e\ \sigma : \tau[\sigma/\alpha]}$$

Suppose we have $\rho \in D[\Delta]$, $\gamma \in G[\Gamma]\rho$, and $\delta : \Delta \rightharpoonup \text{Type}$.

By IH, $\delta\gamma e \in E[\forall\alpha.\tau]\rho$.

Thus, $\delta\gamma e \downarrow \Lambda\alpha.e' \in V[\forall\alpha.\tau]\rho$.

TS: $(\delta\gamma e)(\delta\sigma) \in E[\tau[\sigma/\alpha]]\rho$.

$(\delta\gamma e)(\delta\sigma) \mapsto_* (\Lambda\alpha.e')(\delta\sigma) \mapsto e'[\delta\sigma/\alpha]$.

By closure under expansion, it suffices to show

$e'[\delta\sigma/\alpha] \in E[\tau[\sigma/\alpha]]\rho$.

By the definition of $V[\forall\alpha.\tau]\rho$., we want to pick some $S \in \text{Cand}$ such that $e'[\delta\sigma/\alpha] \in E[\tau](\rho,\alpha\mapsto S)$ and then (cliff-hanger) relate $E[\tau](\rho,\alpha\mapsto S)$ and $E[\tau[\sigma/\alpha]]\rho$.

[⋯more next time⋯Idea: Pick $S = V[\sigma]\rho$ and show the two sets equal by induction on types.]

Informally, the logical relation lets us pick any S we want. For the proof to go through, we need to be able to pick S to be the interpretation of the syntactic type. Ie, that the interpretation of $V[\sigma]\rho$ is in Cand. In other settings, its not trivial.

For next time: Understand this proof. It'll have to feel like boilerplate in the future: Every model adds “interesting” proof obligations.