Falling Down the Naming Well

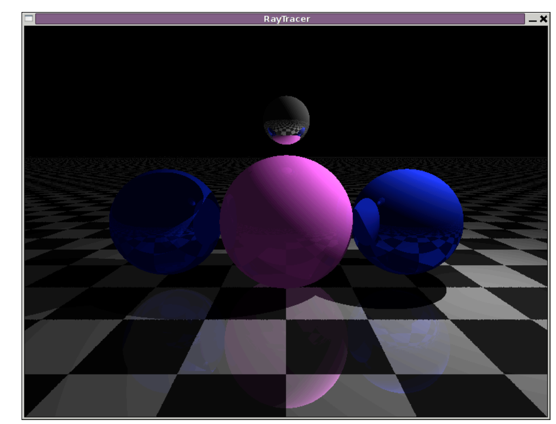
Ben Lippmeier
University of New South Wales
FP-Syd 2011/04/21

One slide summary of Disciple / DDC

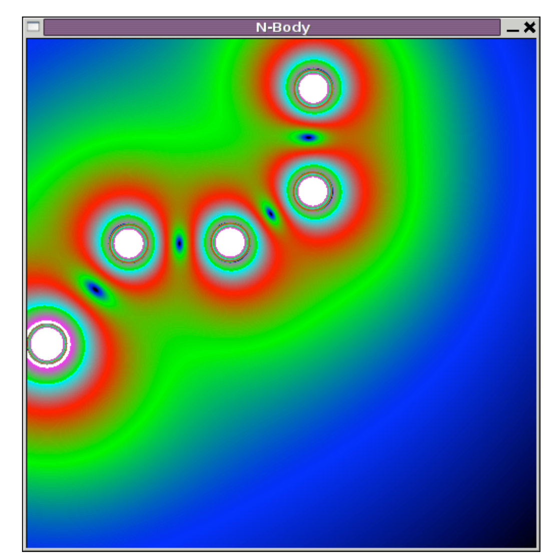
- Disciple is an explicitly lazy dialect of Haskell. Many Haskell programs will work with only minor changes.

- The type system includes effect and closure typing, along with mutability polymorphism. Most of the extras can be inferred.

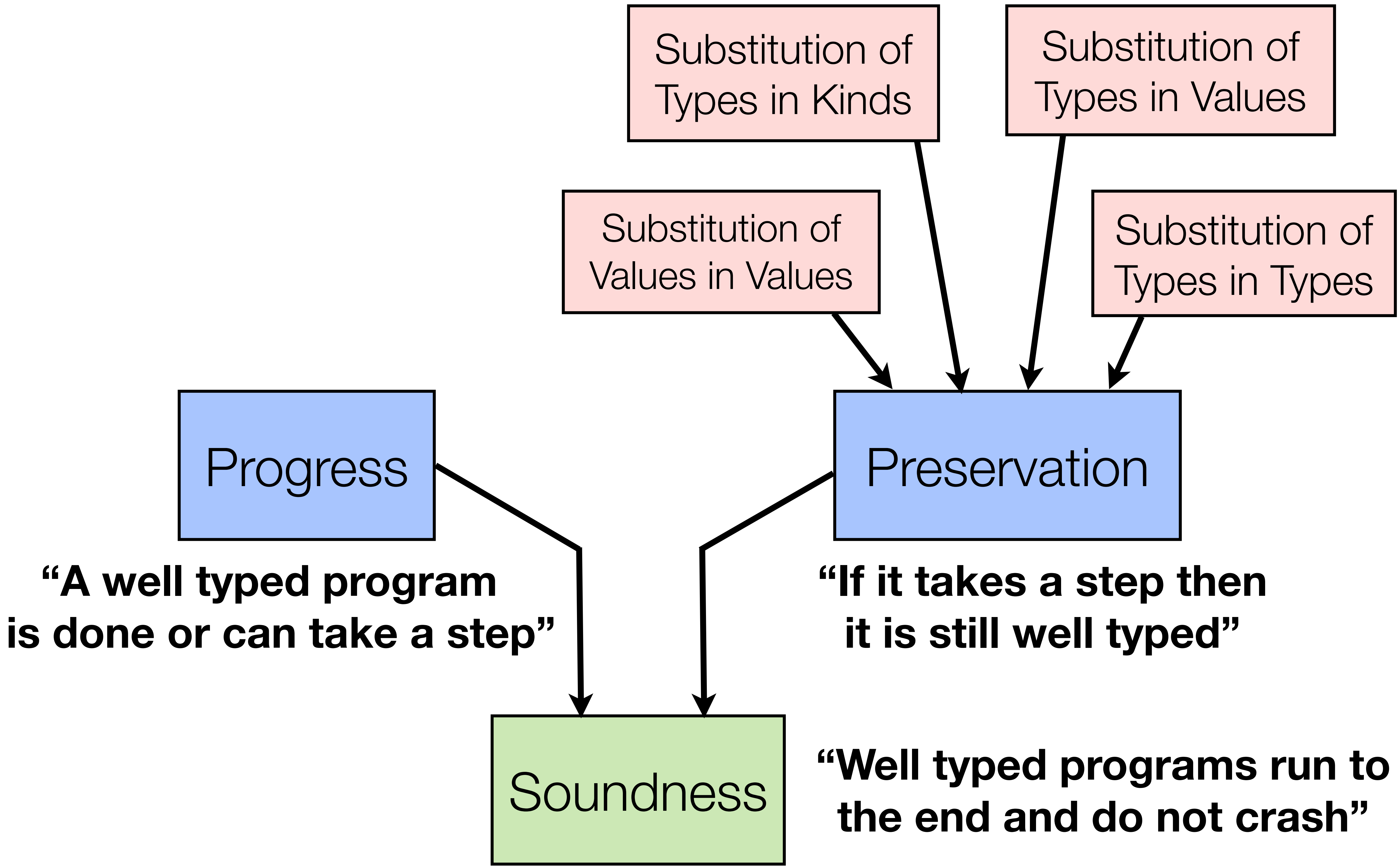
- The compiler (DDC) is still a work in progress.



```
add :: forall r1 r2 r3
    .  Int r1 -> Int r2 -(e1)> Int r3
    :- e1 = Read r1 + Read r2 + Alloc r3
```



Basic Soundness Proof



Drowning, not waving.

Lemma 1.8 (Substitution of Values in Values)

$$\begin{array}{ll} \text{If} & \Gamma, x : \tau_2 \,|\, \Sigma \vdash t \;::\; \tau_1 \,;\, \sigma \\ \text{and} & \Gamma \,|\, \Sigma \vdash v^\circ \;::\; \tau_2 \,;\, \bot \\ \text{then} & \Gamma \,|\, \Sigma \vdash t[v^\circ / x] \;::\; \tau_1 \,;\, \sigma \end{array}$$

Case: $t = t_1\ \varphi_2$ / TyAppT

$$\frac{(3)\ \Gamma, x : \tau_2 \,|\, \Sigma \vdash t_1 \;::\; \forall(a : \kappa_{11}).\, \varphi_{12} \,;\, \sigma \qquad (4)\ \Gamma, x : \tau_2 \,|\, \Sigma \vdash_{\mathrm{T}} \varphi_2 \;::\; \kappa_{11}}{(\underline{1})\ \Gamma, x : \tau_2 \,|\, \Sigma \vdash t_1\ \varphi_2 \;::\; \varphi_{12}[\varphi_2 / a] \,;\, \sigma[\varphi_2 / a]}$$

$(\underline{2})$	$\Gamma \,|\, \Sigma \vdash v^\circ \;::\; \tau_2 \,;\, \bot$	(assume)
(5)	$\Gamma \,|\, \Sigma \vdash t_1[v^\circ / x] \;::\; \forall(a : \kappa_{11}).\, \varphi_{12} \,;\, \sigma$	(IH 3 2)
(6)	$\Gamma \,|\, \Sigma \vdash_{\mathrm{T}} \varphi_2 \;::\; \kappa_2$	(Str. Type Env 4)
(7)	$\Gamma \,|\, \Sigma \vdash t_1[v^\circ / x]\ \varphi_2 \;::\; \varphi_{12}[\varphi_2 / a] \,;\, \sigma[\varphi_2 / a]$	(TyAppT 5 6)
(8)	$\Gamma \,|\, \Sigma \vdash (t_1\ \varphi_2)[v^\circ / x] \;::\; \varphi_{12}[\varphi_2 / a] \,;\, \sigma[\varphi_2 / a]$	(Def. Sub. 7)

Drowning, not waving.

Lemma 1.9 (*Substitution of Types in Values*)

$$\begin{array}{ll} \text{If} & \Gamma,\, a : \kappa_2 \,|\, \Sigma \vdash t :: \tau_1 ; \sigma \\ \text{and} & \Gamma \,|\, \Sigma \vdash_{\mathrm{T}} \varphi_2 :: \kappa_2 \\ \text{then} & \Gamma[\varphi_2/a] \,|\, \Sigma \vdash t[\varphi_2/a] :: \tau_1[\varphi_2/a] ; \sigma[\varphi_2/a] \end{array}$$

Case: $t = t_{11}\ \varphi_{12}$ / TyAppT

$$\frac{(3)\ \Gamma,\, a : \kappa_3 \,|\, \Sigma \vdash t_1 :: \forall (a_1 : \kappa_{11}).\, \varphi_{12} ; \sigma_1 \qquad (4)\ \Gamma,\, a : \kappa_3 \,|\, \Sigma \vdash_{\mathrm{T}} \varphi_2 :: \kappa_{11}}{(\underline{1})\ \Gamma,\, a : \kappa_3 \,|\, \Sigma \vdash t_1\ \varphi_2 :: \varphi_{12}[\varphi_2/a_1] ; \sigma_1[\varphi_2/a_1]}$$

(2̲) $\Gamma \,|\, \Sigma \vdash_{\mathrm{T}} \varphi_3 :: \kappa_3$ (assume)

(5) $\Gamma[\varphi_3/a] \,|\, \Sigma$
$\vdash t_1[\varphi_3/a] :: (\forall (a_1 : \kappa_{11}).\, \varphi_{12})[\varphi_3/a] ; \sigma_1[\varphi_3/a]$ (IH 3 2)

(6) $\Gamma[\varphi_3/a] \,|\, \Sigma$
$\vdash t_1[\varphi_3/a] :: \forall (a_1 : \kappa_{11}[\varphi_3/a]).\, \varphi_{12}[\varphi_3/a] ; \sigma_1[\varphi_3/a]$ (Def. Sub. 5)

(7) $\Gamma[\varphi_3/a] \,|\, \Sigma \vdash_{\mathrm{T}} \varphi_2[\varphi_3/a] :: \kappa_{11}[\varphi_3/a]$ (Sub. Type/Type 4 2)

(8) $\Gamma[\varphi_3/a] \,|\, \Sigma \vdash t_1[\varphi_3/a]\ \ \varphi_2[\varphi_3/a]$
$:: (\varphi_{12}[\varphi_3/a])[\varphi_2[\varphi_3/a]/a_1] ; (\sigma_1[\varphi_3/a])[\varphi_2[\varphi_3/a]/a_1]$ (TyAppT 6 7)

(9) $\Gamma[\varphi_3/a] \,|\, \Sigma \vdash (t_1\ \varphi_2)[\varphi_3/a]$
$:: (\varphi_{12}[\varphi_2/a_1])[\varphi_3/a] ; (\sigma_1[\varphi_2/a_1])[\varphi_3/a]$ (Def. Sub. 8)

...and typing this all up in Latex isn't fun either...

```
\statement{
        If      &~ $\tyJudge{\Gamma, \ a : \kappa_2}{\Sigma}{t}{\tau_1}{\sigma}$
\\
        \ and   &~ $\kiJudge{\Gamma}{\Sigma}{\varphi_2}{\kappa_2}$
\\
        then    &~ $\tyJudge{\Gamma[\varphi_2/a]}{\Sigma}
                            {t[\varphi_2/a]}{\tau_1[\varphi_2/a]}{\sigma[\varphi_2/a]}$ \\
}

\tabbedstmts{
        (\un{2})
                \> $\kiJudgeGS{\varphi_3}{\kappa_3}$
                \> (assume)
\\[1ex]
        (5)     \> ${\Gamma[\varphi_3/a]} ~|~ \Sigma ~\vdash~  t_1[\varphi_3/a]$
\\
                \> \quad
                        $::~  (\tyForall{a_1}{\kappa_{11}}{\varphi_{12}})[\varphi_3/a]
                         ~;~  { \sigma_1[\varphi_3/a] }$
                \> (IH 3 2)
\\[1ex]
        (6)     \> ${\Gamma[\varphi_3/a]} ~|~ \Sigma \vdash {t_1[\varphi_3/a]}$
\\
                \> \quad
                        $::~    \tyForall
                                        {a_1}
                                        {\kappa_{11}[\varphi_3/a]}
                                        {\varphi_{12}[\varphi_3/a] }
                        ~;~     \sigma_1[\varphi_3/a]$
                \> (Def. Sub. 5)
```

Let's just not and say we did.

Lemma: (Weaken Type Environment)

$$\begin{array}{ll} \text{If} & \Gamma \,|\, \Sigma \vdash t :: \tau_1 ; \sigma \\ \text{and} & x \notin fv(t) \\ \text{then} & \Gamma, x : \tau_2 \,|\, \Sigma \vdash t :: \tau_1 ; \sigma \end{array} \qquad \begin{array}{ll} \text{If} & \Gamma \,|\, \Sigma \vdash_{\text{T}} \varphi :: \kappa \\ \text{and} & a \notin fv(\varphi) \\ \text{then} & \Gamma, a : \varphi_2 \,|\, \Sigma \vdash_{\text{T}} \varphi :: \kappa \end{array}$$

By induction over the derivations of the first statements of each.

Proving something more interesting

Lemma 1.12 (*Visible Actions*)
If the reduction of an expression reads, writes or allocates mutable data in a pre-existing region then it has the corresponding effect.

$$
\begin{array}{ll}
\text{If} & \emptyset \,|\, \Sigma \vdash t :: \tau\,;\sigma \ \text{ and } \ H\,;t \downarrow H'\,;t' \vdash B \\
\text{and} & \Sigma \models H \quad \text{and} \quad \Sigma \in H \quad \text{and} \quad \underline{\rho} \in \Sigma \\
\text{then} & (\text{If mread } l^{\rho} \in B \text{ for some } l \text{ then } \Gamma \,|\, \Sigma \vdash \mathit{Read}\ \underline{\rho} \sqsubseteq \sigma) \\
\text{and} & (\text{If write } l^{\rho} \in B \text{ for some } l \text{ then } \Gamma \,|\, \Sigma \vdash \mathit{Write}\ \underline{\rho} \sqsubseteq \sigma) \\
\text{and} & (\text{If malloc } l^{\rho} \in B \text{ for some } l \text{ then } \Gamma \,|\, \Sigma \vdash \mathit{Alloc}\ \underline{\rho} \sqsubseteq \sigma)
\end{array}
$$

Proving something more interesting

Lemma 1.14 (*Non-Interfering Effects Yield Commutable Actions*)

$$
\begin{array}{ll}
\text{If} & (1)\ \Gamma \,|\, \Sigma_1 \vdash t_1 :: \tau_1 ; \sigma_1 \ \text{ and } \ (2)\ H_1 ; t_1 \downarrow H_1' ; t_1' \vdash B_1 \\
\text{and} & (3)\ \Gamma \,|\, \Sigma_2 \vdash t_2 :: \tau_2 ; \sigma_2 \ \text{ and } \ (4)\ H_2 ; t_2 \downarrow H_2' ; t_2' \vdash B_2 \\
\text{and} & (5)\ \text{NonInterfering}(\Gamma,\ \sigma_1,\ \sigma_2) \\
\text{then} & \text{CommutableActions}(B_1,\ B_2)
\end{array}
$$

Case: $t = K\ \overline{\varphi}\ \overline{t}$ / TyAlloc / EiAlloc

$$\frac{(7)\ \Gamma\,|\,\Sigma \vdash_{\tau} \underline{\rho} :: \% \qquad \begin{array}{c}(6)\ \overline{\Gamma\,|\,\Sigma \vdash t :: \tau_i[\underline{\rho}/r\ \overline{\varphi'/a}]\,;\,\sigma_i}^{\,i\leftarrow 0..n} \\ (8)\ K :: \forall(r:\%).\forall(\overline{a:\kappa}).\overline{\tau} \to T\ r\ \overline{a} \in \mathrm{ctorTypes}(T)\end{array}}{(\underline{1})\ \Gamma\,|\,\Sigma \vdash K\ \underline{\rho}\ \overline{\varphi'}\ \overline{t} :: T\ \underline{\rho}\ \overline{\varphi'}\,;\,\sigma_0 \vee \sigma_1 \ldots \vee \sigma_n \vee Alloc\ \underline{\rho}}\ (\mathrm{TyAlloc})$$

$$\frac{(9)\ \overline{H_j\,;\,t_j \downarrow H_{j+1}\,;\,v_j^{\circ} \vdash B_j}^{\,j\leftarrow 0..n} \qquad (10)\ \rho \in H_{n+1} \qquad (11)\ l\ \mathrm{fresh}}{(\underline{2})\ H_0\,;\,K\ \underline{\rho}\ \overline{\varphi_i'}\ \overline{t_j} \downarrow H_{n+1},\ l \overset{\rho}{\mapsto} C_k\ \overline{v_j^{\circ}}\,;\,l \vdash \bigcup \overline{B_j} \cup B' \cup \{\mathrm{test}\ \rho\}}\ (\mathrm{EiAlloc})$$

$$(12)\ B' = \begin{cases} \{\mathrm{malloc}\ l^{\rho}\} & \text{if mutable } \rho \in H_{n+1}, \\ \{\mathrm{palloc}\ l^{\rho}\} & \text{if const } \rho \in H_{n+1}. \end{cases}$$

(3..5)	$\Sigma \models H_0,\ \Sigma \vdash H_0,\ \underline{\rho'} \in \Sigma$	(assume)
(13)	If mread $l^{\rho'} \in B_j \ldots \Gamma\,|\,\Sigma' \vdash Alloc\ \underline{\rho'} \sqsubseteq \sigma_j$	(Via IH, sim. to TyAppT case)
(14)	malloc $l^{\rho'} \in B'$	(assume)
(15)	Case $\underline{\rho'} \neq \underline{\rho}$:	
(16)	Case (mutable ρ) $\in H_{n+1}$	
(17)	$B' = \{\mathrm{malloc}\ l^{\rho}\}$	(12 16)
(18)	Suppose $\neg(\Gamma\,|\,\Sigma' \vdash Alloc\ \underline{\rho'} \sqsubseteq Alloc\ \underline{\rho})$	
(19)	Contradiction	(14 15 17)
(20)	Cases for (const ρ) $\in H_{n+1}$ and $B' = \emptyset$ similarly.	
(21)	Case $\underline{\rho'} = \underline{\rho}$:	
(22)	Case (mutable ρ) $\in H_{n+1}$	
(23)	$B' = \{\mathrm{malloc}\ l^{\rho}\}$	(12 22)
(24)	$\Gamma\,|\,\Sigma' \vdash Alloc\ \underline{\rho} \sqsubseteq Alloc\ \underline{\rho}$	(immediate)
(25)	If malloc $l^{\rho'} \in B'$ then $\Gamma\,|\,\Sigma' \vdash Alloc\ \underline{\rho} \sqsubseteq Alloc\ \underline{\rho'}$	(Imp. Intro 14 - 24)

It needs one of these for every expression form.

Many details are still omitted. **ok, bored now...**

Implicit assumptions in informal proofs.

Lemma 1.8 (Substitution of Values in Values)

$$\begin{array}{ll} \text{If} & \Gamma, x : \tau_2 \,|\, \Sigma \vdash t \;::\; \tau_1 \,;\, \sigma \\ \text{and} & \Gamma \,|\, \Sigma \vdash v^\circ \;::\; \tau_2 \,;\, \bot \\ \text{then} & \Gamma \,|\, \Sigma \vdash t[v^\circ/x] \;::\; \tau_1 \,;\, \sigma \end{array}$$

(MUMBLE)

... assuming no free variables in *v* are bound by *t* ...

Variable capture... I hates it.

```
(\y. \x. x + y) (x * 2) 5
 => (\x. x + (x * 2)) 5
 => 5 + (5 * 2)
 => 15
```

capturing

```
(\y. \x. x + y) (x * 2) 5
 => (\z. z + (x * 2)) 5
 => 5 + (x * 2)
```

non-capturing

You can sneak past with closed values

```
Lemma subst_value_value
 :   forall env x val t1 T1 T2
 ,   (forall z, freeX z val -> noBindsX z t1)
 -> TYPE (extend env x T2) t1  T1
 -> TYPE env                   val T2
 -> TYPE env  (subst x val t1) T1.
```

```
Lemma subst_value_value_closed
 :   forall env val t1 T1 T2
 ,   closedX val
 -> TYPE (extent env x T2) t1  T1
 -> TYPE env                   val T2
 -> TYPE env  (subst x val t1) T1.
```

This trick isn't enough for System-F

Lemma 1.9 (Substitution of Types in Values)

$$\begin{array}{ll} \text{If} & \Gamma,\, a : \kappa_2 \,|\, \Sigma \vdash t \,::\, \tau_1 \,;\, \sigma \\ \text{and} & \Gamma \,|\, \Sigma \vdash_{\mathrm{T}} \varphi_2 \,::\, \kappa_2 \\ \text{then} & \Gamma[\varphi_2/a] \,|\, \Sigma \vdash t[\varphi_2/a] \,::\, \tau_1[\varphi_2/a] \,;\, \sigma[\varphi_2/a] \end{array}$$

Case: $t = t_{11}\ \varphi_{12}$ / TyAppT

$$\frac{(3)\ \Gamma,\, a : \kappa_3 \,|\, \Sigma \vdash t_1 \,::\, \forall(a_1 : \kappa_{11}).\, \varphi_{12} \,;\, \sigma_1 \qquad (4)\ \Gamma,\, a : \kappa_3 \,|\, \Sigma \vdash_{\mathrm{T}} \varphi_2 \,::\, \kappa_{11}}{(\underline{1})\ \Gamma,\, a : \kappa_3 \,|\, \Sigma \vdash t_1\ \varphi_2 \,::\, \varphi_{12}[\varphi_2/a_1] \,;\, \sigma_1[\varphi_2/a_1]}$$

(2) $\Gamma \,|\, \Sigma \vdash_{\mathrm{T}} \varphi_3 \,::\, \kappa_3$ (assume)

(5) $\Gamma[\varphi_3/a] \,|\, \Sigma$
$\vdash t_1[\varphi_3/a] \,::\, (\forall(a_1 : \kappa_{11}).\, \varphi_{12})[\varphi_3/a] \,;\, \sigma_1[\varphi_3/a]$ (IH 3 2)

(6) $\Gamma[\varphi_3/a] \,|\, \Sigma$
$\vdash t_1[\varphi_3/a] \,::\, \forall(a_1 : \kappa_{11}[\varphi_3/a]).\, \varphi_{12}[\varphi_3/a] \,;\, \sigma_1[\varphi_3/a]$ (Def. Sub. 5)

(7) $\Gamma[\varphi_3/a] \,|\, \Sigma \vdash_{\mathrm{T}} \varphi_2[\varphi_3/a] \,::\, \kappa_{11}[\varphi_3/a]$ (Sub. Type/Type 4 2)

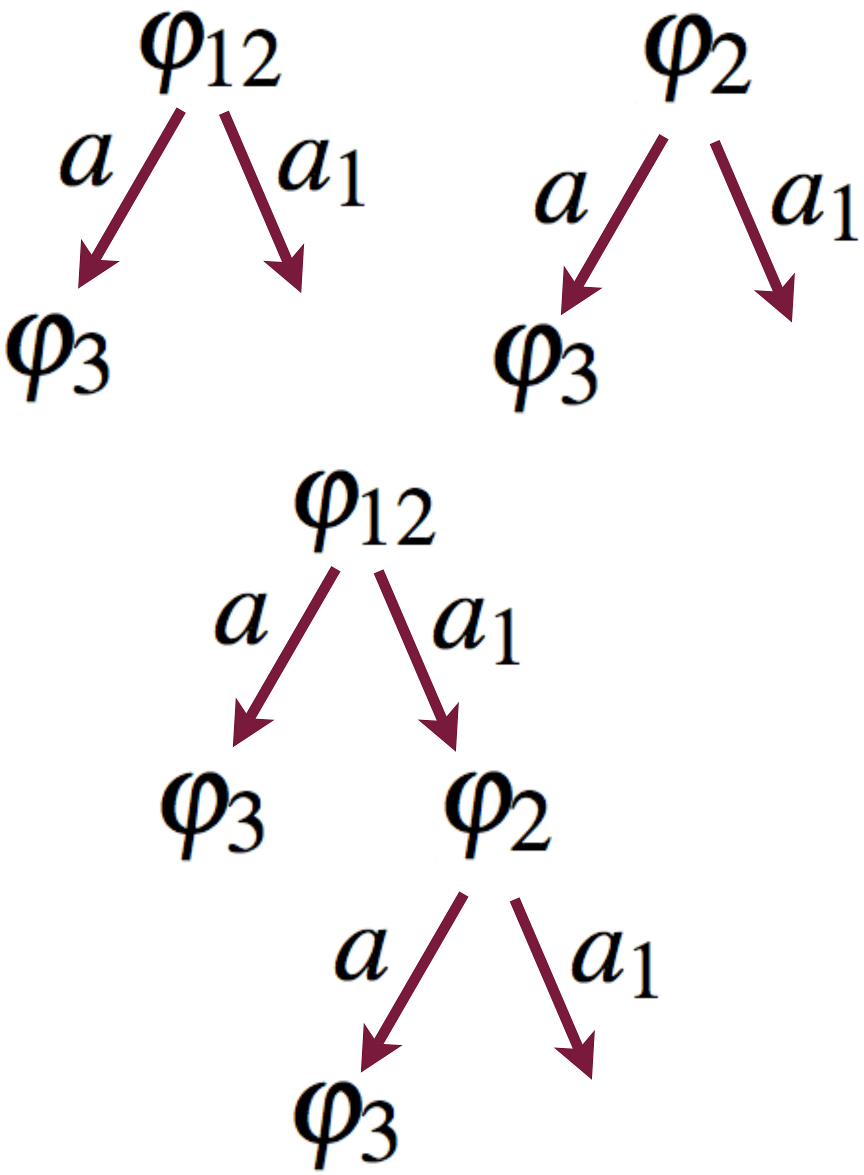
(8) $\Gamma[\varphi_3/a] \,|\, \Sigma \vdash t_1[\varphi_3/a]\ \ \varphi_2[\varphi_3/a]$
$::\ (\varphi_{12}[\varphi_3/a])[\varphi_2[\varphi_3/a]/a_1];\ (\sigma_1[\varphi_3/a])[\varphi_2[\varphi_3/a]/a_1]$ (TyAppT 6 7)

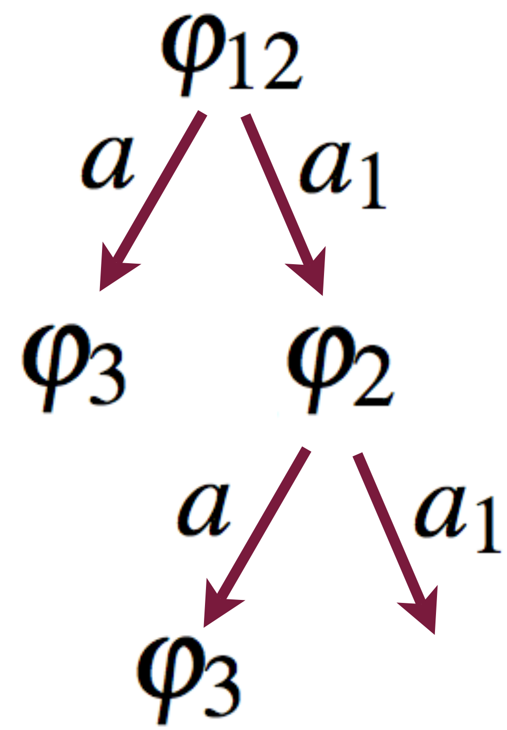
(9) $\Gamma[\varphi_3/a] \,|\, \Sigma \vdash (t_1\ \varphi_2)[\varphi_3/a]$
$::\ (\varphi_{12}[\varphi_2/a_1])[\varphi_3/a];\ (\sigma_1[\varphi_2/a_1])[\varphi_3/a]$ (Def. Sub. 8)

needs an identity for substitution

$$\frac{(3)\ \Gamma,\ a : \kappa_3 \,|\, \Sigma \vdash t_1\ ::\ \forall(a_1 : \kappa_{11}).\, \varphi_{12}\,;\, \sigma_1 \qquad (4)\ \Gamma,\ a : \kappa_3 \,|\, \Sigma \vdash_{\mathrm{T}} \varphi_2\ ::\ \kappa_{11}}{(\underline{1})\ \Gamma,\ a : \kappa_3 \,|\, \Sigma \vdash t_1\ \varphi_2\ ::\ \varphi_{12}[\varphi_2/a_1]\,;\, \sigma_1}$$

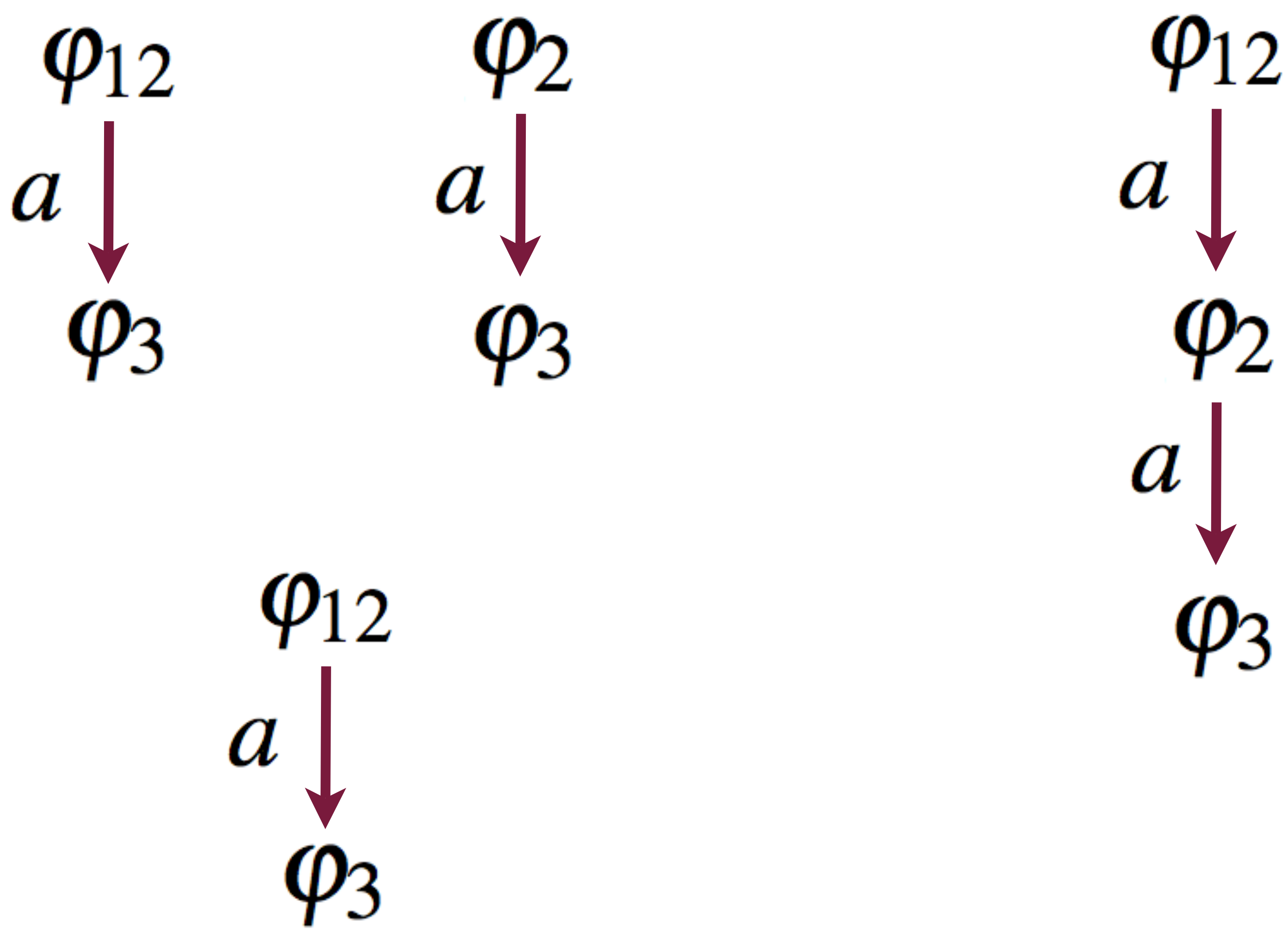
$$(\varphi_{12}[\varphi_3/a])[\varphi_2[\varphi_3/a]/a_1] \ \equiv\ (\varphi_{12}[\varphi_2/a_1])[\varphi_3/a]$$





$$\frac{(3)\ \Gamma,\ a : \kappa_3 \,|\, \Sigma \vdash t_1\ ::\ \forall(a_1 : \kappa_{11}).\, \varphi_{12}\ ;\ \sigma_1 \qquad (4)\ \Gamma,\ a : \kappa_3 \,|\, \Sigma \vdash_{\mathrm{T}} \varphi_2\ ::\ \kappa_{11}}{(\underline{1})\ \Gamma,\ a : \kappa_3 \,|\, \Sigma \vdash t_1\ \varphi_2\ ::\ \varphi_{12}[\varphi_2/a_1]\ ;\ \sigma_1}$$

$$(\varphi_{12}[\varphi_3/a])[\varphi_2[\varphi_3/a]/a\ ]\ \equiv\ (\varphi_{12}[\varphi_2/a\ ])[\varphi_3/a]$$




Score:35 of 158
Sound:on
>_

Another brick in the wall

Lemma 1.9 (Substitution of Types in Values)

$$\begin{array}{ll} \text{If} & \Gamma,\ a : \kappa_2 \,|\, \Sigma \vdash t \ :: \ \tau_1\ ;\ \sigma \\ \text{and} & \Gamma \,|\, \Sigma \vdash_{\text{T}} \varphi_2 \ :: \ \kappa_2 \\ \text{then} & \Gamma[\varphi_2/a] \,|\, \Sigma \vdash t[\varphi_2/a] \ :: \ \tau_1[\varphi_2/a]\ ;\ \sigma[\varphi_2/a] \end{array}$$

(FFS)

... assuming no free variables in φ_2 are bound by t ...

... assuming a is not bound by t ...

The arbitrariness requirement

$$\frac{\forall x.\, P(x)}{P(a)}$$

$$\frac{\Gamma \vdash e :: \forall a.\sigma}{\Gamma \vdash e :: \sigma[a := \tau]}$$

$$\frac{P(a) \qquad (a \text{ arbitrary})}{\forall x.\, P(x)}$$

$$\frac{\Gamma \vdash e :: \sigma \qquad a \notin \mathsf{fv}(\Gamma)}{\Gamma \vdash e :: \forall a.\sigma}$$

Breaching the arbitrariness requirement

- When generalising for a variable, all proofs steps must be possible for all members of the domain.

1	$(\mathrm{Cat}(\mathrm{kitty}) \rightarrow \mathrm{HasFur}(\mathrm{kitty})) \wedge \mathrm{Cat}(\mathrm{kitty})$
2	$\mathrm{Cat}(\mathrm{kitty}) \rightarrow \mathrm{HasFur}(\mathrm{kitty})$
3	$\mathrm{Cat}(\mathrm{kitty})$
4	$\mathrm{HasFur}(\mathrm{kitty})$
5	$\forall x.\ \mathrm{HasFur}(\mathrm{x})$ ***WRONG***

Where do we pull a fresh variable from?

```
(\y. \x. x + y) (x * 2) 5
 => (\z. z + (x * 2)) 5
 => 5 + (x * 2)
```

$$\frac{\Gamma \vdash e :: \sigma \qquad a \notin \mathsf{fv}(\Gamma)}{\Gamma \vdash e :: \forall a.\sigma}$$

deBruijn indices

```
    (\y. \x. x + y) (x * 2) 5
=> (\z. z + (x * 2)) 5
=> 5 + (x * 2)

    (\. \. 0 + 1) (0 * 2) 5
=> (\. 0 + (1 * 2)) 5
=> 5 + (0 * 2)
```

Locally nameless

```
    (\y. \x. x + y) (x * 2) 5
=> (\z. z + (x * 2)) 5
=> 5 + (x * 2)

    (\. \. 0 + 1) (x * 2) 5
=> (\. 0 + (x * 2)) 5
=> 5 + (x * 2)
```

Nominal Reasoning

- If you get into trouble then swap the names around.
- Relies on tool support / proof assistant extensions to generate the various freshness and alpha-conversion lemmas.

```
   (\y. \x. x + y) (x * 2) 5
=> (\x. \y. y + x) (x * 2) 5   SWAP
=> (\y. y + (x * 2)) 5
=> 5 + (x * 2)
```

Higher order abstract syntax

- Shift the problem into the meta-language.
- Works well in Twelf, problems with induction principles in Coq.
- Eliminates need for substitution lemmas, but they you must argue that the HOAS representation is adequate wrt original.

Explicit names

```
data Exp
  = Var Name
  | Lam Name Exp
  | App Exp Exp
```

HOAS

```
data Exp
  = Var Name
  | Lam (Exp -> Exp)
  | App Exp Exp
```

Substitution with names vs deBruijn indices

```
Theorem subst_value_value_names
 :  forall env x val t1 T1 T2
 ,  (forall z, freeX z val -> noBindsX z t1)
 -> TYPE (extend env x T2) t1  T1
 -> TYPE env                val T2
 -> TYPE env  (subst x val t1) T1.


Theorem subst_value_value_debruijn
 :  forall ix tenv t1 t2 T1 T2
 ,  get tenv ix = Some T2
 -> closedX t2
 -> TYPE tenv             t1 T1
 -> TYPE (drop ix tenv) t2 T2
 -> TYPE (drop ix tenv) (subst ix t2 t1) T1.
```

HELLO

my name is

<u>0</u>