ALP

Autor: Agustín Fernández Bergé

ALP

- Either en lambda calculo
- Ejercicio 2
- Parcial 1 desconocido
- Parcial desconocido
- Practica 2
- Practica 3
- Practica 4

Either en lambda calculo

Either en lambda calculo

$\text{either}(\text{left}\, a) =_\beta \lambda f\ g.\ f\ a$
$\text{either}(\text{right}\, b) =_\beta \lambda f\ g.\ g\ b$

$\text{either} \equiv \lambda x.\ x$
$\text{left} \equiv \lambda x.\ \lambda f\ g.\ f\ x$
$\text{left} \equiv \lambda x.\ \lambda f\ g.\ g\ x$

Ejercicio 2

Ejercicio 2

```
unZipF zs = fold zs (\(x,y) (xs,ys) -> (cons x xs, cons y ys)) nil
map f zs = fold zs (\x y -> cons (f x) y) nil
unZipF zs = pair (map fst zs) (map snd zs)
```

$\text{map} :: \forall X. \forall Y. (X \to Y) \to \text{List}\, X \to \text{List}\, Y$
$\text{map} = \Lambda X. \Lambda Y. \lambda f : X \to Y. \lambda xs : \text{List}\, X.\ xs \langle \text{List}\, Y \rangle\ (\lambda x : X. \lambda y : \text{List}\, Y.\ \text{cons} \langle Y \rangle\ (f\ x)\ y)\ \ \text{nil} \langle Y \rangle$

$\text{unZipF} :: \forall X. \forall Y. \text{List}(\text{Pair}\ X\ Y) \to \text{Pair}(\text{List}\ X)(\text{List Y})$
$\text{unZipF} = \Lambda X. \Lambda Y. \lambda zs : \text{List}(\text{Pair}\ X\ Y).\ \text{pair} \langle X \rangle \langle Y \rangle\ (\text{map} \langle \text{pair X Y} \rangle \langle X \rangle\, \text{fst} \langle X \rangle\ zs)\ (\text{map} \langle \text{pair X Y} \rangle \langle X \rangle\, \text{snd} \langle X \rangle\ zs)$

Parcial 1 desconocido

Ejercicio 1

Ejercicio 1

> **Ejercicio 1**
> c)
> Quiero probar que: $\rho \vdash v \ \wedge \ \rho \vdash v' \implies v \equiv v'$
>
> Pruebo por inducción en la derivación
>
> CB) Num o Var
> No se pueden aplicar mas reglas por lo que vale la propiedad

---

Parcial desconocido

Ejercicio

Ejercicio

> **Ejercicio**
> $\text{suma} \equiv \lambda n\ m.\, \text{foldn}\, n\, \text{Succ}\, m$
>
> $\text{resta} \equiv \lambda n\ m.\, \text{foldn}\, m\, \text{pred}\, n$
> Donde
> $\text{pred} \equiv \lambda n.\, \text{foldn}\ \ n\ (\lambda p, \text{pair}(\text{snd}\, p)(\text{Succ}(\text{snd p}))(\text{pair 0 0})$
>
> $\text{prod} \equiv \lambda n\ m.\, \text{foldn}\, n(\text{suma}\, n)m$
>
> $\text{gt} \equiv \lambda n\ m.\, \text{isNotZero}(\text{resta}\ \ n\ m)$
> $\text{lte} \equiv \text{not}\ \ isZero$
>
> $\text{resto}\, n\ m =_\beta \text{if}(\text{lte}\, n\ m)\, \text{then}\, n\, \text{else}(\text{Succ}(\text{resto}(\text{resta n m})\, m))$
> $\text{resto} \equiv \mathbf{Y}B$
> Donde
> $B \equiv \lambda f.\, \lambda n\ m.\, \text{if}(\text{lte}\, n\ m)\, \text{then}\, n\, \text{else}(\text{Succ}(\text{f}(\text{resta n m})\, m))$
>
> ---
>
> $\text{fibo}'\, n =_\beta \text{if (isZero}\, n)\, \text{then 0 (if (isZero(pred}\, n))\, \text{then 1 else fibo}'(\text{pred n}) + \text{fibo}'(\text{pred(pred n)}))$
> $\text{fibo}\, n =_\beta \text{pred fibo}'$
>
> $B \equiv \lambda f.\, \lambda n.\, \text{if (isZero}\, n)\, \text{then 0}\ \ (\text{if (isZero(pred}\, n))\, \text{then 1 else suma}\, (f(\text{pred}\, n))\ \ (\text{f(pred(pred n))}))$
> $\text{fibo} \equiv \mathbf{Y}B$

Practica 2

- Ejercicio 10
- Ejercicio 12
- Ejercicio 5
- Ejercicio 6
- Ejercicio 8
- Ejercicio 9

Ejercicio 10

> **Ejercicio 10**
> 1. Pruebo por inducción en la derivación $r \rightarrow s$
>   CB) E-IfTrue
> $r = \text{if T then } t_2 \text{ else } t_3$
> $s = t_2$
>   Dado que T es un valor, la única posibilidad es que $\text{T} \rightarrow^* \text{T}$, por lo que en este caso hay que probar que $r \rightarrow^* r$, lo cual es valido por ser $\rightarrow^*$ reflexiva.
>
> CB) E-IfFalse
> Análogo al caso anterior.
>
> PI) E-If
>
> - HI) $t_1 \rightarrow^* t_1' \implies \text{if } t_1 \text{ then } t_2 \text{ else } t_3 \rightarrow^* \text{if } t_1' \text{ then } t_2 \text{ else } t_3$
>   Se aplica la regla E-If teniendo $t_1 \rightarrow t_1'$ como subderivación inmediata, por lo tanto $r \equiv \text{if } t_1 \text{ then } t_2 \text{ else } t_3$ y $s \equiv \text{if } t_1' \text{ then } t_2 \text{ else } t_3$.

Ejercicio 12

> **Ejercicio 12**
> Tengo que probar que todo valor en $\mathbb{NB}$ es una forma normal, es decir:
> $t \in \mathcal{V} \implies \nexists t' \in \mathcal{T} : t \rightarrow t'$
>
> Equivalentemente, para cada termino t:
> $\exists t' \in \mathcal{T} : t \rightarrow t' \implies t \notin \mathcal{V}$
>
> Pruebo por inducción en el conjunto de valores:
> CB) $t = F \ \lor \ t = T \ \lor \ t = 0$, vale ya que no se puede aplicar ninguna regla de derivación a estos términos.
> □
>
> HI) La propiedad vale para un valor v
>
> PI) $t = \text{Succ}\, v$
> La única regla que podría aplicarse es E-Succ, pero esta requiere que se pueda derivar el termino v lo cual por HI es imposible.
>
> Por inducción, todo valor es una forma normal.
> ■

Ejercicio 5

> **Ejercicio 5**
> Quiero probar que:
>
> $$t \rightarrow t' \land t \rightarrow t'' \implies t' = t''$$

> Pruebo por inducción en la derivación $t \to t'$
>
> Caso base (Aplicación E-IFTrue)
> Se tiene que t es de la forma $\text{if T then } t' \text{ else } t_3$ y que la regla evaluó a t'.
> Para evaluar a t'' hay que usar otra regla, pero:
>
> - No se puede usar E-IFFalse, ya que no se tiene un termino de la forma $\text{if F then } \ldots$
> - No se puede usar E-If, ya que para eso se requiere que T evalue a algun termino, pero esto no se cumple.
>   Por lo tanto la única regla que se puede utilizar es E-IfTrue y por ende $t' = t''$
>
> Caso base (Aplicación E-IFFalse)
> Análogo al caso anterior
>
> Paso inductivo(Aplicación E-If)
> Se tiene que t es de la forma $\text{if } t_1 \text{ then } t_2 \text{ else } t_3$
> Y t' es de la forma, $\text{if } t_1' \text{ then } t_2 \text{ else } t_3$
> Y como hipótesis de inducción(HI), $t_1 \to t_1' \wedge t_1 \to t_1'' \implies t_1' = t_1''$
>
> Donde $t_1 \to t_1'$:
> Por hipótesis de inducción, se tiene que t_1 solo puede evaluar a t_1'.
> Además, t_1 no puede ser ni T ni F, ya que de ser así entonces t_1 no puede evaluar a ningún termino.
> Por lo tanto, solo se puede aplicar la regla E-If, donde $t_1 \to t_1'$ es la única derivación posible que se puede utilizar, $\therefore$ se cumple que $t = t''$

Ejercicio 6

> **Ejercicio 6**
> Quiero probar que
>
> $$\nexists\, t' \in \mathcal{T} : t \to t' \implies t = \text{T} \vee t = \text{F}$$
>
> O lo que es lo mismo, por propiedad contrarrecíproca:
>
> $$t \neq \text{T} \wedge t \neq \text{F} \implies \exists t' \in \mathcal{T} : t \to t'$$
>
> Pruebo la propiedad por inducción en el conjunto de términos:
>
> $CB)\ t = \text{T} \vee t = \text{F}$
> En este caso, la propiedad vale por vacuidad
>
> HI) Existen t_1, t_2, t_3, términos tales que $t_i \neq \text{T} \wedge t_i \neq \text{F} \implies \exists t' \in \mathcal{T} : t_i \to t'$
>
> PI) Quiero probar la propiedad a demostrar para el termino $\text{if } t_1 \text{ then } t_2 \text{ else } t_3$
>
> Si $t_1 = T$, se puede aplicar la regla E-IfTrue por lo que vale la propiedad.
> Si $t_1 = F$, se puede aplicar la regla E-IfFalse por lo que vale la propiedad.
> Si $t_1 = \text{if } t_4 \text{ then } \ldots$, por HI existe t' tal que $t_1 \to t'$ y se puede aplicar la regla E-If por lo que vale la propiedad.
>
> Por inducción primitiva en el conjunto de términos, vale la propiedad.

Ejercicio 8

> **Ejercicio 8**
> Quiero probar la propiedad diamante, es decir, si $r, s, t, u \in \mathcal{T}$:
>
> $$\left.\begin{matrix} r \to s \\ r \to t \end{matrix}\right\} \implies \begin{matrix} s \to^* u \\ t \to^* u \end{matrix}$$
>
> Pruebo esta propiedad por inducción en la derivación $r \to s$:

CB) E-IfTrue
La ultima regla aplicada fue E-IfTrue por lo que
$r = \text{if T then } t_1 \text{ else } t_2$
$s = t_1$

No es posible aplicar reglas de evaluación para t_1 ya que es un valor, ni para t_3. Pero podría ser posible aplicar una regla de evaluación E-Funny2(suponiendo que $t_1 \to t_1'$) para llegar a que $r \to \text{if T then } t_1' \text{else } t_2$, tomo este ultimo termino como t .

Se tiene que
$r \to s \equiv t_1 \to t_1' \implies s \to^* t_1'$
$r \to t \overset{E-IfTrue}{\to} t_1' \implies t \to^* t_1'$

Por lo que se cumple la propiedad

CB) E-IfFalse
Análogo al anterior

PI) E-If
Suponiendo que $r = \text{if } t_1 \text{ then } t_2 \text{ else } t_3$.

Supongo que $t_1 \to t_1'$

Mi HI es que $\left.\begin{matrix} t_1 \to s \\ t_1 \to t \end{matrix}\right\} \implies \begin{matrix} s \to^* u \\ t \to^* u \end{matrix}$

Aplico la regla E-If para derivar s de r, se tiene que $s = \text{if } t_1' \text{ then } t_2 \text{ else } t_3$

t_1 no es un valor(ya que se puede aplicar una regla de derivación), por lo que hay 2 escenarios.

Escenario A) Se podría aplicar la regla E-Funny2 para derivar $t \equiv \text{if } t_1 \text{ then } t_2' \text{ else } t_3$, (sabiendo que $t_2 \to t_2'$).

Se puede aplicar E-Funny2 a s y E-If a t para derivar el termino $u = \text{if } t_1' \text{ then } t_2' \text{ else } t_3$ desde ambos, por lo que vale la propiedad.

Escenario B) Se podría aplicar la regla $E - If$ para derivar $t \equiv \text{if } t_1'' \text{ then } t_2 \text{ else } t_3$(sabiendo que $t_1 \to t_1''$), pero por HI sabemos que $\exists u \in \mathcal{T} : t_1' \to^* u \ \wedge t_1'' \to^* u$.

Por lo que, mediante la sucesión de múltiples aplicaciones de E-IF, se tiene que $s \to^* \text{if } u \text{ then } t_2 \text{ else } t_3$ y $t \to^* \text{if } u \text{ then } t_2 \text{ else } t_3$.

PI) E-Funny2
Análogo a E-If, solo que trabajando sobre el argumento del then

---

Ejercicio 9

Ejercicio 9

1. Determinismo de semántica de paso grande
   Quiero probar que si $t \Downarrow v \ \wedge t \Downarrow v' \implies v = v'$
   Pruebo por inducción sobre la derivación $t \Downarrow v$.
   \
   CB) B-Val
   No se puede aplicar ninguna otra regla, por lo que vale por vacuidad
   PI) B-IfTrue
   Se tiene que $t = \text{if } t_1 \text{ then } t_2 \text{ else } t_3$, donde $t_1 \Downarrow \text{T}$.
   $t_1 \not\Downarrow \text{F}$, ya que eso invalida la hipótesis inductiva. Por lo que no se puede aplicar B-IfFalse.
   Si $t_2 \Downarrow v \implies t \Downarrow v$
   Si $t_2 \Downarrow v' \implies t \Downarrow v'$
   Por HI sobre t_2, $v = v'$ y por ende vale la propiedad para t.
   PI) B-IfFalse
   Análogo al caso anterior.

> 2. Terminación
> Quiero probar que para todo termino t existe un valor v tal que $t \Downarrow v$
> Pruebo por inducción sobre la derivación $t \Downarrow v$
> CB) B-Val
> En este caso, $v \Downarrow v$, por lo que el valor buscado es el mismo v
> PI) B-IfTrue
> Se tiene que $t = \text{if } t_1 \text{ then } t_2 \text{ else } t_3$, donde $t_1 \Downarrow T$ y $t_2 \Downarrow v$
> Por lo tanto $t \Downarrow v$, y v es el valor buscado.
> PI) B-IfFalse
> Análogo al anterior.

Practica 3

- Ejercicio 10
- Ejercicio 11
- Ejercicio 12
- Ejercicio 6
- Ejercicio 8
- Ejercicio 9

Ejercicio 10

> **Ejercicio 10**
> a)
> $\underline{0} = \lambda x.\, \text{false}$
> $\underline{n+1} = \text{pair true } \underline{n}$
>
> $\text{isNotZero} \equiv \text{fst} \equiv \lambda p.\, p \text{ true}$
> $\text{isNotZero } \underline{0}$
> $\equiv$
> $(\lambda p.\, p \text{ true})\ \underline{0}$
> $=_\beta$(E-App Abs)
> $\underline{0} \text{ true}$
> $\equiv$
> $(\lambda x.\, \text{false}) \text{ true}$
> $\equiv$
> false
>
> $\text{isNotZero (pair true } \underline{n})$
> $\equiv$
> $\text{fst (pair true } \underline{n})$
> $=_\beta$
> true
>
> $isZero \equiv \text{not isNotZero}$
>
> $\text{succ} \equiv \lambda x.\, \text{pair true } x$
> $\text{pred} \equiv \lambda\text{x. if (isNotZero x) then (snd x) else } \underline{0}$
>
> Defino foldn:
> $\text{foldn} \equiv \mathbf{Y}\ (\lambda f.\, \lambda p\ s\ z.\ \text{ if (isNotZero } p)\ (s\ (f((\text{snd } p)\ s\ z))) \text{ else } z)$
>
> $\text{suma} \equiv \lambda n\ m.\, \text{foldn } n \text{ succ } m$

Ejercicio 11

> **Ejercicio 11**
> $\text{prod } n\ m \equiv \underline{n \cdot m} \equiv \text{foldn } \underline{m} \text{ (suma } \underline{n})\ \underline{0}$
> $\underline{n^m} \equiv \text{foldn } \underline{m} \text{ (prod } \underline{n})\ \underline{1}$

Ejercicio 12

> **Ejercicio 12**
> $\text{pred } \underline{n} =_\beta \underline{n-1}$

> Uso tuplamiento y foldn
> $\text{pred} \equiv \lambda n.\, \text{fst}\, \text{foldn}\, \underline{n}\; B\; (\text{pair}\, 0\, 0)$
> $B \equiv \lambda p.\, \text{pair}(\text{snd}\, p, \text{Succ}(\text{snd p}))$
>
> Luego la resta es
> $\text{resta} \equiv \lambda n\; m.\, \text{foldn}\, m\, \text{pred}\, n$

Ejercicio 6

> **Ejercicio 6**
> a)
> $(\lambda x.\, \lambda y.\, xy)(\lambda y.\, yz)$
> $\rightarrow_\beta$ (E-App ABS)
> $\lambda y.\, (\lambda y.\, yz)y$
> $\rightarrow_B$ (E-ABS)
> $\lambda y.\, yz$
> $\rightarrow_\eta$
> z
>
> b)
> $(\lambda x.\, \lambda y.\, xy)(\lambda z.\, yz)z$
> $\rightarrow_a$
> $(\lambda x.\, \lambda y.\, xy)(\lambda u.\, vu)w$
> $\rightarrow_\beta$
> $(\lambda y.\, (\lambda u.\, vu)y)w$
> $\rightarrow_\beta$
> $(\lambda y.\, vy)w$
> $\rightarrow_\beta$
> vw
>
> c)
> $(\lambda x.\, (\lambda y.\, x)y(\lambda z.\, z))(\lambda y.\, yz)$
> $\rightarrow_\beta$
> $(\lambda x.\, x(\lambda z.\, z))(\lambda y.\, yz)$
> $\rightarrow_\beta$
> $(\lambda y.\, yz)(\lambda z.\, z)$
> $\rightarrow_\beta$
> $(\lambda z.\, z)z$
> $\rightarrow_\beta$
> z
>
> d)
> $\Omega\, \Omega$
> $\rightarrow_\beta$
> $(\lambda x.\, f(x\; x))(\lambda x.\, f(x\; x))$
> $\rightarrow_\beta$
> $f((\lambda x.\, f(x\; x))\; (\lambda x.\, f(x\; x)))$
> $\rightarrow_\alpha$
> $f(\Omega\, \Omega)$
>
> $(\lambda f.\, (\lambda x.\, f(x\; x))(\lambda x.\, f(x\; x)))$
> $\rightarrow_\alpha$
> $\lambda f.\, \Omega\, \Omega$
> $\rightarrow_\beta^*$
> $\lambda f.\, f(\Omega\, \Omega)$
> $\rightarrow_\beta^*$
> $\lambda f.\, f(f(\Omega\, \Omega))$

Ejercicio 8

Ejercicio 8
a)
Tengo que probar que
$M[N/x]$ es una β-nf $\implies$ M es una β-nf
Equivalentemente
$M[N/x]$ no tiene un β-redex $\implies$ M no tiene un β-redex
Por contrarrecíproca
M tiene un β-redex $\implies$ $M[N/x]$ tiene un β-redex

Pruebo por inducción en la sustitución

CB) $M \equiv x$, $M \equiv y$
M no tiene un β-redex, vale por vacuidad.
□

CB) $M \equiv \lambda x. P \vee M \equiv \lambda y. P$, $x \notin FV(P)$
$M \equiv M[N/x]$, por lo que la propiedad vale trivialmente.
□

PI) $M \equiv P\,Q$

Tengo las siguientes hipótesis inductivas:
HI.1) P tiene un β-redex $\implies$ $P[N/x]$ tiene un β-redex
HI.2) Q tiene un β-redex $\implies$ $Q[N/x]$ tiene un β-redex

Supongo que $P\,Q$ tiene un β-redex
$M[N/x] \equiv (P\,Q)[N/x] \equiv (P[N/x]\;Q[N/x])$

Si P o Q tienen un β-redex entonces por HI.1) o por HI.2), $M[N/x]$ contiene un β-redex.
Si ni P ni Q tienen un β-redex entonces M no tiene un β-redex, lo que contradice la hipótesis.

Se tiene entonces que M tiene un β-redex $\implies$ $M[N/x]$ tiene un β-redex.
□

PI) $M \equiv \lambda y. P$, $x \in FV(P) \wedge y \notin FV(N)$

HI) P tiene un β-redex $\implies$ $P[N/x]$ tiene un β-redex

Supongo que M tiene un β-redex
$M[N/x] \equiv \lambda y. P[N/x]$
Si P tiene un β-redex, entonces por HI, $M[N/x]$ tiene un β-redex
Si P no tiene un β-redex, M tampoco. Contradicción.
□

PI) $M \equiv \lambda y. P$, $x \in FV(P) \wedge y \in FV(N)$

Sea z un identificador tal que $z \notin FV(NP)$(existe al menos uno, ya que P y N contienen una cantidad finita de identificadores).

HI) Si $P[z/y]$ tiene un β-redex $\implies$ $(P[z/y])[N/x]$ tiene un β-redex.

$M[N/x] \equiv \lambda z. (P[z/y])[N/x]$
Supongo que M tiene un β-redex:
Si P no tiene un β-redex entonces M no tiene un β-redex, lo cual es una contradicción.

Si P tiene un β-redex, entonces $P[z/y]$ tiene un β-redex. Ya que la operación $P[z/y]$ es una α-conversión las β-conversiones no se ven afectadas.

Por HI, $(P[z/y])[N/x]$ tiene un β-redex y por ende $M[N/x]$ también.
□

Por inducción en la sustitución, vale la propiedad.
■

Ejercicio 9

Ejercicio 9
a)
$Y = \lambda x.\,(\lambda y.\,x(y\;y))(\lambda y.\,x(y\;y))$

YX
$\equiv$
$(\lambda x.\,(\lambda y.\,x(y\;y))(\lambda y.\,x(y\;y)))X$
$=_\beta$(E-App ABS)
$(\lambda y.\,X(y\;y))(\lambda y.\,X(y\;y))$
$=_\beta$(E-App ABS)
$X((\lambda y.\,X(y\;y))\;(\lambda y.\,X(y\;y)))$
$=_\beta$(E-App ABS inversa)
$X((\lambda x.\,(\lambda y.\,x(y\;y))(\lambda y.\,x(y\;y)))\;X)$
$\equiv$
$X(YX)$

b)
Dado $Y = \lambda x.\,(\lambda y.\,x(y\;y))(\lambda y.\,x(y\;y))$
La única aplicación posible es E-ABS en el cuerpo del lambda termino
Y
$=_\beta$(E-ABS)
$\lambda x.\,x((\lambda y.\,x(y\;y)(\lambda y.\,x(y\;y))$

El termino que estábamos reduciendo se repite, por lo que no existe una forma normal.

---

c)
Sea Z un λ-termino, quiero resolver la ecuación $x\;y_1\ldots\;y_n = Z$, es decir, encontrar el termino X tal que:

$$X\;y_1\ldots\;y_n =_\beta Z[X/x]$$

$X = \lambda\;y_1\ldots y_n.\;Z[X/x]$(esto no sirve pero no me acuerdo porque)

Practica 4

- Ackermann en sistema T
- Ejercicio 14
- Ejercicio 6
- Ejercicio 8

Ackermann en sistema T

> **Ackermann en sistema T**
> La función de Ackermann se define como
>
> $$A(m,n) = \begin{cases} n+1 & \text{si } m = 0 \\ A(m-1,1) & \text{si } m > 0 \wedge n = 0 \\ A(m-1, A(m,n-1)) & \text{si } m > 0 \wedge n > 0 \end{cases}$$
>
> $A\,0\,0 \equiv 1$
> $A\,0\,n \equiv n+1$
>
> $A\;0 = \lambda n.\,\mathrm{succ}\,n$
> $A\;m\;0 = A$
>
> $A = \lambda m.\,R\;(\lambda n.\,\mathrm{succ}\,n)\;(\lambda f.\,\lambda r.\,)\;m$
> ???

Ejercicio 14

> **Ejercicio 14**
> Se definen los términos nil y cons que tienen que cumplir la especificación:
> $\mathrm{fold}\,\mathrm{nil}\,f\,b \equiv b$
> $\mathrm{fold}\;\;(\mathrm{cons}\,x\;xs)\;f\,b \equiv\;\; f\;x\;(\mathrm{fold}\;\;xs\;f\;b)$
>
> Fijando $\mathrm{fold} = \mathrm{id}$ se pueden definir los otros operadores como:
>
> - $\mathrm{nil} \equiv \lambda f.\,\lambda b.\,b$
> - $\mathrm{cons} \equiv \lambda x.\,\lambda xs.\,\lambda f.\,\lambda b.\,f\;x\;(xs\;f\;b)$
>
> Y se pueden tipar las listas como $\mathrm{List}\,X = \Lambda Y.\,(X \to Y \to Y) \to Y \to Y$
> Luego:
> $\mathrm{nil} : \forall X.\,\mathrm{List}\,X$
> $\mathrm{nil} \equiv \Lambda X.\,\Lambda R.\,\lambda f : (X \to Y \to Y).\,\lambda b : Y \to b$
> $\mathrm{cons} : \forall X.\,X \to \mathrm{List}\,X \to \mathrm{List}\,X$
> $\mathrm{cons} \equiv \Lambda X.\,\lambda x : X.\,\lambda xs : \mathrm{List}\,X.\,\Lambda Y.\,\lambda f : (X \to Y \to Y).\,\lambda b : Y.\,f\;x\;(xs\langle Y\rangle\;f\;b)$
>
> A partir de acá se pueden definir las funciones del enunciado:
> map se puede definir a partir de fold:
> `map f xs = fold xs (\x ys-> cons (f x) ys) nil`
>
> O aplicando eta-reduce:
> `map f xs = fold xs (cons . f) nil`
>
> Pasando a sistema F:
> $\mathrm{map} : \forall X\;\forall Y.\,(X \to Y) \to \mathrm{List}\,X \to \mathrm{List}\,Y$
> $\mathrm{map} \equiv \Lambda X.\,\Lambda Y.\,\lambda f : (X \to Y).\,\lambda xs : \mathrm{List}\,X.\,xs\langle \mathrm{List}\,Y\rangle\;(\lambda x : X.\,\lambda ys : \mathrm{List\,Y}.\,\mathrm{cons}\langle Y\rangle\;(f\;x)\;ys)\;\;\mathrm{nil}\langle Y\rangle$
>
> append se define como:
> `append xs ys = fold xs (\x ys' -> cons x ys') ys`

> O aplicando eta-reduce:
> `append xs = fold xs cons`
>
> En sistema F:
> $\text{append} : \forall X. \text{List } X \to \text{List } X \to \text{List } X$
> $\text{append} \equiv \Lambda X. \lambda xs : \text{List } X. xs\langle \text{List } X\rangle \, \text{cons}\langle X\rangle$
>
> $reverse$ se define como:
> `snoc x xs = append xs (cons x nil)`
> `reverse xs = fold xs snoc nil`
>
> En sistema F:
> $\text{snoc} : \forall X. X \to \text{List } X \to \text{List } X$
> $\text{snoc} \equiv \Lambda X. \lambda x : X. \lambda xs : \text{List } X. \text{append}\langle X\rangle \; xs \; (\text{cons}\langle X\rangle \; x \; \text{ nil})$
> $\text{reverse} : \forall X. \text{List } X \to \text{List } X$
> $\text{reverse} \equiv \Lambda X. \lambda xs : \text{List } X. xs\langle \text{List X}\rangle \; \text{ snoc}\langle X\rangle \; \text{ nil}$
>
> (revisar bien los tipos)

Ejercicio 6

> **Ejercicio 6**
> Si se considera un sistema de este tipo, entonces todos los tipos son de la forma:
> $T := \text{Unit} \mid T \to T$
> Mientras que los términos son de la forma:
> $t := x \mid c \mid \lambda x. t \mid tt$
>
> Quiero probar que $\forall t : (\exists t' \in t : t \to t' \land \Gamma \vdash t' : T) \implies \Gamma \vdash t : T$
>
> Pruebo por inducción en el conjunto de términos:
>
> La propiedad solo puede valer para términos no atascados, ya que estos son los únicos que pueden tener un tipo. Por lo tanto hago inducción asumiendo que t es un tipo no atascado.
>
> CB.1) $t = \star$ o $t = x$
> Estos términos no evalúan a ningún otro termino, por lo que la propiedad vale.
>
> PI.1) Si la propiedad vale para t entonces vale para $\lambda x. t$
> No se puede aplicar ninguna regla de evaluación para este termino por lo que la propiedad vale
>
> PI.2) Si la propiedad vale para los términos t_1 y t_2 entonces vale para $t_1 t_2$
>
> Si $t_1 \to t_1'$ entonces se puede aplicar la regla E-APP1 para que $t_1 t_2$ evalué a $t_1' t_2$.
> Si $\Gamma \vdash t_1' t_2 : T_2$ entonces solo se puede aplicar la regla T-APP para inferir los tipos de t_1' y t_2. Se tiene que $t_1' : T_1 \to T_2$ y $t_2 : T_2$. T_1 y T_2 son tipos genéricos.
>
> Por HI, $\Gamma \vdash t_1 : T_1 \to T_2$, luego por regla T-APP, $\Gamma \vdash t_1 t_2 : T_2$. La propiedad se cumple en este caso.
>
> Si t_1 no evalúa a ningún termino entonces t_1 es un valor, solo quedan las posibles evaluaciones de t_2.
>
> Si $t_2 \to t_2'$ y $\Gamma \vdash t_2' : T_2$ entonces por HI, $\Gamma \vdash t_2 : T_2$. Luego por la regla T-APP $\Gamma \vdash t_1 t_2 : T_2$ y $\Gamma \vdash t_1 t_2' : T_2$, como la única regla aplicable es T-APP2 se tiene que $t_1 t_2 \to t_1 t_2'$ y la propiedad se cumple.
>
> PI.3) $t \equiv (\lambda x. t) v$
> En este caso la propiedad se tiene que cumplir para $\lambda x. t$ y v, ambos son valores por lo tanto la propiedad se cumple per se.
>
> Ahora se tiene que cumplir la propiedad cuando se aplica la regla E-APPABS:
> $(\lambda x. t) v \to t[v/x]$
> $\Gamma \vdash t[v/x] : T \implies \Gamma \vdash (\lambda x. t) v : T$
>
> Si se asume que $\Gamma \vdash v : T$ y $\Gamma, x : T \vdash t : T$ entonces

> - Por T-APP, $t : T$
> - Por lema de sustitución, $t[v/x]$
>
> La propiedad se cumple
>
> ■

Ejercicio 8

> **Ejercicio 8**
> $\text{fst} \equiv \lambda x\ y.\, x$
> $\text{snd} \equiv \lambda x\ y.\, y$
> $\text{pred} \equiv \lambda n.\, R\, 0\, \text{snd}\, n$
>
> ---
>
> $$\text{suma} \equiv \lambda n\ m.\, R\, n\ (\lambda x\ y.\, \text{succ}\, x) m$$
>
> $$\text{mult} \equiv \lambda n\ m.\, R\, 0\ (\lambda i\ j.\, \text{suma}\, i\ n)\ m$$
>
> $$\text{is0} \equiv \lambda n.\, R\ \text{ true}\ (\lambda a\ b.\, \text{false})\ n$$
>
> $\text{sub} \equiv \lambda n\ m.\, R\ n\ (\lambda i\ j.\, \text{pred}\, i) m$
> $\text{and} \equiv \lambda a\ b.\, D\ a\ (D\ b\, \text{true}\, \text{false})\, \text{false}$
> $\text{eq0} \equiv \lambda n\ m.\, \text{and}\, (\text{is0}(\text{sub}\, \text{n}\, \text{m}))\ (\text{is0}(\text{sub}\, \text{m}\, \text{n}))$
