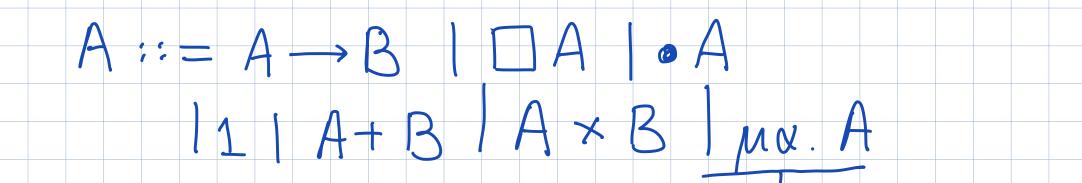
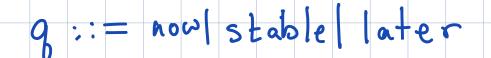
An Alternate Presentation







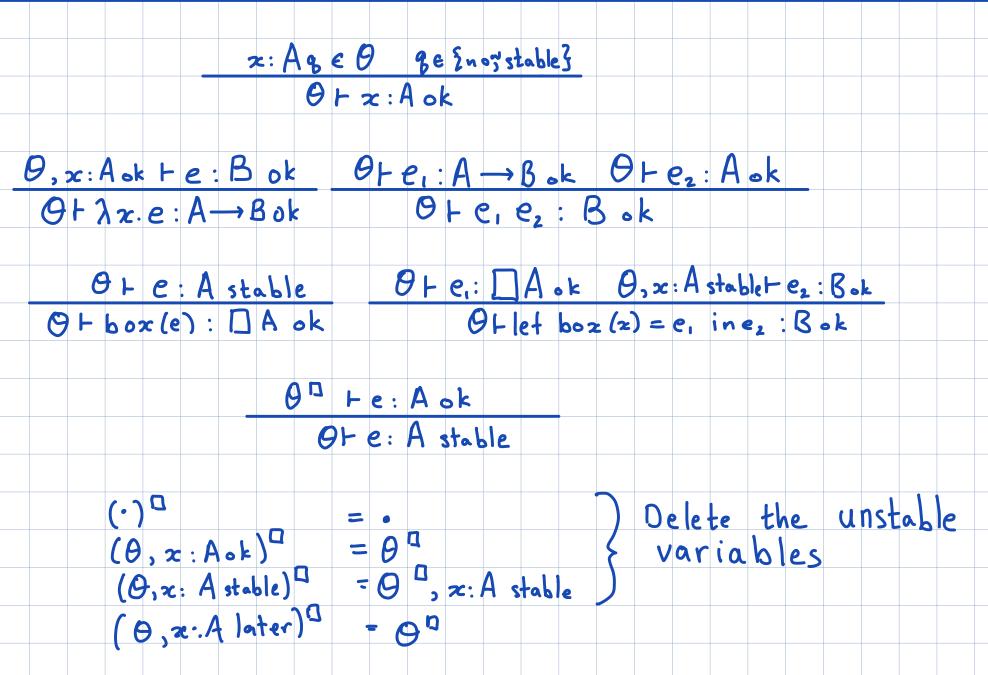


LTyping judge parameterized

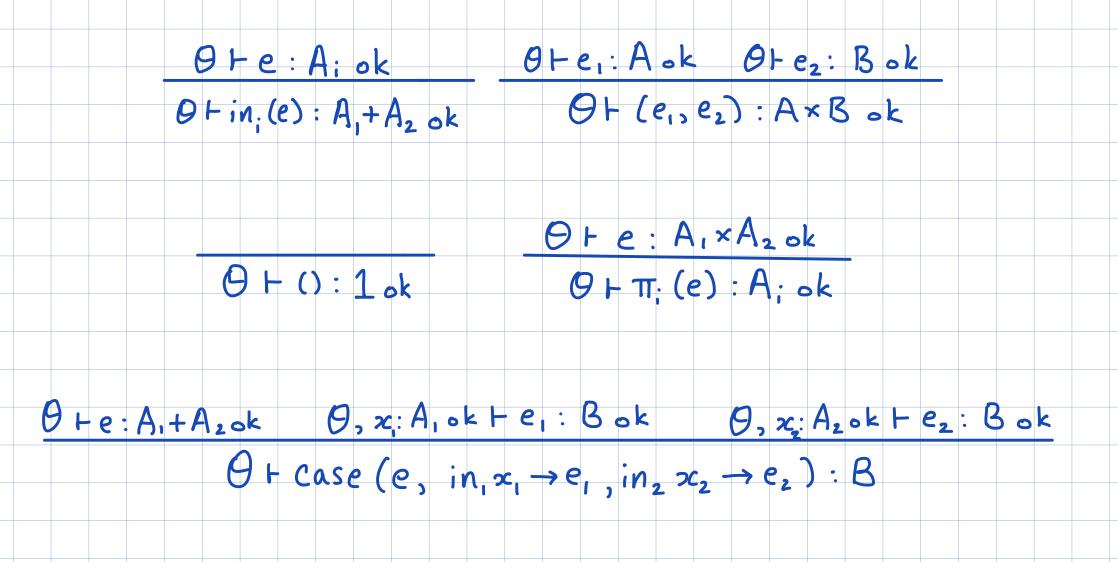
by mode

recursive types!

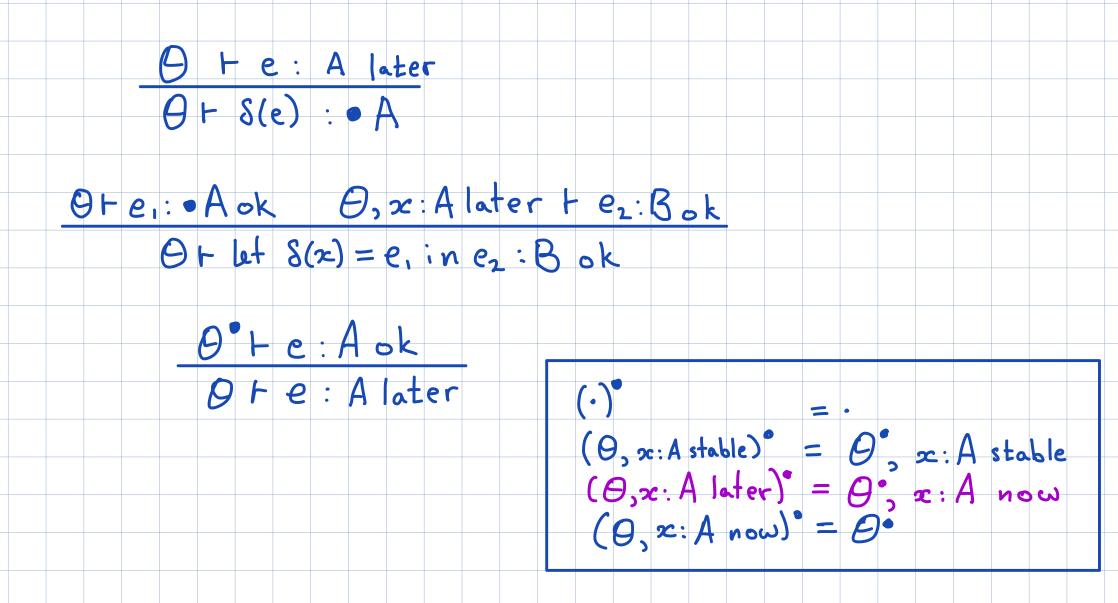




Sums and Products



The Later Modality



Recursive Types



 $D^{D}, x: A$ later $\vdash e: A \circ k$ $O \vdash fix x: A \cdot e: A \circ k$

na. Axa > Ax · (na. Axa) > Ax · (Ax · (na. Axa))

What the Modalities Mean

DA are stable values

- They do not change as time passes

- Stable variables are persistent:

once available, always available

• A are "As tomorrow"

- They are lazy, and scheduled for the next tick - Later variables cannot be used now, but turn into ok variables in one tick



bool = 1+1 true = in, () false = in₂()

promote: bool -> [bool

promote b = case(b) $in_1 \longrightarrow box(true),$ $in_2 \longrightarrow box(false))$

 $U \rightarrow \Box U$ definable for

 $U := 1 | u + u | u \times u | \Box A$

Guarded Recursion

fix x: A. e is guarded recursion

- It is a recursive definition, but

the recursive call is only on the next tick

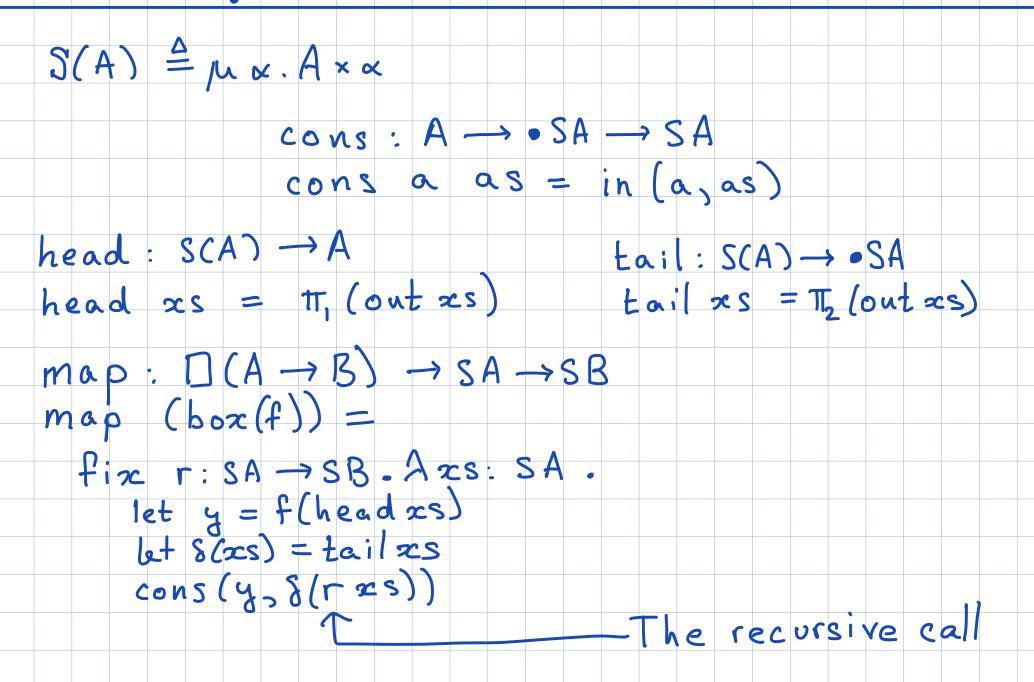
- Works well with guarded recursive types

Guarded Recursive Types

- Each recursive occurence is later

- S(A) = Ma. Axa SCA) = ma. A×a $\theta + e: S(A)$ $\Theta \vdash out(e): A \times \bullet S(A)$ [• $\mu \alpha \cdot (A \times \alpha) / \alpha](A \times \alpha)$ $= A \times \bullet \mu \alpha . (A \times \alpha)$ $= A \times \cdot S(A)$

Example: Streams



More Examples

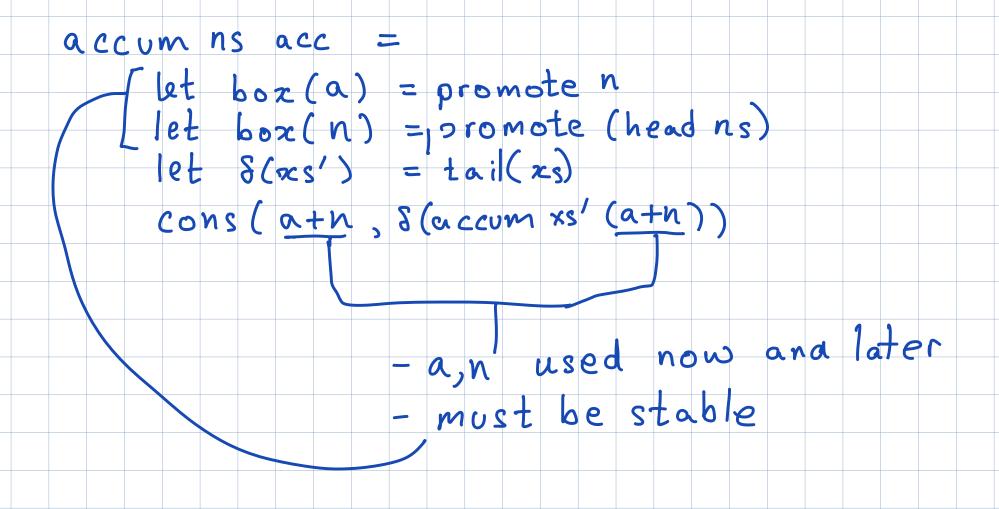
 $accum : S(int) \rightarrow int \rightarrow S(int)$

a ccum ns acc = let box(a) = promote acclet box(n) = promote (head ns)let $\delta(xs') = tail(xs)$ cons(a+n, $\delta(accum xs'(a+n))$

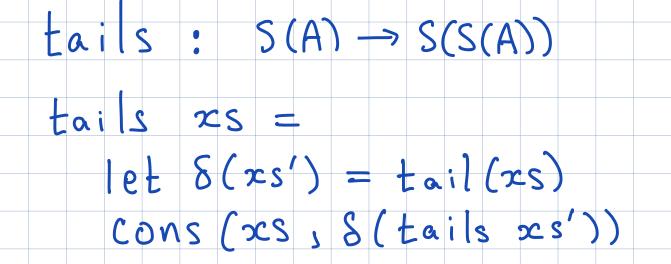
accum [0,1,2,...] O = [0,1,3,6,10,...]

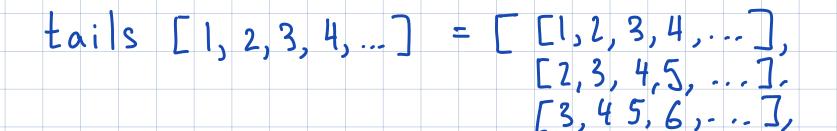
More Examples





More Examples





Unfold for Streams

unfold: $\Box(X \rightarrow B \times \bullet X) \rightarrow X \rightarrow S(B)$

unfold (box f) x =let $(b, \delta(x')) = f(x)$ cons $(b, \delta(unfold (boxf) x'))$



 $switch: \Box int \rightarrow s(A) \rightarrow s(A) \rightarrow s(A)$

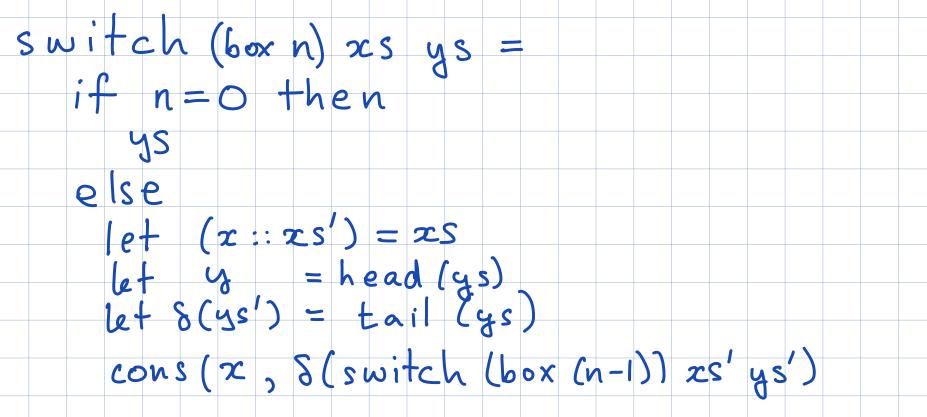
switch (box n) xs ys = if n=0 then

 $\begin{array}{c} ys \\ e \mid se \\ |ef (x :: zs') = zs \\ ef (y :: ys') = ys \end{array}$

cons(x, S(switch (box (n-1)) xs' ys')



 $switch: \Box int \rightarrow s(A) \rightarrow s(A) \rightarrow s(A)$



Events

A stream S(A) yields an infinite stream of As

Some things happen <u>once</u>

 $E(A) = \mu \alpha \cdot A + \alpha \qquad \begin{pmatrix} E(A) = Done of A \\ Wait of e E(A) \end{pmatrix}$



 $E(A) = \mu \alpha \cdot A + \alpha \qquad \begin{pmatrix} E(A) = Done of A \\ | Wait of \bullet E(A) \end{pmatrix}$

return : $A \rightarrow E(A)$ return x = Done(x)

bind: $E(A) \rightarrow \Box(A \rightarrow E(B)) \rightarrow E(B)$

bind (Done ∞) (box f) = f π

bind (Wait $\delta(e')$) (box f) = Wait($\delta(binde'(box f))$)

Events

 $E(A) = \mu \alpha . A + \alpha \qquad E(A) = Done of A$ |Wait of e E(A) $map : [](A \to B) \to E(A) \to E(B)$

map (boxf) (Done a) = Done (fa) map (boxf) (Wait S(e)) = Wait S(map (boxf) e)



 $E(A) = \mu \alpha \cdot A + \alpha$ E(A) = Done of A $|Wait of \cdot E(A)$

 $select: E(A) \rightarrow E(B) \rightarrow E(A \times E(B)) + (E(A) \times B)$

select (Done a) $e_2 = Done(in, (a, e_2))$ select e_1 (Done b) = Done(inz(e_1, b)) select (Wait S(e'_1)) (Wait S(e'_2)) = Wait(S(select e'_1, e'_2))

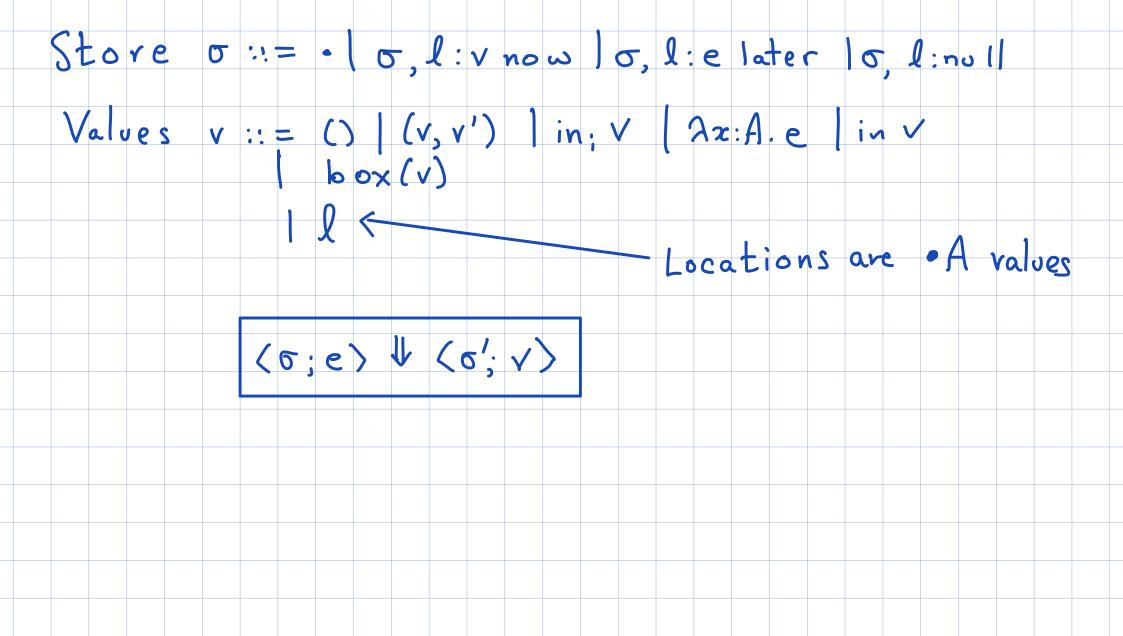
Kecursion Let $X = \mu \alpha \cdot \Box (\alpha \rightarrow A)$ selfapp : $(A \rightarrow A) \rightarrow X \rightarrow A$ selfapp f v =let $box(w: X \rightarrow A) = out(r)$ in Variant $f \delta(\omega (in(box \omega)))$ of the Ycombinator $fix : \Box(\bullet A \to A) \to A$ fix (box f) =selfapp f (in(box (selfappf)))

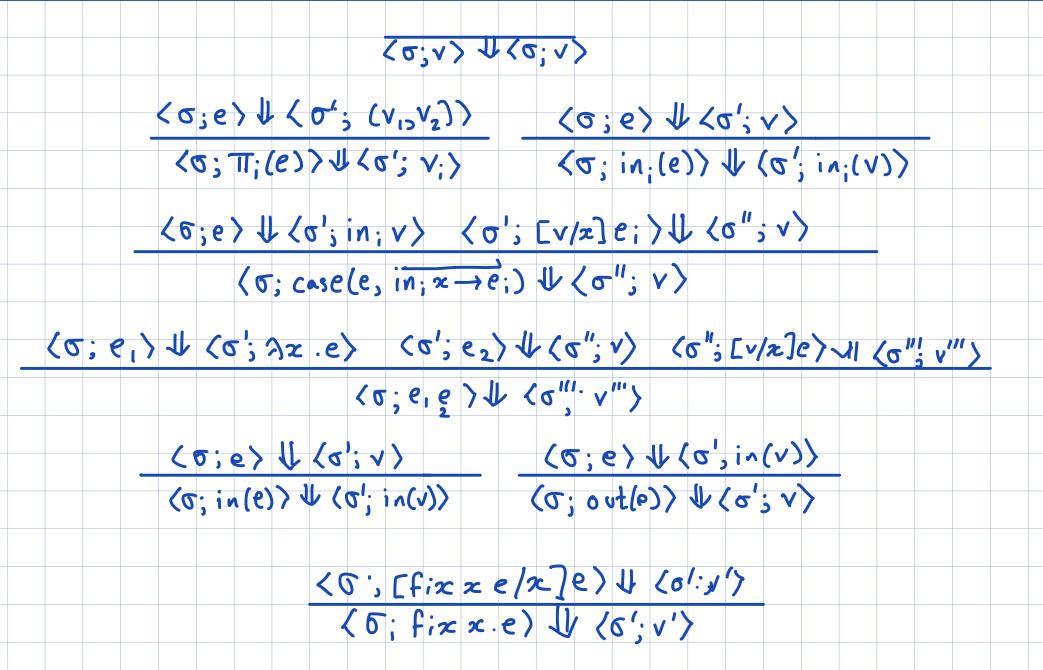
· Values of • A are lazily evaluated

· We must preserve sharing for efficiency

. We implement • A with a memoized

code pointer





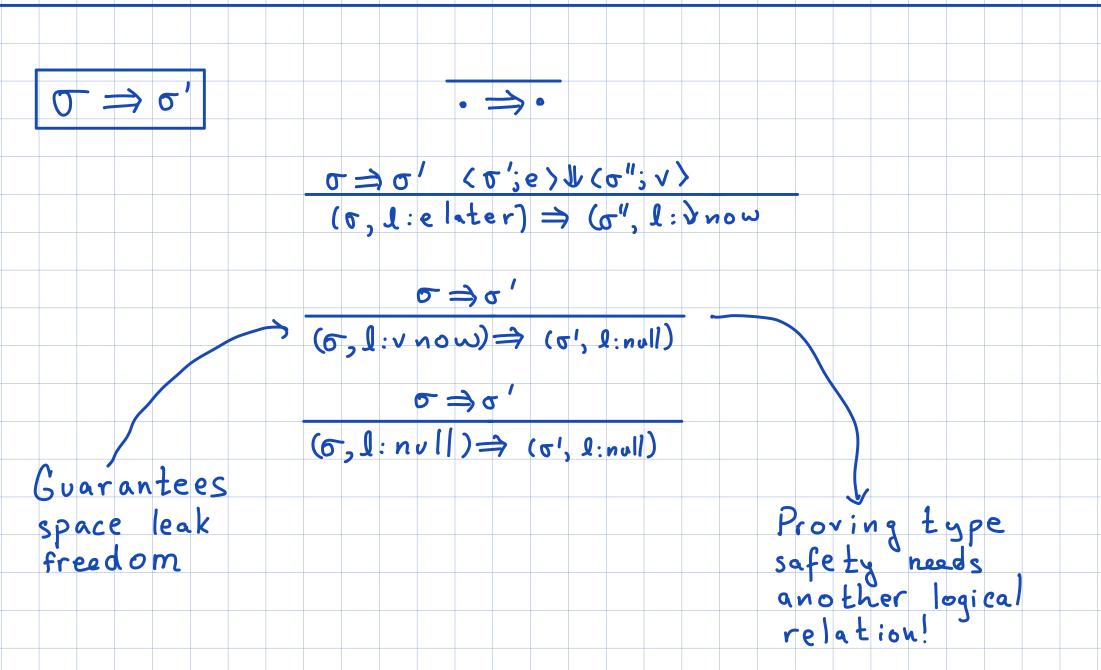
 $\langle \sigma; \delta(e) \rangle \Downarrow \langle [\sigma, l:e | ater]; l \rangle$

 (σ_{i},e_{i}) \forall $\langle \sigma'_{i},l \rangle \langle \langle \sigma'_{i},l \rangle \langle \langle \sigma''_{i},v'' \rangle \rangle$

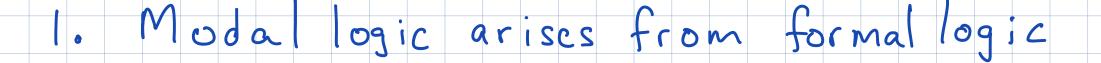
 $\langle \sigma; b \neq S(z) = e, in e_2 \rangle \Downarrow \langle \sigma''; v'' \rangle$

 $\frac{l:v \text{ now } \in \sigma}{\langle \sigma: 1 l \rangle \Downarrow \langle \sigma; v \rangle}$

Tick Semantics

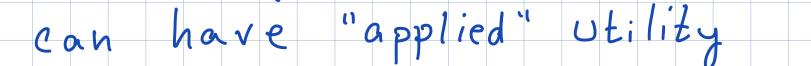


Conclusion



2. Same rules can have many applications





4. Many more applications to be

discovered

