El CL puro sólo contiene variables, aplicaciones y la notación lambda (llamada abstracción).

$\langle exp \rangle$	$::=$		expresiones o términos
		$\langle var \rangle$	variables
		$\mid \langle exp \rangle \, \langle exp \rangle$	aplicación
		$\mid \lambda \, \langle var \rangle \, . \; \langle exp \rangle$	abstracción o expresión lambda

Convención: La aplicación asocia a izquierda. En $\lambda v.e$, la primer ocurrencia de v es ligadora y su alcance es e. Por ejemplo,

$\lambda x.(\lambda y.xyx)x$ es lo mismo que

$\lambda x.(\lambda y.(xy)x)x$

Variables libres:

$$\begin{aligned} FV(v) &= \{v\} \\ FV(ee') &= FV(e) \cup FV(e') \\ FV(\lambda v.e) &= FV(e) \; - \; \{v\} \end{aligned}$$

Conjunto de sustituciones: $\Delta = \langle var \rangle \to \langle exp \rangle$

Operador sustitución: $\_/\_ \in \langle exp \rangle \times \Delta \to \langle exp \rangle$

$$
\begin{aligned}
v/\delta &= \delta v \\
(ee')/\delta &= (e/\delta)(e'/\delta) \\
(\lambda v.e)/\delta &= \lambda v_{new}.\ e/[\delta | v : v_{new}] \\
&\quad \text{donde } v_{new} \notin \bigcup_{w \in FV(e)-\{v\}} FV(\delta\ w)
\end{aligned}
$$

Renombre: Cambio en $\lambda v.e$ de la variable ligada v (y todas sus ocurrencias) por una variable v' que no ocurra libre en e: $\lambda v'.\ e/v \mapsto v'$ donde $v' \notin FV(e)$.

α-**conversión**: Si e_1 se obtiene a partir de e_0 por 0 o más renombres de ocurrencias de subfrases. También se dice que e_0 α-convierte a e_1.

Notación para expresiones α- **convertibles**: $e_0 \equiv e_1$

Redex: Es una expresión de la forma $(\lambda v.e)e'$

Contracción β: Reemplaza en e_0 una ocurrencia de un redex $(\lambda v.e)e'$ por su contracción $(e/v \mapsto e')$, y luego efectúa cero o más renombres de cualquier subexpresión.

Notación: Si e_1 es el resultado de una contracción β de e_0, entonces escribimos

$$e_0 \to e_1$$

Forma normal: expresión sin redices. Las formas normales representan configuraciones terminales. Por eso la semántica operacional del cálculo lambda consiste en efectuar contracciones β hasta obtener formas normales.

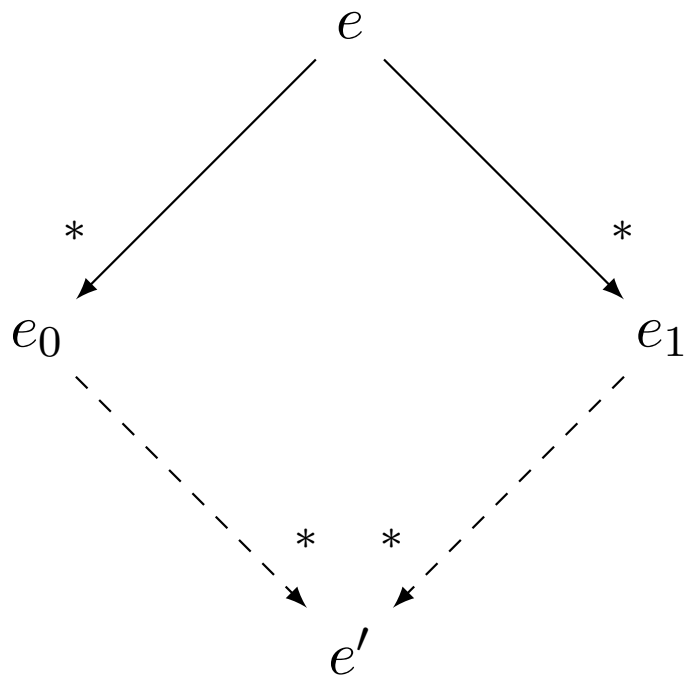
$\to^*$ denota la clausura transitiva y refexiva de $\to$
(o sea, aplicar $\to$ cero o más veces)

Formalmente:
$e \to^* e'$ si y sólo si existen $e_0, ..., e_n$ (con $n \geq 0$) tales que:

$$e = e_0 \to e_1 \to ... \to e_n = e'$$

Notar que si $n = 0$ entonces $e = e'$

Teorema de Church-Rosser Si $e \to^* e_0$ y $e \to^* e_1$, entonces existe e' tal que $e_0 \to^* e'$ y $e_1 \to^* e'$.



Corolario 1. Salvo renombre, toda expresión tiene a lo sumo una forma normal.

Regla η: Un η-redex es una expresión de la forma $\lambda v.ev$, donde $v \notin FV\ e$

$$\frac{}{\lambda v.e\, v \to e} \text{ si } v \notin FV\ e \qquad (\eta)$$

La idea de ejecución (llamada evaluación) que se implementa habitualmente tiene las siguientes diferencias con la relación

- sólo se evalúan expresiones cerradas (es decir, sin variables libres)
- es determinística,
- no busca formas normales sino formas canónicas.

Evaluación (en orden) normal: lenguajes funcionales lazy (Haskell)

Evaluación eager o estricta: lenguajes estrictos (ML).

La noción de forma canónica depende de la definición de evaluación. Se define una noción de forma canónica para la evaluación normal, y otra para la evaluación eager. En el caso del cálculo lambda coinciden: **son las abstracciones**

Propiedad: Una aplicación cerrada no puede ser forma normal.

Corolario: Una expresión cerrada que es forma normal es también forma canónica.

Semántica natural o big-step: En este tipo de semántica, uno no describe un paso de ejecución, sino directamente una relación entre los términos y sus valores (que también son términos, son formas canónicas).Llamaremos $\Rightarrow$ a esta relación.

Reglas para $\Rightarrow_N$

Regla para las formas canónicas

$$\frac{}{\lambda v.e \;\Rightarrow_N\; \lambda v.e}$$

Regla para la aplicación

$$\frac{e \;\Rightarrow_N\; \lambda v.e_0 \qquad (e_0/v \mapsto e') \;\Rightarrow_N\; z}{ee' \;\Rightarrow_N\; z}$$

Reglas para $\Rightarrow_E$

Regla para las formas canónicas

$$\frac{}{\lambda v.e \;\Rightarrow_E\; \lambda v.e}$$

Regla para la aplicación

$$\frac{e \;\Rightarrow_E\; \lambda v.e_0 \qquad e' \;\Rightarrow_E\; z' \qquad (e_0/v \mapsto z') \;\Rightarrow_E\; z}{ee' \;\Rightarrow_E\; z}$$

Asumimos la existencia de un dominio D_∞, junto con un isomorfismo:

$\phi \in D_\infty \to [D_\infty \to D_\infty] \quad \psi \in [D_\infty \to D_\infty] \to D_\infty$

$\phi \circ \psi = Id_{[D_\infty \to D_\infty]}$ y $\psi \circ \phi = Id_{D_\infty}$

Ambientes (Entornos): $\eta \in Env = \langle var \rangle \to D_\infty$

Función semántica: $[\![\_]\!] \in \quad \langle exp \rangle \to Env \to D_\infty$

Ecuaciones semánticas:

$$[\![v]\!]\eta = \eta v$$

$$[\![e_0 e_1]\!]\eta = \phi([\![e_0]\!]\eta)\ [\![e_1]\!]\eta$$

$$[\![\lambda v.e]\!]\eta = \psi(\lambda d \in D_\infty.[\![e]\!][\eta | v : d])$$

Se puede probar que $[\![\Delta\ \Delta]\!]\eta = \bot$.

Teorema de Coincidencia: Si $\eta w = \eta' w$ para todo $w \in FV\ e$, entonces $[\![e]\!]\eta = [\![e]\!]\eta'$.

Teorema de Renombre: Si $v_{new} \notin FV\ e - \{v\}$, entonces $[\![\lambda v_{new}.(e/v \mapsto v_{new})]\!] = [\![\lambda v.e]\!]$.

Sustituciones: $\Delta \ = \quad \langle var \rangle \to \langle exp \rangle$

Teorema de Sustitución: Si $[\![\delta w]\!]\eta = \eta' w$ para todo $w \in FV\ e$, entonces $[\![e/\delta]\!]\eta = [\![e]\!]\eta'$.

Propiedad 3. *(correctitud de la regla β):* $[\![(\lambda v.e)e']\!]\eta = [\![e/v \mapsto e']\!]\eta$

Propiedad 4. *(correctitud de la regla η):* $[\![\lambda v.e\, v]\!]\eta = [\![e]\!]\eta,$ si $v \notin FVe$

Corolario: Si $e \to^* e'$, entonces $[\![e]\!] = [\![e']\!]$.

Semántica Denotacional Normal

$D = V_\perp$, donde $V \approx [D \to D]$

$$\phi \in V \to [D \to D] \qquad \phi \circ \psi = Id_{D\to D}$$
$$\psi \in [D \to D] \to V \qquad \psi \circ \phi = Id_V$$

Notación: $\iota_\perp \in V \to D$

Dominio Semántico: $D = V_\perp \qquad V \approx [D \to D]$

Ambientes: $Env = \langle var \rangle \to D$

Función semántica: $[\![\_]\!] \in \langle exp \rangle \to Env \to D$

Ecuaciones semánticas:

$$\begin{aligned}
[\![v]\!]\eta &= \eta v \\
[\![e_0 e_1]\!]\eta &= \phi_{\perp\!\!\perp}([\![e_0]\!]\eta)([\![e_1]\!]\eta) \\
[\![\lambda v.e]\!]\eta &= \ \iota_\perp \circ \psi\ (\lambda d \in D.[\![e]\!][\eta|v:d])
\end{aligned}$$

Vale la regla β, que utiliza la igualdad:

$$\phi_{\perp\!\!\perp} \circ (\iota_\perp \circ \psi) = Id_{D \to D}$$

Semántica Denotacional Eager

Dominio Semántico: $D \ = V_\perp \qquad V \ \approx \ [V \to D]$

$$\begin{aligned}
\phi \in & V \to [V \to D] & \qquad \phi\circ \ &= Id_{V \to D} \\
\psi \in & [V \to D] \to V & \qquad \psi \circ \phi &= Id_V
\end{aligned}$$

Notación: $\iota_\perp \in V \to D$ **Ambientes:** $Env = \langle var \rangle \to V$

Función semántica: $[\![\_]\!] \in \ \ \langle exp \rangle \to Env \to D$

Ecuaciones semánticas:

$$\begin{aligned}
[\![v]\!]\eta &= \iota_\perp(\eta v) \\
[\![e_0 e_1]\!]\eta &= (\phi_{\perp\!\!\perp}([\![e_0]\!]\eta))_{\perp\!\!\perp}([\![e_1]\!]\eta) \\
[\![\lambda v.e]\!]\eta &= \ \iota_\perp \circ \psi\ (\lambda z \in V.[\![e]\!][\eta|v:z])
\end{aligned}$$

Ya no vale la regla β, (para contra-ejemplo alcanza un $\perp$):

$$[\![(\lambda x\, y.y)\,(\Delta\,\Delta)]\!]\eta = (\lambda z \in V.[\![e]\!][\eta|v:z])_{\perp\!\!\perp}\perp = \perp$$

pero

$$[\![(\lambda y.y)/x \mapsto (\Delta\,\Delta)]\!]\eta = [\![\lambda y.y]\!]\eta \neq \perp$$

Puesto que queremos modelar la evaluación eager, deberíamos esperar que $e \Rightarrow_E z$ implique $[\![e]\!]\eta = [\![z]\!]\eta$.

Lo podemos probar por inducción en la derivación $e \Rightarrow_E z$.