Contextual equivalence $\Gamma \vdash e_1 \approx^{ctx} e_2 : \tau = \forall C : (\Gamma \vdash \tau) \rightsquigarrow (\cdot \vdash bool)$
$\Rightarrow C[e_1] \Downarrow v \iff C[e_2] \Downarrow v$

$Atom[\tau_1, \tau_2] = \{(e_1, e_2) \mid \cdot \vdash e_1 : \tau_1 \wedge \cdot \vdash e_2 : \tau_2\}$ $\quad Atom[\tau]\rho \stackrel{def}{=} Atom[\rho_1(\tau), \rho_2(\tau)]$
$Rel[\tau_1, \tau_2] = \{R = \mathcal{P}(Atom^{val}[\tau_1, \tau_2])\}$

$\mathcal{V}[\![bool]\!]\rho = \{(v_1, v_2) \in Atom[bool]\rho \mid v_1 = v_2 = true \ \vee \ v_1 = v_2 = false\}$
$\mathcal{V}[\![\tau_1 \to \tau_2]\!]\rho = \{(\lambda x{:}\rho_1(\tau_1).\, e_1, \lambda x{:}\rho_2(\tau_2).\, e_2) \in Atom[\tau_1 \to \tau_2]\rho$
$\quad \mid \forall (v_1, v_2) \in \mathcal{V}[\![\tau_1]\!]\rho.\ (e_1[^{v_1}/_x], e_2[^{v_2}/_x]) \in \mathcal{E}[\![\tau_2]\!]\rho\}$
$\mathcal{V}[\![\forall \alpha.\tau]\!]\rho = \{(\Lambda\alpha.e_1, \Lambda\alpha.e_2) \mid \forall \tau_1, \tau_2.\ \forall R \in Rel[\tau_1, \tau_2].$
$\quad (e_1[^{\tau_1}/_\alpha], e_2[^{\tau_2}/_\alpha]) \in \mathcal{E}[\![\tau]\!]\rho[\alpha \mapsto (\tau_1, \tau_2, R)]\}$
$\mathcal{V}[\![\alpha]\!]\rho = \{(v_1, v_2) \in Atom[\alpha]\rho \mid (v_1, v_2) \in \rho_R(\alpha)\}$
$\mathcal{V}[\![\exists \alpha.\tau]\!]\rho = \{(pack(\tau_1, v_1), pack(\tau_2, v_2) \in Atom[\exists \alpha.\tau]\rho \mid \exists R \in Rel[\tau_1, \tau_2].$
$\quad (v_1, v_2) \in \mathcal{V}[\![\tau]\!]\rho[\alpha \mapsto (\tau_1, \tau_2, R)]$

$\mathcal{E}[\![\tau]\!]\rho = \{(e_1, e_2) \in Atom[\tau] \mid \exists v_1, v_2.\ e_1 \mapsto^* v_1 \wedge e_2 \mapsto^* v_2 \wedge (v_1, v_2) \in \mathcal{V}[\![\tau]\!]\rho\}$

$\mathcal{D}[\![\cdot]\!] = \{\emptyset\}$
$\mathcal{D}[\![\Delta, \alpha]\!] = \{\rho[\alpha \mapsto (\tau_1, \tau_2, R)] \mid \rho \in \mathcal{D}[\![\Delta]\!] \wedge R \in Rel[\tau_1, \tau_2]\}$

$\mathcal{G}[\![\cdot]\!]\rho = \{\emptyset, \emptyset\}$
$\mathcal{G}[\![\Gamma, x{:}\tau]\!]\rho = \{\gamma_1[x_1 \mapsto v_1], \gamma_2[x_2 \mapsto v_2] \mid (\gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\rho$
$\quad \wedge (v_1, v_2) \in \mathcal{V}[\![\tau]\!]\rho\}$

$\Delta; \Gamma \vdash e_1 \approx e_2 : \tau \stackrel{def}{=} \forall \rho \in \mathcal{D}[\![\Delta]\!].\ \forall (\gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\rho.$
$\quad (\rho_1\gamma_1(e_1), \rho_2\gamma_2(e_2)) \in \mathcal{E}[\![\tau]\!]\rho$

Up on the board is what we had from Saturday. Where were we? We were trying to program equivalence with logical relations in System F, which is up on the board. I didn't say this, but System F is technically STLC with polymorphism; we just added existentials. Our relation was a binary relation rather than a unary relation that we were using before; we setup interpretations of types saying how to values are related at some interpretation. And as an aside, I'm using a notation where v1, v2 \in V[t], but I could have written this differently v1 ~ v2 : t iff ... This is just a notational thing, and you'll see both styles. The pair-ness is not essential.

$(v_1, v_2) \in \mathcal{V}[\![\tau]\!]$ $\quad$ $v_1 \sim v_2 : \tau$ iff ...

You, the person who is using the logical relation, gets to pick the relation. But if I'm trying to show something is related, I have to work under an arbitrary relation when I actually use the relation. So we used rho to help us keep track of these relations. Existentials are the dual of the universals.

What have we not done? We have only talked about closed terms, but we also need to consider open terms. Our sense of contextual equivalence is actually not quite right, since we need to add type variables to the context.

Contextual equivalence $\Delta; \Gamma \vdash e_1 \approx^{ctx} e_2 : \tau = \forall C : (\Delta; \Gamma \vdash \tau) \rightsquigarrow (\cdot; \cdot \vdash bool)$
$\Rightarrow C[e_1] \Downarrow v \iff C[e_2] \Downarrow v$

Now, we want to define a logical relation for two open terms.

$\Delta; \Gamma \vdash e_1 \approx e_2 : \tau \stackrel{def}{=} ?$

We'll do the same thing as we did previously, which is "give me a bunch of related substitutions to close me off" and then we relate the closed version. We have to do this for both term variables and type variables. Rho will let us do this for types. So let's define relations for delta and gamma.

$D[\![\cdot]\!] = \{\emptyset\}$
$D[\![\Delta, \alpha]\!] = \{\rho[\alpha \mapsto (\tau_1, \tau_2, R)] \mid \rho \in D[\![\Delta]\!] \wedge R \in Rel[\tau_1, \tau_2]\}$

Notice that we were always careful to make sure R was a Rel[t1, t2]. That is all I am checking over here.

Q: Can't I always pick R to be the complete relation, in which case the existential trivially holds?
A: Yes, but it won't be useful. It will be important it won't be all values of t1 and t2, because what the statement says is that there exists some special relation, a cut down version, it could even be singleton. What you define in your relation has to be precisely what you want to be related.

Note that we don't have very many restrictions on Rel at the moment. But when we add recursive types, Rel will now need some restrictions reminiscent of the monotonicity restrictions we saw previously.

Gamma will need two substitutions!

$$\mathcal{G}[\![\cdot]\!]\rho = \{\emptyset, \emptyset\}$$

$$\mathcal{G}[\![\Gamma, x:\tau]\!]\rho = \{\gamma_1[x_1 \mapsto v_1], \gamma_2[x_2 \mapsto v_2] \mid (\gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\rho \wedge (v_1, v_2) \in \mathcal{V}[\![\tau]\!]\rho\}$$

$$\Delta;\Gamma \vdash e_1 \approx e_2 : \tau \stackrel{\text{def}}{=} \forall \rho \in \mathcal{D}[\![\Delta]\!].\ \forall (\gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\rho.\ (\rho_1\gamma_1(e_1), \rho_2\gamma_2(e_2)) \in \mathcal{E}[\![\tau]\!]\rho$$

Don't forget that gamma needs rho, so it can pass it on appropriately. We now have a logical relation for open terms. So we should now be able to do our proof of equivalence using our logical relation. Once again, we have a two step process: we need to show the relation is sound and complete for showing contextual equivalence. But before that, we should check it upholds the FUNDAMENTAL PROPERTY.

Theorem (Fundamental Property) Also known as parametricity

$$\Delta;\Gamma \vdash e:\tau \Rightarrow \Delta;\Gamma \vdash e \approx e : \tau$$

Recall, last time, we did something a little different: $\Gamma \vdash e:\tau \Rightarrow \Gamma \vDash e:\tau$
These are almost the same thing (one is closed, the other is open), but in the new case, our relation is binary.

Q: Aren't you missing indices?
A: No! Although this looks very benign (obviously e is equivalent to e), there is some serious stuff hiding under the surface of this statement. To prove this, you will need a number of compatibility lemmas. You'll have one per type inference rule in the language, and they will do "the work you want".

Think of the fundamental property as "reflexivity".

Compatibility Lemmas

$$\frac{\Gamma(x) = \tau}{\Delta; \Gamma \vdash x \approx x : \tau}$$

$$\frac{\Delta; \Gamma, x:\tau \vdash e_1 \approx e_2 : \tau'}{\Delta; \Gamma \vdash \lambda x{:}\tau.\, e_1 \approx \lambda x{:}\tau.\, e_2 : \tau \to \tau'}$$

(Ex)
$$\frac{\Delta; \Gamma \vdash e_1 \approx e_2 : \forall \alpha.\tau \qquad \Delta \vdash \tau'}{\Delta; \Gamma \vdash e_1[\tau'] \approx e_2[\tau'] : \tau[\tau'/\alpha]}$$

$$\frac{\Delta, \alpha : \Gamma \vdash e_1 \approx e_2 : \tau}{\Delta; \Gamma \vdash \Lambda\alpha.\, e_1 \approx \Lambda\alpha.\, e_2 : \forall \alpha.\tau}$$

⋮

(Pack and unpack are also very interesting)

Once you have this, you have the fundamental property easily, by induction on the derivation. Each case is discharged by the corresponding compatibility lemma (which is actually more general, because it's different on both sides.) Let's prove the compatibility lemma for type abstraction.

Compat If $\Delta, \alpha : \Gamma \vdash e_1 \approx e_2 : \tau$ then $\Delta; \Gamma \vdash \Lambda\alpha.\, e_1 \approx \Lambda\alpha.\, e_2 : \forall \alpha.\tau$

Proof Suppose $\rho \in \mathcal{D}[\![\Delta]\!]$, $(\gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\rho$, show

$$(\rho_1\gamma(\Lambda\alpha.\, e_1), \rho_2\gamma(\Lambda\alpha.\, e_2)) \in \mathcal{E}[\![\forall x.\tau]\!]\rho$$

$$\equiv (\Lambda\alpha.\, \rho_1\gamma(e_1), \Lambda\alpha.\, \rho_2\gamma(e_2))$$

pushing the substitution in since alpha is not in rho or gamma.

Conveniently, our expressions are already values.

It suffices to show: $(\Lambda\alpha.\, \rho_1\gamma(e_1), \Lambda\alpha.\, \rho_2\gamma(e_2)) \in \mathcal{V}[\![\forall\alpha,\tau]\!]\rho$

Consider arbitrary τ_1, τ_2 and $R \in \mathrm{Rel}[\tau_1, \tau_2]$.

Show: $(\rho_1\gamma_1(e_1)[\tau_1/\alpha], \rho_2\gamma_2(e_2)[\tau_2/\alpha]) \in \mathcal{E}[\![\tau]\!]\rho[\alpha \mapsto (\tau_1, \tau_2, R)]$

Now we can use our hypothesis.

We need $\rho' \in \mathcal{D}[\![\Delta, \alpha]\!]$ and $(\gamma_1', \gamma_2') \in \mathcal{G}[\![\Gamma]\!]\rho'$

We have $\rho \in \mathcal{D}[\![\Delta]\!]$, so $\rho' = \rho[\alpha \mapsto (\tau_1, \tau_2, R)]$ (checking $\mathcal{D}[\![\Delta, \alpha]\!]$ rule)

We have $(\gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\rho$, so weakening gives us this under ρ'

$\therefore (\rho_1'\gamma_1(e_1), \rho_2'\gamma_2(e_2)) \in \mathcal{E}[\![\tau]\!]\rho[\alpha \mapsto (\tau_1, \tau_2, R)]$

$\rho_1\gamma_1(e_1)[\tau_1/\alpha]$

Goal:

The most interesting compatibility lemma is type application, which will need a compositionality lemma (also known as a semantic substitution lemma).

Lemma (Compositionality)
If $\Delta \vdash \tau'$ and $\Delta, \alpha \vdash \tau$ and $\rho \in \mathcal{D}[\![\Delta]\!]$ and $R = \mathcal{V}[\![\tau']\!]\rho$,
then $\mathcal{V}[\![\tau[\tau'/\alpha]]\!]\rho = \mathcal{V}[\![\tau]\!]\rho[\alpha \mapsto (\rho_1\tau', \rho_2\tau', R)]$

That is, if you look at the interpretation of some arbitrary tau, where tau' does not have the variable in it, if you substitute t' for alpha syntactically, you should get exactly the same relation, as if you just took the interpretation of tau with the free binding (and your interpretation was rho1 of t' and rho2 of t' and R having v of t'. Semantically substituting t' for alpha is the same thing syntactically substituting t' for alpha. It lets you shift your view from semantics to syntax, which comes in handy. syntax $\Longleftrightarrow$ semantics

Now, the next thing you need to show is soundness and completeness w.r.t. contextual equivalence:

Theorem (Soundness w.r.t. contextual equivalence)
If $\Delta; \Gamma \vdash e_1 \approx e_2 : \tau$ then $\Delta; \Gamma \vdash e_1 \approx^{ctx} e_2 : \tau$

Theorem (Completeness w.r.t. contextual equivalence)
If $\Delta; \Gamma \vdash e_1 \approx^{ctx} e_2 : \tau$ then $\Delta; \Gamma \vdash e_1 \approx e_2 : \tau$

TT-closure
or ⊥⊥-closure

Completeness sounds like if you had been able to pull it off, you'd have a method that would work for everything in the world. This is suspicious. So if you see that someone has claimed they have completeness, look to see if they've used biorthorgonality (or top-top closure, or bottom-bottom closure). This uses a different definition of relatedness, where if I put related expressions in related contexts, I get related values. What does it mean for evaluation contexts to be related? They'd be related where if you give them logically related expressions, you get logically related expressions. This is kind of circular. This is the standard trick for making the logical relation complete, but if you then go and try to use a top-top closed logical relation to do examples, e.g. weird and interesting examples (esp. mutable references), you may not be able to show them equivalent. The top-top trick just baked contextual equivalence into the logical relation. You're pulling the definition of contextual equivalence into your definition to get the completeness theorem work. I'm not deriding this, but it is an important distinction to make. You may need to setup a logical relation which is sound and complete, for technical reasons, but you won't necessarily need to lean on it for an example. (Though, these days, with the state of the art, things are quite good).

Q: Why is it called completeness?
A: Well, that's what it is! (The definitions are just funny). It's traditional.

Funny story: one of the reviewers said: your logical relation isn't even complete with regards to contextual equivalence. And back then, it was not well understood that you could have top-top to get completeness, while not being able to take all examples and prove them logically equivalent.

If you want to understand exactly how this proof works, soundness mostly follows from the fundamenta theorem. As for completeness, I spell this out in a summer school lecture two years ago, OPLSS 2011, lecture three. All of this treatment, and for step-indexed relations, it can be found in a tech report ESOP 2006 "Syntactic step-indexed logical relations for polymorphic lambda calculus and lambda calculus." This tech report has every proof step spelled out in every detail. It is very detailed, you'll have to squint at some notational differences but it is really helpful. Completeness for this is very difficult, so this TR will be useful.

Free theorem: If $\cdot \vdash e : \forall\alpha.\alpha\to\alpha$ and $\cdot \vdash v : \tau$ then $e[\tau]\,v \mapsto^* v$

Proof By fundamental property, $\cdot \vdash e \approx e : \forall\alpha.\alpha\to\alpha$

Therefore, $(e,e) \in \mathcal{E}[\![\forall\alpha.\alpha\to\alpha]\!]\emptyset$

$\exists f.\ e \mapsto^* f,\ (f,f) \in \mathcal{V}[\![\forall\alpha.\alpha\to\alpha]\!]\emptyset$

We're going to lean on determinacy, (we know $f = \Lambda\alpha.e_f$)

How are going to use this fact? We get to provide τ_1, τ_2 and R

this is the only interesting thing in the proof.

$\tau_1 := \tau \qquad \tau_2 := \tau \qquad R := \{(v,v)\}$ (singleton!)

$\therefore (e_f[\tau/\alpha], e_f[\tau/\alpha]) \in \mathcal{E}[\![\alpha]\!]\emptyset[\alpha \mapsto (\tau,\tau,R)]$ (unwind some more)

$\exists g.\ e_f[\tau/\alpha] \mapsto^* g,\ (g,g) \in \mathcal{V}[\![\alpha\to\alpha]\!]\emptyset[\alpha\mapsto(\tau,\tau,R)]$

Now we get to provide $(v_1,v_2) \in \mathcal{V}[\![\alpha]\!]\emptyset[\alpha\mapsto(\tau,\tau,R)]$ (we know $g = \lambda x{:}\tau.e_g$)

$\equiv R$ by definition!

So: $v_1 := v \qquad v_2 := v$

$\therefore (e_g[v/x], e_g[v/x]) \in \mathcal{E}[\![\alpha]\!]\emptyset[\alpha\mapsto(\tau,\tau,R)]$

$\exists v'.\ e_g[v/x] \mapsto^* v' \wedge (v',v') \in \mathcal{V}[\![\alpha]\!]\emptyset[\alpha\mapsto(\tau,\tau,R)]$

$\therefore v' = v$ as it is the only thing in the relation.

What we have done is taken e[t] v and run it down to a value, and found out that the result was v', which must be v!

$e[\tau]\,v \mapsto^* f[\tau]\,v \mapsto^* g\,v \mapsto^* v' = v$

Q: It looks like some sort of abstract interpretation.
A: It is true that in logical relations, you tend to run things for a while, and then find out something about the result. But it's not abstract interpretation.

Exercise: show that our pack example is equivalent. Here, you'll need to pick an R appropriately, and it will be the singleton (4, true)

Q: Can you do a unary relation for polymorphic calculus?
A: Yes, it would look quite simple, just with one side omitted.

Q: What happens if you have full beta reduction?
A: I'm not sure.

Modifications for recursive types (w/o normalization)

Don't get confused for the function case and the expression case; function case talks about doing something later.

Contextual approximation $\Gamma \vdash e_1 \lesssim^{ctx} e_2 : \tau = \forall C : (\Gamma \vdash \tau) \rightsquigarrow (\cdot \vdash bool)$
$\Rightarrow C[e_1] \Downarrow v \Rightarrow C[e_2] \Downarrow v$

$Atom[\tau_1, \tau_2] = \{(e_1, e_2) \mid \cdot \vdash e_1 : \tau_1 \wedge \cdot \vdash e_2 : \tau_2\}$ $\quad Atom[\tau]\rho \stackrel{def}{=} Atom[e_1(\tau), e_2(\tau)]$

$Rel[\tau_1, \tau_2] = \{R = \mathbb{N} \to \mathcal{P}(Atom^{val}[\tau_1, \tau_2]) \mid \forall j < k. \forall v_1, v_2.\ (v_1, v_2) \in R(k) \to (v_1, v_2) \in R(j)\}$

If things belong to R at k, then it has to belong in R at j < k (monotonicity)

$\mathcal{V}_k[\![bool]\!]\rho = \{(v_1, v_2) \in Atom[bool]\rho \mid v_1 = v_2 = true \ \vee \ v_1 = v_2 = false\}$

$\mathcal{V}_k[\![\tau_1 \to \tau_2]\!]\rho = \{(\lambda x{:}\rho_1(\tau_1).\, e_1, \lambda x{:}\rho_2(\tau_2).\, e_2) \in Atom[\tau_1 \to \tau_2]\rho$
$\mid \forall j \le k.\ \forall (v_1, v_2) \in \mathcal{V}_j[\![\tau_1]\!]\rho.\ (e_1[v_1/x], e_2[v_2/x]) \in \mathcal{E}_j[\![\tau_2]\!]\rho\}$

$\mathcal{V}[\![\forall \alpha.\tau]\!]\rho = \{(\Lambda\alpha.e_1, \Lambda\alpha.e_2) \mid \forall j \le k.\ \forall \tau_1, \tau_2.\ \forall R \in Rel[\tau_1, \tau_2](j).$
$(e_1[\tau_1/\alpha], e_2[\tau_2/\alpha]) \in \mathcal{E}_j[\![\tau]\!]\rho[\alpha \mapsto (\tau_1, \tau_2, R)]\}$

$\mathcal{V}_k[\![\alpha]\!]\rho = \{(v_1, v_2) \in Atom[\alpha]\rho \mid (v_1, v_2) \in \rho_R(\alpha)(k)\}$ It/Ite doesn't seem to matter

$\mathcal{V}_k[\![\exists \alpha.\tau]\!]\rho = \{(pack(\tau_1, v_1), pack(\tau_2, v_2)) \in Atom[\exists \alpha.\tau]\rho \mid j < k \ \exists R \in Rel[\tau_1, \tau_2].$
$(v_1, v_2) \in \mathcal{V}_j[\![\tau]\!]\rho[\alpha \mapsto (\tau_1, \tau_2, R)]\}$

$\mathcal{V}_k[\![\mu\alpha.\tau]\!]\rho = \{(fold\ v_1, fold\ v_2) \mid \forall j < k.\ (v_1, v_2) \in \mathcal{V}_j[\![\tau[\mu\alpha.\tau/\alpha]]\!]\}$

** $\mathcal{E}_k[\![\tau]\!]\rho = \{(e_1, e_2) \in Atom[\tau] \mid \forall j < k.$

We can't put j here, because that would imply that they would have to take the same number of steps.

$\forall v_1.\ e_1 \overset{j}{\mapsto} v_1 \Rightarrow \exists v_2.\ e_2 \overset{*}{\mapsto} v_2 \wedge (v_1, v_2) \in \mathcal{V}_{k-j}[\![\tau]\!]\rho\}$

yes! an arbitrary number of steps. the previous intuition of step-indices breaks down here. It really is just an inductive principle.

ezyang: I think D does not need an index.

$D[\![\cdot]\!] = \{\emptyset\}$

$D[\![\Delta, \alpha]\!] = \{\rho[\alpha \mapsto (\tau_1, \tau_2, R)] \mid \rho \in D[\![\Delta]\!] \wedge R \in Rel[\tau_1, \tau_2]\}$

$\mathcal{G}_k[\![\cdot]\!]\rho = \{\emptyset, \emptyset\}$

$\mathcal{G}_k[\![\Gamma, x{:}\tau]\!]\rho = \{\gamma_1[x_1 \mapsto v_1], \gamma_2[x_2 \mapsto v_2] \mid (\gamma_1, \gamma_2) \in \mathcal{G}_k[\![\Gamma]\!]\rho$
$\wedge (v_1, v_2) \in \mathcal{V}_k[\![\tau]\!]\rho\}$

$\Delta; \Gamma \vdash e_1 \lesssim e_2 : \tau \stackrel{def}{=} \forall k \ge 0.\ \forall \rho \in D[\![\Delta]\!].\ \forall (\gamma_1, \gamma_2) \in \mathcal{G}_k[\![\Gamma]\!]\rho.$
$(\rho_1\gamma_1(e_1), \rho_2\gamma_2(e_2)) \in \mathcal{E}_k[\![\tau]\!]\rho$

This weird asymmetry in the E case causes interesting problems for step-indexed logical relations. In particular, you can't prove transitivity anymore. If you have e1 <= e2 : t, and e2 <= e3 : t, if you want to show e1 <= e3 : t, the problem is spelled out in ESOP, but the problem is what substitutions are you picking? You have something bounding the thing on the left, but not the thing on the right, that can take an arbitrary number of steps. So what do you do on the rhs e2? Read the paper for the details. There are a lot of tricks to try to bound the other step, but you will always end up demanding both sides have to reduce with the same number of steps. So now everyone agrees this is the correct definition.

Wrapup remarks and references. Last time, in the beginning we started with step indices (third and fourth lecture). The other thing I did not cover is how to do step-indexed relations with mutable references. This is very interesting for practical languages, so if you want to understand this better, 2011 and 2012 will work; check the fourth lecture/fifth lecture respectively. Tomorrow, when I survey logical relations, I'll talk about more pointers to the literature. This afternoon, I'm going to talk about compiler correctness. This will be a "state of the compiler correctness", giving a tiny sense of how to take a trivial translation and setup the logical relation. But what I really want to say is, what is it we really want to prove?