We have four more lectures left but a lot of topics to cover.
We'll move on, perhaps continuing the example from last time
via HW.

Today we'll talk about higher-order state. It's another basic direction
from which to generalize the work of Pitts and Stark. We'll talk about
higher-order state and discuss some examples.

For the next three lectures, we'll see a high-level overview
of a couple papers on more advanced topics. We'll discuss the ICFP'10
paper on Thursday, the POPL'13 concurrency paper after that, and
recent work on dependent types.

— Higher-order state

The easiest way to undertand how step-indexing helps with higher-order
state is to work through what happens when you try to make the obvious
changes to the model.

Syntax:

$\tau ::= \cdots \mid \text{ref}\ \tau$

We dropped ref (integer references)
in favor of ML-style references.

Statics:

$\Delta; \Gamma \vdash e : \tau$
—
$\Delta; \Gamma \vdash \text{ref}\ e : \text{ref}\ \tau$

$\Delta; \Gamma \vdash e : \text{ref}\ \tau$
—
$\Delta; \Gamma \vdash\ !e : \tau$

$\Delta; \Gamma \vdash e_1 : \text{ref}\ \tau$
$\Delta; \Gamma \vdash e_2 : \tau$
—
$\Delta; \Gamma \vdash e_1 := e_2 : 1$

What do we expect the logical relation at the ref type to be?

Recall what we had done for first-order state:

$V[\text{ref}]\rho = \{ (W,\text{ell}_1,\text{ell}_2) \mid \exists i.\ W.\omega.i = \{ ([\text{ell}_1 \mapsto n],[\text{ell}_2 \mapsto n]) \mid n \in \mathbb{Z} \} \}$

Consider:

$V[\text{ref}\ \tau]\rho = \{ (W,\text{ell}_1,\text{ell}_2) \mid \exists i.$
$\quad W.\omega.i = \{ ([\text{ell}_1 \mapsto v_1],[\text{ell}_2 \mapsto v_2]) \mid (W,v_1,v_2) \in V[\tau]\rho \} \}$

What about montonicity? As you move into the future, more things are related. The set in the island we introduced changes as W goes to W'. But the island must stay the same.

You could imagine using a STS to sort this out. In any case, we should be able to make this work with the simple notion of fixed invariants. We want to define our heap invariant, somehow, in terms of the logical relation. The world changes as you move to future worlds.
As you move to future worlds, the invariant about ref τ changes.
Somehow, the heap invariant has to dynamically change in a way that's dependent on the current world. The problem: There's nothing in our notion of world extension that allows for this change.

We want to be able to change the invariants in
a world in such a way that they depend on what the world is.

Claim: We want to write something like:

$V[\text{ref}\ \tau]\rho = \{ (W,\text{ell}_1,\text{ell}_2) \mid \exists i.$
$\quad W.\omega.i = \{ (W',[\text{ell}_1 \mapsto v_1],[\text{ell}_2 \mapsto v_2]) \mid (W',v_1,v_2) \in V[\tau]\rho \} \}$

The idea is to parametrize the heap relation by a world.
You want it to be the case that as we move from W to a future
world W_2, the invariant due to W remains exactly the same.

The world structure has changed: Each island now contains a world-indexed heap relation.

The idea (will need to be corrected later):

Island := Sub(World × Heap × Heap)

$h_1,h_2 : W$ if
$\quad W = (j,\omega) \wedge \omega \in \text{Island}^n \wedge$
$\quad \forall i\in\{1,2\}.\ \exists h^1\_i, \cdots, h^n\_i.$
$\quad\quad (h\_i = h^1\_i \uplus \cdots \uplus h^n\_i \wedge$

$\forall k \in 1..n.\ (W, h_1\text{^}k, h_2\text{^}k) \in \omega.k)$.

The problem: Islands now contain sets of worlds. We have a basic cardinality problem. We're trying to define world in terms of powersets of worlds.

The trick is remarkably simple, but kind of annoying. Propogate the step indices into places where you'd not previously wanted them. We want to define the set World_n by induction on n.

World = ∪_n World_n
World_n = { W = (j,ω) | j < n ∧ ∃m. ω ∈ Island^m_j }

> So World_n contains all the worlds useful for reasoning for less than n steps of computation.
>
> Note if $j_1 \le j_2$, then World_j_1 ⊆ World_j_2. We're building a chain of these indexed worlds. They just get bigger.

Island_n = { ι ∈ Sub(World_n × Heap × Heap) |
    $\forall (W,h_1,h_2) \in \iota.\ \forall W' \sqsupseteq W.\ (W',h_1,h_2) \in \iota$ }

> We baked in monotonicty. For now, trust Derek: We need it. The idea: We've made all invariants depend on the whole world. We need montonicity to, in a proof, focus on a particular island, advance the world, and "frame out" the islands we don't know about. (In the past, the islands did not change when moving to future worlds. Now they do.)
>
> Aside: The notion of world extension is now part of these mutually recursive definitions. Implicitly, when we write W' ⊒ W, we mean W' ⊒_j W where j = W'.j.
> The point: You only need to quantify over worlds that could possibly be a future world of W.

Our later modality now does a bit more. We seek the property
    If W ∈ World_n,
    then ▹W ∈ World_{n-1}.

If W = (j,ω) and j>0,
then define
    ▹W = (j-1,|ω|_{j-1})

Aside: Derek used $\llcorner - \lrcorner$ rather than $|-|$.

Where
$|(\iota_1, \cdots, \iota\_m)|\_n := (|\iota_1|\_n, \cdots, |\iota\_m|\_n)$
$|\iota|\_n := \{ (W,h_1,h_2) \mid (W,h_1,h_2) \in \iota \wedge W \in World\_n \}$

We now have to sqash things in world extension:
$W' \sqsupseteq W$ if
$W' = (j',\omega') \wedge W = (j,w) \wedge j' \leq j \wedge \omega' \sqsupseteq |\omega|\_j'$

$(\iota'_1,\ldots,\iota'\_m') \sqsupseteq (\iota_1,\ldots,\iota\_m)$ if
$m' \geq m \wedge$
$\forall k \in 1..m.\ \omega'.k = \omega.k.$

When do two heaps satisfy a world?

$h_1,h_2 : W$ if
$W = (j,\omega) \wedge \omega \in Island\^m\_j \wedge$
$\forall i \in \{1,2\}.\ \exists h^1\_i, \cdots, h\^m\_i.$
$(h\_i = h^1\_i \uplus \cdots \uplus h\^m\_i \wedge$
$\forall k \in 1..m.\ (\triangleright W, h_1\^k, h_2\^k) \in \omega.k).$

We have to squash the current world down one level.
We solved the circularity with step indices, but got
a weaker notion of world satisfaction than we were shooting
for. Since we must always take a step of computation to observe
differences between heaps, we do not lose anything.

$V[ref\ \tau]\rho = \{ (W,ell_1,ell_2) \mid$
$W \in World\_n \wedge$
$\exists i.W.\omega.i = \{ (W',[ell_1 \mapsto v_1],[ell_2 \mapsto v_2]) \mid (W',v_1,v_2) \in V[\tau]\rho$
$\wedge\ W' \in World\_\{n-1\} \} \}$

It's important to be clear what the world index is for W'.

—

Lemma (Compatibility for get):
If $(W,v_1,v_2) \in V[ref\ \tau]\rho$
then $(W,!v_1,!v_2) \in E[\tau]\rho$.
Proof:
WK: $v_1=ell_1$, $v_2=ell_2$. $\exists i.\ W.\omega.i = \{\cdots\}$.
Suppose $(W,K_1,K_2) \in K[\tau]\rho$ and $h_1,h_2 : W$.
TS: $h_1;K_1[!ell_1] \downarrow\downarrow\_\{W.j\}\ h_2; K_2[!ell_2]$.

From {⋯},
WK: $h\_i = h'\_i \uplus h''\_i \wedge$
$(\triangleright W, h'_1, h'_2) \in$ {⋯}
$\Rightarrow$ $h'_1 = [ell_1 \mapsto v'_1] \wedge h'_2 = [ell_2 \mapsto v'_2]$.
$\Rightarrow$ $(\triangleright W, v'_1, v'_2) \in V[\tau]\rho$.

STS: $h_1;K_1[v'_1] \downarrow\downarrow\_\{W.j-1\}\ h_2;\ K_2[v'_2]$.

We're moving from W to ▹W. So
STS: $h_1,h_2 : \triangleright W \wedge$
$(\triangleright W, K_1, K_2) \in K[\tau]\rho \wedge$
$(\triangleright W, v'_1, v'_2) \in V[\tau]\rho$.
The second and third conjuncts are easy, by monotonicty.
We're moving to ▹W since we only know
h_1 and h_2 are related in future worlds from W.
Note also that $W.j-1 = (\triangleright W).j$.

The first conjunct follows from monotonicty of
the islands. (This is one point where
the newly-introduced monotonicity comes in.)

Aside: Note that the steps really mattered.
Q.E.D.

The other compatibility proofs are not so difficult.

—

Aside from a bit of tedium (sqashing the worlds down the lowest
step-level as needed), the examples we've looked at so far don't really
change at all when moved to this more general model.
The invariants we looked at so far were fairly simple. We imposed
invariants on int refs that didn't care about the world more generally.
The generalization here: Those islands take a world parameter, then ignore it.

Our invariants only need to care about the world when we're working
with higher-order state.

In essense, this higher-order state is another orthogonal extension.

— Brain teaser

Here's an interesting example you can puzzle over until next time.

We proved it in our POPL'09 paper using a gross hack, exploiting the fact that we didn't quite know what we were doing.

This example won't be provable in the current model. (And that's good. We shouldn't be able to prove it.)

Example (Callback with lock):

```
    τ = ((1 ⟶ 1) ⟶ 1) × (1 ⟶ int)
    e₁ = C[f(); x := !x + 1]
    e₂ = C[let n = !x in f(); x := n + 1]
    C = let b = ref true in let x = ref 0 in
          <λf.if !b then (b := false; •; b := true) else (),
           λ_.!x>.
```

This is basically an object with two methods and two pieces of local state. The lock b $\hookrightarrow$ true iff “unlocked”.
The second method gets the current value of a counter.
The first method gives up if locked; otherwise, it acquires the lock, “does something”, and release the lock.
The point is, the hole happens in a critical section.
In e_1, we invoke the callback and bump the counter.
In e_2, we record the value in the counter, invoke the callback, then bump the counter based on the initial value of the counter.

What does it mean for e_1 and e_2 to be equivalent? We'd need to know f cannot change the counter (without putting it back).

Imagine callcc and first-order state. Then f could screw up our invariants about b by returning prematurely. This does not break the equivalence.

Now imagine callcc with higher-order state: Together these do ruin the equivalence. See page 509 of the Dreyer, Neis, Birkedal JFP paper for the distinguishing context. The idea is to stick f's continuation into a reference cell. Once the e_i return, jump to the saved continuation.