Referencias - Motivación

- En una expresión como $let\ x : Nat = \underline{2}\ in\ M$
  - x es una variable declarada con valor 2.
  - El valor de x permanece inalterado a lo largo de la evaluación de M.
  - En este sentido x es inmutable: no existe una operación de asignación.
- En programación imperativa pasa todo lo contrario.
  - Todas las variables son mutables.
- Vamos a extender Cálculo Lambda Tipado con variables mutables.

Operaciones básicas

Alocación (Reserva de memoria)

$ref\ M$ genera una referencia fresca cuyo contenido es el valor de M.

Derreferenciación (Lectura)

$!x$ sigue la referencia x y retorna su contenido.

Asignación

$x := M$ almacena en la referencia x el valor de M.

Ejemplos

Nota: En los ejemplos de esta clase omitiremos los tipos de las let-expresiones para facilitar la lectura.

- $\mathit{let}\ x = \mathit{ref}\ \underline{2}\ \mathit{in}\ !x$ evalúa a

Ejemplos

Nota: En los ejemplos de esta clase omitiremos los tipos de las let-expresiones para facilitar la lectura.

- $\mathit{let}\ x = \mathit{ref}\ \underline{2}\ \mathit{in}\ !x$ evalúa a $\underline{2}$.
- $\mathit{let}\ x = \mathit{ref}\ \underline{2}\ \mathit{in}\ (\lambda_{\_} : \mathit{unit}.!x)\,(x := \mathit{succ}(!x))$ evalúa a

Ejemplos

Nota: En los ejemplos de esta clase omitiremos los tipos de las let-expresiones para facilitar la lectura.

- $let\ x = ref\ \underline{2}\ in\ !x$ evalúa a $\underline{2}$.
- $let\ x = ref\ \underline{2}\ in\ (\lambda\_ : unit.!x)\,(x := succ(!x))$ evalúa a $\underline{3}$.
- $let\ x = \underline{2}\ in\ x$ evalúa a

Ejemplos

Nota: En los ejemplos de esta clase omitiremos los tipos de las let-expresiones para facilitar la lectura.

- $let\ x = ref\ \underline{2}\ in\ !x$ evalúa a $\underline{2}$.
- $let\ x = ref\ \underline{2}\ in\ (\lambda\_ : unit.!x)\,(x := succ(!x))$ evalúa a $\underline{3}$.
- $let\ x = \underline{2}\ in\ x$ evalúa a $\underline{2}$.
- ¿$let\ x = ref\ \underline{2}\ in\ x$ a qué evalúa?

Ejemplos

Nota: En los ejemplos de esta clase omitiremos los tipos de las let-expresiones para facilitar la lectura.

- $let\ x = ref\ \underline{2}\ in\ !x$ evalúa a $\underline{2}$.
- $let\ x = ref\ \underline{2}\ in\ (\lambda\_ : unit.!x)\,(x := succ(!x))$ evalúa a $\underline{3}$.
- $let\ x = \underline{2}\ in\ x$ evalúa a $\underline{2}$.
- ¿$let\ x = ref\ \underline{2}\ in\ x$ a qué evalúa?
- $let\ x = ref\ \underline{2}\ in\ let\ y = x\ in\ (\lambda\_ : unit.!x)\,(y := succ(!y))$ evalúa a $\underline{3}$.
  - x e y son alias para la misma celda de memoria.

Comandos = Expresiones con efectos

- El término $\mathit{let}\ x = \mathit{ref}\ \underline{2}\ \mathit{in}\ x := \mathit{succ}(!x)$, ¿A qué evalúa?

Comandos = Expresiones con efectos

- El término $let\ x = ref\ \underline{2}\ in\ x := succ(!x)$, ¿A qué evalúa?
- La asignación es una expresión que interesa por su efecto y no su valor.
  - No tiene interés preguntarse por el valor de una asignación.
  - ¡Sí tiene sentido preguntarse por el efecto!

Comando

Expresión que se evalúa para causar un efecto; definimos a *unit* como su valor.

- Un lenguaje funcional puro es uno en el que las expresiones son puras en el sentido de carecer de efectos.

Expresiones de tipos

Las expresiones de tipos se extienden del siguiente modo

$$\sigma \quad ::= \quad \mathit{Bool} \mid \mathit{Nat} \mid \sigma \to \tau \mid \mathit{Unit} \mid \mathit{Ref}\ \sigma$$

Descripción informal:

- $\mathit{Ref}\ \sigma$ es el tipo de las referencias a valores de tipo σ.
- Ej. $\mathit{Ref}\ (\mathit{Bool} \to \mathit{Nat})$ es el tipo de las referencias a funciones de Bool en Nat.

Términos

$$
\begin{array}{rcl}
M & ::= & x \\
 & | & \lambda x : \sigma . M \\
 & | & M\,N \\
 & | & unit \\
 & | & ref\ \ M \\
 & | & !M \\
 & | & M := N \\
 & | & \ldots
\end{array}
$$

Reglas de tipado - Preliminares

$$\frac{\Gamma \rhd M_1 : \sigma}{\Gamma \rhd \mathit{ref}\ \ M_1 : \mathit{Ref}\ \ \sigma}\ (\textsc{T-Ref})$$

$$\frac{\Gamma \rhd M_1 : \mathit{Ref}\ \ \sigma}{\Gamma \rhd !M_1 : \sigma}\ (\textsc{T-DeRef})$$

$$\frac{\Gamma \rhd M_1 : \mathit{Ref}\ \ \sigma_1 \quad \Gamma \rhd M_2 : \sigma_1}{\Gamma \rhd M_1 := M_2 : \mathit{Unit}}\ (\textsc{T-Assign})$$

Motivación

Al intentar formalizar la semántica operacional surgen las preguntas:

- ¿Cuáles son los valores de tipo $\mathit{Ref}\ \sigma$?
- ¿Cómo modelar la evaluación del término $\mathit{ref}\ M$?

Las respuestas dependen de otra pregunta.

¿Qué es una referencia?

Motivación

Al intentar formalizar la semántica operacional surgen las preguntas:

- ¿Cuáles son los valores de tipo $Ref\ \sigma$?
- ¿Cómo modelar la evaluación del término $ref\ M$?

Las respuestas dependen de otra pregunta.

¿Qué es una referencia?

Rta. Es una abstracción de una porción de memoria que se encuentra en uso.

Memoria o "store"

- Usamos direcciones (simbólicas) o "locations" $l, l_i \in \mathcal{L}$ para representar referencias.

  Memoria (o "store"): función parcial de direcciones a valores.
- Usamos letras μ, μ' para referirnos a stores.
- Notación:
  - $\mu[l \mapsto V]$ es el store resultante de pisar $\mu(l)$ con V.
  - $\mu \oplus (l \mapsto V)$ es el store extendido resultante de ampliar μ con una nueva asociación $l \mapsto V$ (asumimos $l \notin Dom(\mu)$).

Los juicios de evaluación toman la forma:

$$M \mid \mu \to M' \mid \mu'$$

Valores

Intuición:

$$\frac{l \notin Dom(\mu)}{ref\ \ V \mid \mu \to l \mid \mu \oplus (l \mapsto V)}\ (\text{E-RefV})$$

Los valores posibles ahora incluyen las direcciones.

$$V \quad ::= \quad unit \mid \lambda x : \sigma.M \mid l$$

Dado que los valores son un subconjunto de los términos,

- debemos ampliar los términos con direcciones;
- éstas son producto de la formalización y no se pretende que sean utilizadas por el programador.

Términos extendidos

$$
\begin{array}{rcl}
M & ::= & x \\
 & | & \lambda x : \sigma . M \\
 & | & M\ N \\
 & | & unit \\
 & | & ref\ \ M \\
 & | & !M \\
 & | & M := N \\
 & | & l \\
 & | & \ldots
\end{array}
$$

Juicios de tipado

$$\Gamma \triangleright I : \mathbf{?}$$

Juicios de tipado

$$\Gamma \rhd l : ?$$

- Depende de los valores que se almacenen en la dirección l.
- Situación parecida a las variables libres.
- Precisamos un "contexto de tipado" para direcciones:
  - Σ función parcial de direcciones en tipos.

Nuevo juicio de tipado

$$\Gamma | \Sigma \rhd M : \sigma$$

Reglas de tipado - Definitivas

$$\frac{\Gamma|\Sigma \rhd M_1 : \sigma}{\Gamma|\Sigma \rhd \mathit{ref}\ \ M_1 : \mathit{Ref}\ \ \sigma} \ (\textsc{T-Ref})$$

$$\frac{\Gamma|\Sigma \rhd M_1 : \mathit{Ref}\ \ \sigma}{\Gamma|\Sigma \rhd ! M_1 : \sigma} \ (\textsc{T-DeRef})$$

$$\frac{\Gamma|\Sigma \rhd M_1 : \mathit{Ref}\ \ \sigma_1 \qquad \Gamma|\Sigma \rhd M_2 : \sigma_1}{\Gamma|\Sigma \rhd M_1 := M_2 : \mathit{Unit}} \ (\textsc{T-Assign})$$

$$\frac{\Sigma(l) = \sigma}{\Gamma|\Sigma \rhd l : \mathit{Ref}\ \ \sigma} \ (\textsc{T-Loc})$$

Juicios de evaluación en un paso

- Retomamos la semántica operacional.
- Vamos a introducir axiomas y reglas que permiten darle significado al juicio de evaluación en un paso.

$$M \mid \mu \to M' \mid \mu'$$

- Recordar el conjunto de valores (expresiones resultantes de evaluar por completo a términos cerrados y bien tipados).

$$V \quad ::= \quad true \mid false \mid 0 \mid \underline{n} \mid unit \mid \lambda x : \sigma.M \mid l$$

Juicios de evaluación en un paso

$$\frac{M_1 \mid \mu \to M_1' \mid \mu'}{!M_1 \mid \mu \to !M_1' \mid \mu'} \text{(E-DEREF)}$$

$$\frac{\mu(l) = V}{!l \mid \mu \to V \mid \mu} \text{(E-DEREFLOC)}$$

Juicios de evaluación en un paso

$$\frac{M_1 \mid \mu \to M_1' \mid \mu'}{M_1 := M_2 \mid \mu \to M_1' := M_2 \mid \mu'} \text{(E-Assign1)}$$

$$\frac{M_2 \mid \mu \to M_2' \mid \mu'}{V := M_2 \mid \mu \to V := M_2' \mid \mu'} \text{(E-Assign2)}$$

$$\frac{}{l := V \mid \mu \to unit \mid \mu[l \mapsto V]} \text{(E-Assign)}$$

Juicios de evaluación en un paso

$$\frac{M_1 \mid \mu \to M_1' \mid \mu'}{ref\ \ M_1 \mid \mu \to ref\ \ M_1' \mid \mu'}\ (\text{E-Ref})$$

$$\frac{l \notin Dom(\mu)}{ref\ \ {\color{red}V} \mid \mu \to l \mid \mu \oplus (l \mapsto {\color{red}V})}\ (\text{E-RefV})$$

Revisitar los juicios de evaluación para los restantes términos

$$\frac{M_1 \mid \mu \to M_1' \mid \mu'}{M_1\ M_2 \mid \mu \to M_1'\ M_2 \mid \mu'}\ (\text{E-App1})$$

$$\frac{M_2 \mid \mu \to M_2' \mid \mu'}{\color{red}{V_1}\ M_2 \mid \mu \to \color{red}{V_1}\ M_2' \mid \mu'}\ (\text{E-App2})$$

$$\frac{}{(\lambda x : \sigma.M)\ \color{red}{V} \mid \mu \to M\{x \leftarrow \color{red}{V}\} \mid \mu}\ (\text{E-AppAbs})$$

Las de los restantes tipos (*Nat*, *Bool*, ...) son similares.

Nota: Estas reglas no modifican el store.

Corrección de sistema de tipos

Corrección $=$ Progreso $+$ Preservación

Progreso

Si M es cerrado y bien tipado entonces

1. M es un valor
2. o bien existe M' tal que $M \to M'$

Preservación

Si $\Gamma \rhd M : \sigma$ y $M \to N$, entonces $\Gamma \rhd N : \sigma$

Debemos reformular estos resultados en el marco de referencias.

Preservación - Formulación ingenua

La formulación ingenua siguiente es errónea:

$$\Gamma|\Sigma \triangleright M : \sigma \quad \text{y} \quad M \mid \mu \to M' \mid \mu' \quad \text{implica} \quad \Gamma|\Sigma \triangleright M' : \sigma$$

- El problema: puede que la semántica no respete los tipos asumidos por el sistema de tipos para las direcciones (i.e. Σ).
- Vamos a ver un ejemplo concreto.

Preservación - Formulación ingenua

$$\Gamma | \Sigma \triangleright M : \sigma \quad \text{y} \quad M \mid \mu \to M' \mid \mu' \quad \text{implica} \quad \Gamma | \Sigma \triangleright M' : \sigma$$

Preservación - Formulación ingenua

$$\Gamma|\Sigma \rhd M : \sigma \quad \text{y} \quad M \mid \mu \to M' \mid \mu' \quad \text{implica} \quad \Gamma|\Sigma \rhd M' : \sigma$$

Supongamos que

- $M = !l$
- $\Gamma = \emptyset$
- $\Sigma(l) = Nat$
- $\mu(l) = true$

Observar que

- $\Gamma|\Sigma \rhd M : Nat$ y
- $M \mid \mu \to true \mid \mu$
- pero $\Gamma|\Sigma \rhd true : Nat$ no vale.

Preservación - Formulación ingenua

$$\Gamma|\Sigma \rhd M : \sigma \quad \text{y} \quad M \,|\, \mu \to M' \,|\, \mu' \quad \text{implica} \quad \Gamma|\Sigma \rhd M' : \sigma$$

Supongamos que

- $M = !l$
- $\Gamma = \emptyset$
- $\Sigma(l) = \boxed{Nat}$
- $\mu(l) = \boxed{true}$

Observar que

- $\Gamma|\Sigma \rhd M : Nat$ y
- $M \,|\, \mu \to true \,|\, \mu$
- pero $\Gamma|\Sigma \rhd true : Nat$ no vale.

Preservación - Reformulada

- Precisamos una noción de compatibilidad entre el store y el contexto de tipado para stores.
  - Debemos "tipar" los stores.
- Introducimos un nuevo "juicio de tipado":

$$\Gamma|\Sigma \triangleright \mu$$

- Este juicio se define del siguiente modo:

$$\Gamma|\Sigma \triangleright \mu \text{ sii}$$

  1. $Dom(\Sigma) = Dom(\mu)$ y
  2. $\Gamma|\Sigma \triangleright \mu(l) : \Sigma(l)$ para todo $l \in Dom(\mu)$.

Preservación - Reformulada

Reformulamos preservación del siguiente modo.

$$\text{Si} \quad \Gamma|\Sigma \rhd M : \sigma \quad \text{y} \quad M \,|\, \mu \to N \,|\, \mu' \quad \text{y} \quad \Gamma|\Sigma \rhd \mu,$$

$$\text{entonces} \quad \Gamma|\boxed{\Sigma} \rhd N : \sigma.$$

- Esto es casi correcto.

Preservación - Reformulada

Reformulamos preservación del siguiente modo.

$$\text{Si} \quad \Gamma|\Sigma \triangleright M : \sigma \quad \text{y} \quad M \,|\, \mu \to N \,|\, \mu' \quad \text{y} \quad \Gamma|\Sigma \triangleright \mu,$$

$$\text{entonces} \quad \Gamma|\boxed{\Sigma} \triangleright N : \sigma.$$

- Esto es casi correcto.
- No contempla la posibilidad de que el Σ encuadrado haya crecido en dominio respecto a Σ.
  - Por posibles reservas de memoria.

Preservación - Definitiva

Si

- $\Gamma|\Sigma \triangleright M : \sigma$
- $M \mid \mu \to N \mid \mu'$
- $\Gamma|\Sigma \triangleright \mu$

implica que existe $\Sigma' \supseteq \Sigma$ tal que

- $\Gamma|\Sigma' \triangleright N : \sigma$
- $\Gamma|\Sigma' \triangleright \mu'$

Progreso - Reformulado

Si M es cerrado y bien tipado (i.e. $\emptyset|\Sigma \triangleright M : \sigma$ para algún Σ, σ) entonces:

1. M es un valor
2. o bien para cualquier store μ tal que $\emptyset|\Sigma \triangleright \mu$, existe M' y μ' tal que $M \mid \mu \to M' \mid \mu'$.

Ejemplo

$$let\ x = ref\ \underline{2}\ in\ (\lambda\_ : Unit.!x)\,(x := succ(!x))\ |\ \mu$$

$$\rightarrow$$

Ejemplo

$$
\begin{array}{rl}
& let\ x = ref\ \underline{2}\ in\ (\lambda\_ : Unit.!x)\,(x := succ(!x)) \mid \mu \\
\to & let\ x = l_1\ in\ (\lambda\_ : Unit.!x)\,(x := succ(!x)) \mid \mu \oplus (l_1 \mapsto \underline{2}) \\
\to &
\end{array}
$$

Ejemplo

$$
\begin{array}{rl}
& let\ x = ref\ \underline{2}\ in\ (\lambda\_ : Unit.!x)\,(x := succ(!x)) \mid \mu \\
\rightarrow & let\ x = l_1\ in\ (\lambda\_ : Unit.!x)\,(x := succ(!x)) \mid \mu \oplus (l_1 \mapsto \underline{2}) \\
\rightarrow & (\lambda\_ : Unit.!l_1)\,(l_1 := succ(!l_1)) \mid \mu \oplus (l_1 \mapsto \underline{2}) \\
\rightarrow &
\end{array}
$$

Ejemplo

$$
\begin{array}{ll}
& let\ x = ref\ \underline{2}\ in\ (\lambda\_ : Unit.!x)\,(x := succ(!x)) \mid \mu \\
\to & let\ x = l_1\ in\ (\lambda\_ : Unit.!x)\,(x := succ(!x)) \mid \mu \oplus (l_1 \mapsto \underline{2}) \\
\to & (\lambda\_ : Unit.!l_1)\,(l_1 := succ(!l_1)) \mid \mu \oplus (l_1 \mapsto \underline{2}) \\
\to & (\lambda\_ : Unit.!l_1)\,(l_1 := succ(\underline{2})) \mid \mu \oplus (l_1 \mapsto \underline{2}) \\
\to &
\end{array}
$$

Ejemplo

$$
\begin{array}{ll}
 & let\ x = ref\ \underline{2}\ in\ (\lambda\_ : Unit.!x)\,(x := succ(!x)) \mid \mu \\
\to & let\ x = l_1\ in\ (\lambda\_ : Unit.!x)\,(x := succ(!x)) \mid \mu \oplus (l_1 \mapsto \underline{2}) \\
\to & (\lambda\_ : Unit.!l_1)\,(l_1 := succ(!l_1)) \mid \mu \oplus (l_1 \mapsto \underline{2}) \\
\to & (\lambda\_ : Unit.!l_1)\,(l_1 := succ(\underline{2})) \mid \mu \oplus (l_1 \mapsto \underline{2}) \\
\to & (\lambda\_ : Unit.!l_1)\ unit \mid (\mu \oplus (l_1 \mapsto \underline{2}))[l_1 \mapsto \underline{3}] \\
\to &
\end{array}
$$

Ejemplo

$$
\begin{array}{ll}
 & let\ x = ref\ \underline{2}\ in\ (\lambda_- : Unit.!x)\,(x := succ(!x)) \mid \mu \\
\to & let\ x = l_1\ in\ (\lambda_- : Unit.!x)\,(x := succ(!x)) \mid \mu \oplus (l_1 \mapsto \underline{2}) \\
\to & (\lambda_- : Unit.!l_1)\,(l_1 := succ(!l_1)) \mid \mu \oplus (l_1 \mapsto \underline{2}) \\
\to & (\lambda_- : Unit.!l_1)\,(l_1 := succ(\underline{2})) \mid \mu \oplus (l_1 \mapsto \underline{2}) \\
\to & (\lambda_- : Unit.!l_1)\ unit \mid (\mu \oplus (l_1 \mapsto \underline{2}))[l_1 \mapsto \underline{3}] \\
\to & !l_1 \mid \mu \oplus (l_1 \mapsto \underline{3}) \\
\to & \underline{3} \mid \mu \oplus (l_1 \mapsto \underline{3})
\end{array}
$$

Ejemplo

Sea

$$\begin{array}{rcl} M & = & \lambda r : Ref(Unit \to Unit). \\ & & \quad let\ f\ = !r \\ & & \quad\ in\ (r := \lambda x : Unit.f\ x); (!r)\ unit \end{array}$$

$$\begin{array}{ll} & M\,(ref\ \ (\lambda x : Unit.x)) \mid \mu \\ \to & \end{array}$$

Ejemplo

Sea

$$\begin{array}{rcl} M & = & \lambda r : Ref(Unit \to Unit). \\ & & \quad let\ f = !r \\ & & \quad\ in\ (r := \lambda x : Unit.f\ x); (!r)\ unit \end{array}$$

$$\begin{array}{ll} & M\,(ref\ (\lambda x : Unit.x)) \mid \mu \\ \to & M\, l_1 \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.x) \\ \to & \end{array}$$

Ejemplo

Sea

$$\begin{array}{rcl} M & = & \lambda r : Ref(Unit \to Unit). \\ & & \quad let\ f = !r \\ & & \quad\ in\ (r := \lambda x : Unit.f\ x); (!r)\ unit \end{array}$$

$$\begin{array}{ll} & M\,(ref\ \ (\lambda x : Unit.x)) \mid \mu \\ \to & M\ l_1 \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.x) \\ \to & let\ f = !l_1\ in\ (l_1 := \lambda x : Unit.f\ x); (!l_1)\ unit \mid \ldots \\ \to & \end{array}$$

Ejemplo

Sea

$$
\begin{array}{rcl}
M & = & \lambda r : Ref(Unit \to Unit). \\
 & & \quad let\ f = !r \\
 & & \quad\ in\ (r := \lambda x : Unit.f\ x); (!r)\ unit
\end{array}
$$

$$
\begin{array}{ll}
 & M\,(ref\ (\lambda x : Unit.x)) \mid \mu \\
\to & M\,l_1 \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.x) \\
\to & let\ f = !l_1\ in\ (l_1 := \lambda x : Unit.f\ x); (!l_1)\ unit \mid \ldots \\
\to & let\ f = \lambda x : Unit.x\ in\ (l_1 := \lambda x : Unit.f\ x); (!l_1)\ unit \mid \ldots \\
\to &
\end{array}
$$

Ejemplo

Sea

$$\begin{array}{rcl} M & = & \lambda r : Ref(Unit \to Unit). \\ & & \quad let\ f = !r \\ & & \quad\ in\ (r := \lambda x : Unit.f\ x); (!r)\ unit \end{array}$$

$$\begin{array}{ll} & M\,(ref\ (\lambda x : Unit.x)) \mid \mu \\ \to & M\, l_1 \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.x) \\ \to & let\ f = !l_1\ in\ (l_1 := \lambda x : Unit.f\ x); (!l_1)\ unit \mid \ldots \\ \to & let\ f = \lambda x : Unit.x\ in\ (l_1 := \lambda x : Unit.f\ x); (!l_1)\ unit \mid \ldots \\ \to & (l_1 := \lambda x : Unit.(\lambda x : Unit.x)\ x); (!l_1)\ unit \mid \ldots \\ \to & \end{array}$$

Ejemplo

Sea

$$M \;=\; \lambda r : Ref(Unit \to Unit).\; let\ f = !r\ in\ (r := \lambda x : Unit.f\ x); (!r)\ unit$$

$$\begin{array}{ll}
 & M\,(ref\ (\lambda x : Unit.x)) \mid \mu \\
\to & M\, l_1 \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.x) \\
\to & let\ f = !l_1\ in\ (l_1 := \lambda x : Unit.f\ x); (!l_1)\ unit \mid \ldots \\
\to & let\ f = \lambda x : Unit.x\ in\ (l_1 := \lambda x : Unit.f\ x); (!l_1)\ unit \mid \ldots \\
\to & (l_1 := \lambda x : Unit.(\lambda x : Unit.x)\ x); (!l_1)\ unit \mid \ldots \\
\to & unit; (!l_1)\ unit \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.(\lambda x : Unit.x)\ x) \\
\to &
\end{array}$$

Ejemplo

Sea

$$M \;=\; \lambda r : Ref(Unit \to Unit).\ let\ f = !r\ in\ (r := \lambda x : Unit.f\ x); (!r)\ unit$$

$$\begin{array}{ll}
& M\,(ref\ (\lambda x : Unit.x)) \mid \mu \\
\to & M\,l_1 \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.x) \\
\to & let\ f = !l_1\ in\ (l_1 := \lambda x : Unit.f\ x); (!l_1)\ unit \mid \ldots \\
\to & let\ f = \lambda x : Unit.x\ in\ (l_1 := \lambda x : Unit.f\ x); (!l_1)\ unit \mid \ldots \\
\to & (l_1 := \lambda x : Unit.(\lambda x : Unit.x)\,x); (!l_1)\ unit \mid \ldots \\
\to & unit; (!l_1)\ unit \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.(\lambda x : Unit.x)\,x) \\
\to & (!l_1)\ unit \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.(\lambda x : Unit.x)\,x) \\
\to &
\end{array}$$

Ejemplo

Sea

$$
\begin{array}{rcl}
M & = & \lambda r : Ref(Unit \to Unit). \\
 & & \quad let\ f = !r \\
 & & \quad\ in\ (r := \lambda x : Unit.f\ x); (!r)\ unit
\end{array}
$$

$$
\begin{array}{ll}
 & M\,(ref\ (\lambda x : Unit.x)) \mid \mu \\
\to & M\ l_1 \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.x) \\
\to & let\ f = !l_1\ in\ (l_1 := \lambda x : Unit.f\ x); (!l_1)\ unit \mid \ldots \\
\to & let\ f = \lambda x : Unit.x\ in\ (l_1 := \lambda x : Unit.f\ x); (!l_1)\ unit \mid \ldots \\
\to & (l_1 := \lambda x : Unit.(\lambda x : Unit.x)\ x); (!l_1)\ unit \mid \ldots \\
\to & unit; (!l_1)\ unit \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.(\lambda x : Unit.x)\ x) \\
\to & (!l_1)\ unit \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.(\lambda x : Unit.x)\ x) \\
\to & (\lambda x : Unit.(\lambda x : Unit.x)\ x)\ unit \mid \ldots \\
\to &
\end{array}
$$

Ejemplo

Sea

$$
\begin{array}{rcl}
M & = & \lambda r : Ref(Unit \to Unit). \\
 & & \quad let\ f = !r \\
 & & \quad\ in\ (r := \lambda x : Unit.f\ x); (!r)\ unit
\end{array}
$$

$$
\begin{array}{rl}
 & M\,(ref\ (\lambda x : Unit.x)) \mid \mu \\
\to & M\ l_1 \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.x) \\
\to & let\ f = !l_1\ in\ (l_1 := \lambda x : Unit.f\ x); (!l_1)\ unit \mid \ldots \\
\to & let\ f = \lambda x : Unit.x\ in\ (l_1 := \lambda x : Unit.f\ x); (!l_1)\ unit \mid \ldots \\
\to & (l_1 := \lambda x : Unit.(\lambda x : Unit.x)\ x); (!l_1)\ unit \mid \ldots \\
\to & unit; (!l_1)\ unit \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.(\lambda x : Unit.x)\ x) \\
\to & (!l_1)\ unit \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.(\lambda x : Unit.x)\ x) \\
\to & (\lambda x : Unit.(\lambda x : Unit.x)\ x)\ unit \mid \ldots \\
\to & (\lambda x : Unit.x)\ unit \mid \ldots \\
\to & unit \mid \ldots
\end{array}
$$

Ejemplo

Sea

$$\begin{array}{rcl} M & = & \lambda r : Ref(Unit \to Unit). \\ & & \quad \color{red}{let\ f = !r} \\ & & \quad\ in\ (r := \lambda x : Unit.\color{red}{f}\ x); (!r)\ unit \end{array}$$

Reemplazamos f por $!r$ y nos queda

$$\begin{array}{rcl} M' & = & \lambda r : Ref\ (Unit \to Unit). \\ & & \quad (r := \lambda x : Unit.(!r)\ x); (!r)\ unit \end{array}$$

Vamos a evaluar este nuevo M' aplicado al mismo término que en el slide anterior y ver qué pasa...

Ejemplo

$$
\begin{array}{rcl}
M' & = & \lambda r : Ref\ (Unit \to Unit). \\
 & & \quad (r := \lambda x : Unit.(!r)\,x); (!r)\,unit
\end{array}
$$

$$M'\,(ref\ (\lambda x : Unit.x)) \mid \mu$$

$$\to$$

Ejemplo

$$\begin{array}{rcl} M' & = & \lambda r : Ref\ (Unit \to Unit). \\ & & \quad (r := \lambda x : Unit.(!r)\, x); (!r)\, unit \end{array}$$

$$\begin{array}{rl} & M'\, (ref\ (\lambda x : Unit.x)) \mid \mu \\ \to & M'\, l_1 \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.x) \\ \to & \end{array}$$

Ejemplo

$$
\begin{aligned}
M' \quad = \quad & \lambda r : Ref\ (Unit \to Unit). \\
& (r := \lambda x : Unit.(!r)\, x); (!r)\, unit
\end{aligned}
$$

$$
\begin{aligned}
& M'\, (ref\ (\lambda x : Unit.x)) \mid \mu \\
\to \quad & M'\, l_1 \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.x) \\
\to \quad & (l_1 := \lambda x : Unit.(!l_1)\, x); (!l_1)\, unit \mid \ldots \\
\to \quad &
\end{aligned}
$$

Ejemplo

$$
\begin{array}{rcl}
M' & = & \lambda r : Ref\ (Unit \to Unit). \\
 & & \quad (r := \lambda x : Unit.(!r)\, x); (!r)\, unit
\end{array}
$$

$$
\begin{array}{rl}
 & M'\, (ref\ (\lambda x : Unit.x)) \mid \mu \\
\to & M'\, l_1 \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.x) \\
\to & (l_1 := \lambda x : Unit.(!l_1)\, x); (!l_1)\, unit \mid \ldots \\
\to & unit; (!l_1)\, unit \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.(!l_1)\, x) \\
\to &
\end{array}
$$

Ejemplo

$$
\begin{array}{rcl}
M' & = & \lambda r : \mathit{Ref}\ (\mathit{Unit} \to \mathit{Unit}). \\
& & \quad (r := \lambda x : \mathit{Unit}.(!r)\, x); (!r)\, \mathit{unit}
\end{array}
$$

$$
\begin{array}{rl}
& M'\, (\mathit{ref}\ (\lambda x : \mathit{Unit}.x)) \mid \mu \\
\to & M'\, l_1 \mid \mu \oplus (l_1 \mapsto \lambda x : \mathit{Unit}.x) \\
\to & (l_1 := \lambda x : \mathit{Unit}.(!l_1)\, x); (!l_1)\, \mathit{unit} \mid \ldots \\
\to & \mathit{unit}; (!l_1)\, \mathit{unit} \mid \mu \oplus (l_1 \mapsto \lambda x : \mathit{Unit}.(!l_1)\, x) \\
\to & \boxed{(!l_1)\, \mathit{unit}} \mid \ldots \\
\to &
\end{array}
$$

Ejemplo

$$
\begin{array}{rcl}
M' & = & \lambda r : Ref\ (Unit \to Unit). \\
 & & \quad (r := \lambda x : Unit.(!r)\, x); (!r)\, unit
\end{array}
$$

$$
\begin{array}{rl}
 & M'\, (ref\ (\lambda x : Unit.x)) \mid \mu \\
\to & M'\, l_1 \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.x) \\
\to & (l_1 := \lambda x : Unit.(!l_1)\, x); (!l_1)\, unit \mid \ldots \\
\to & unit; (!l_1)\, unit \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.(!l_1)\, x) \\
\to & \boxed{(!l_1)\, unit} \mid \ldots \\
\to & (\lambda x : Unit.(!l_1)\, x)\, unit \mid \ldots \\
\to &
\end{array}
$$

Ejemplo

$$
\begin{array}{rcl}
M' & = & \lambda r : Ref\ (Unit \to Unit). \\
 & & \quad (r := \lambda x : Unit.(!r)\, x); (!r)\, unit
\end{array}
$$

$$
\begin{array}{ll}
 & M'\, (ref\ (\lambda x : Unit.x)) \mid \mu \\
\to & M'\, l_1 \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.x) \\
\to & (l_1 := \lambda x : Unit.(!l_1)\, x); (!l_1)\, unit \mid \ldots \\
\to & unit; (!l_1)\, unit \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.(!l_1)\, x) \\
\to & \boxed{(!l_1)\, unit} \mid \ldots \\
\to & (\lambda x : Unit.(!l_1)\, x)\, unit \mid \ldots \\
\to & \boxed{(!l_1)\, unit} \mid \ldots \\
\to & \ldots
\end{array}
$$

Ejemplo

$$M' \;=\; \lambda r : Ref\ (Unit \to Unit).\ (r := \lambda x : Unit.(!r)\, x); (!r)\, unit$$

$$\begin{array}{ll} & M'\,(ref\ (\lambda x : Unit.x)) \mid \mu \\ \to & M'\, l_1 \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.x) \\ \to & (l_1 := \lambda x : Unit.(!l_1)\, x); (!l_1)\, unit \mid \ldots \\ \to & unit; (!l_1)\, unit \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.(!l_1)\, x) \\ \to & \boxed{(!l_1)\, unit} \mid \ldots \\ \to & (\lambda x : Unit.(!l_1)\, x)\, unit \mid \ldots \\ \to & \boxed{(!l_1)\, unit} \mid \ldots \\ \to & \ldots \end{array}$$

Nota: no todo término cerrado y bien tipado termina en λ^{bnr} (λ-cálculo con booleanos, naturales y referencias).