PLP - Recuperatorio del Primer Parcial - 1er cuatrimestre de 2024

#Orden	Nro. Libreta	Apellido(s)	Nombre(s)
81	[redacted]	Fialkowski	Valentín.

Corregido por	Nota E1	Nota E2	Nota E3	Nota Final
PERLA	B-	B-	B-	A

Este examen se aprueba obteniendo al menos dos ejercicios bien menos (B-) y uno regular (R). Las notas para cada ejercicio son: -, I, R, B-, B. Entregar cada ejercicio en hojas separadas. Poner nombre, apellido y número de orden en todas las hojas, y numerarlas. Se puede utilizar todo lo definido en las prácticas y todo lo que se dio en clase, colocando referencias claras. El orden de los ejercicios es arbitrario. Recomendamos leer el parcial completo antes de empezar a resolverlo.

Ejercicio 1 - Programación funcional

Aclaración: en este ejercicio no está permitido utilizar recursión explícita, a menos que se indique lo contrario.

En este ejercicio vamos a modelar lógica proposicional en Haskell, de modo de poder construir fórmulas proposicionales y evaluarlas bajo distintas valuaciones.

```
data Prop = Var String | No Prop | Y Prop Prop | O Prop Prop | Imp Prop Prop

type Valuación = String -> Bool
```

Por ejemplo, la expresión: `Y (Var ''P'') (No (Imp (Var ''Q'') (Var ''R'')))` representa la proposición $P \wedge \neg(Q \Rightarrow R)$.

Las valuaciones se representan como funciones que a cada variable proposicional le asignan un valor booleano. Por ejemplo la valuación `\x -> x == ''P''` le asigna el valor verdadero a la variable P y falso a todas las otras variables proposicionales.

a) Dar el tipo y definir las funciones `foldProp` y `recProp`, que implementan respectivamente los esquemas de recursión estructural y primitiva para el tipo `Prop`. Solo en este inciso se permite usar recursión explícita.

b) Definir la función `variables :: Prop -> [String]`, que dada una fórmula devuelve la lista con todas sus variables proposicionales en algún orden, sin elementos repetidos.

Por ejemplo: `variables (O (Var ''P'') (No (Y (Var ''Q'') (Var ''P''))))` debería devolver la lista `[''P'',''Q'']` o la lista `[''Q'',''P'']`.

c) Definir la función `evaluar :: Valuación -> Prop -> Bool`, que indica si una fórmula es verdadera o falsa para una valuación dada.

d) Definir la función `estáEnFNN :: Prop -> Bool`, que indica si una fórmula está en Forma Normal Negada. Es decir, si no tiene implicaciones y la negación se aplica únicamente a variables y no a proposiciones más complejas.

Por ejemplo: `Y (Var ''P'') (No (Imp (Var ''Q'') (Var ''R'')))` no está en FNN, y en cambio `Y (Var ''P'') (Y (Var ''Q'') (No (Var ''R'')))` sí lo está.

Ejercicio 2 - Demostración e inferencia

Considerar las siguientes definiciones sobre árboles con información en las hojas[1]:

```
data AIH a = Hoja a | Bin (AIH a) (AIH a)

     esHoja :: AIH a -> Bool
{E0} esHoja (Hoja x) = True
{E1} esHoja (Bin i d) = False

     izq :: AIH a -> AIH a
{I}  izq (Bin i d) = i

     der :: AIH a -> AIH a
{D}  der (Bin i d) = d

     mismaEstructura :: AIH a -> AIH a -> Bool
{M0} mismaEstructura (Hoja x) = esHoja
{M1} mismaEstructura (Bin i d) = \t -> not (esHoja t) &&
            mismaEstructura i (izq t) && mismaEstructura d (der t
```

a) Demostrar la siguiente propiedad:

$\forall$ `t::AIH a` . $\forall$ `u::AIH a` . mismaEstructura t u = mismaEstructura u t

Se recomienda hacer inducción en el primer árbol, utilizando extensionalidad en el segundo. Se permite definir macros (poner nombres a expresiones largas para no tener que repetirlas).

No es obligatorio reescribir los $\forall$ correspondientes en cada paso, pero es importante recordar que están presentes y escribir los que correspondan al platear la propiedad como predicado unario. Recordar también que los = de las definiciones pueden leerse en ambos sentidos.

Se consideran demostradas todas las propiedades conocidas sobre enteros y booleanos.

b) Usar el algoritmo W para inferir juicios de tipado válidos para las siguientes expresiones, o indicar por qué no es posible (recordar que en inferencia **no está permitido** renombrar variables):

i) $(\lambda x.x(\lambda x.\mathsf{Succ}(x)))(\lambda x.x)$

ii) $\lambda x.\mathsf{if}\ \mathsf{isZero}(x)\ \mathsf{then}\ x\ \mathsf{else}\ x\ \mathsf{zero}$

Ejercicio 3 - Cálculo Lambda Tipado

Se desea extender el cálculo lambda simplemente tipado para modelar **Árboles con información en las hojas**. Para eso se extienden los tipos y expresiones de la siguiente manera:

$\tau ::= \cdots \mid \mathtt{AIH}(\tau)$

$M ::= \cdots \mid \mathtt{Hoja}(M) \mid \mathtt{Bin}(M, M) \mid \mathsf{case}\ M\ \mathsf{of\ Hoja}\ x \leadsto M;\ \mathsf{Bin}(i, d) \leadsto M$

- $\mathtt{AIH}(\tau)$ es el tipo de los árboles con información en las hojas de tipo τ.
- $\mathtt{Hoja}(M)$ es un árbol compuesto por una única hoja con información M.
- $\mathtt{Bin}(M_1, M_2)$ es un árbol compuesto por dos subárboles M_1 y M_2.
- El observador $\mathsf{case}\ M_1\ \mathsf{of\ Hoja}\ x \leadsto M_2;\ \mathsf{Bin}(i, d) \leadsto M_3$ permite acceder al valor de un árbol que es hoja (el cual se ligará a la variable x que puede aparecer libre en M_2), y a los dos subárboles de un árbol que no es hoja (los cuales se ligarán a las variables i y d que pueden aparecer libres en M_3).

a. Introducir las reglas de tipado para la extensión propuesta.

b. Definir el conjunto de valores y las nuevas reglas de semántica operacional a pequeños pasos, tanto de congruencia como de cómputo.

c. Mostrar paso por paso cómo reduce la expresión:

$\mathsf{case}\,(\lambda n : \mathsf{Nat}.\ \mathtt{Hoja}(n))\ \mathsf{Succ(zero)}\ \mathsf{of\ Hoja}\ x \leadsto \mathsf{Succ(Pred}(x));\ \mathsf{Bin}(i, d) \leadsto \mathsf{zero}$

d. Definir como macro la función esHoja_τ, que toma un $\mathtt{AIH}(\tau)$ y devuelve un booleano que indica si es una hoja.

[1] Escritas con recursión explícita para facilitar las demostraciones.

Ejercicio 1
Valentín
Frakowski
Nro.=81

1) a) foldProp :: (String → a) → (a → a) → (a → a → a) → (a → a → a) → (a → a → a) → Prop → a.

~~foldProp fvar fnot fy fo fimp t~~

```
foldProp fvar _ _ _ _ (Var s) = (fvar s)
foldprop fvar fnot fy fo fimp (Prop) = case of Prop
  (No p) = fnot (recn p)
  (Y p1 p2) = fy (recn p1) (recn p2)
  (O p1 p2) = fo (recn p1) (recn p2)
  (Imp p1 p2) = fimp (recn p1) (recn p2).
where recn = foldprop fvar fnot fy fo fimp
```

es Prop con p minúscula. no es ~~Prop~~ el tipo de datos Prop. (con P mayúscula).

recProp :: (String → a) → (Prop → a → a) → (Prop → Prop → a → a → a) → (Prop → Prop → a → a → a) → (Prop → Prop → a → a → a) → Prop → a.

```
recProp fvar _ _ _ _ (Var s) = fvar s
recProp fvar fnot fy fo fimp Prop = case of Prop :
  (No p) = fnot p (recn p)
  (Y p1 P2) = fy p1 p2 (recn p1) (recn p2)
  (O p1 p2) = fo p1 p2 (recn p1) (recn p2)
  (Imp p1 p2) = fimp p1 p2 (recn p1) (recn p2).
  where recn = recProp fvar fnot fy fo fimp.
```

¿DE QUÉ ES LA 'R'?

b) variablesR :: Prop → [String].

variablesR = foldProp (\s → [s]) ~~[illegible]~~ id f f f

where f = (\p1 p2 → p1 ++ p2) = (++)

variables :: Prop → [String]

variables p = eliminarRepetidos (variablesR p).

PODÍAS USAR NUB, O MEJOR, UNION EN LUGAR DE ++

eliminarRepetidos :: [a] → [a]

eliminarRepetidos = rec id (\x xs r → if (~~pertenece~~ elem x xs)

¿recr? []

then r

FUNCIONABA IGUAL CON foldr

else [x] ++ r

(x : r)

~~pertenece :: a → [a] → Bool~~

~~pertenece x = foldr (const false) x~~

Asumo que elem ya está definido en el preludio de Haskell

y rec

y tienen tipos elem :: a → [a] → Bool

rec :: b → (a → [a] → b → b) → [a] → b.

Valentín Fialkowski
N.O. = 81.

c) evaluar :: Valuación → Prop → Bool

~~evaluar val = recProp (\s → evaluar val s) (f val) (f val)~~
~~(f val) (f val)~~
~~where f₁ val = \p r → ¬ (val p)~~

```
                                   val              not
evaluar val = foldprop (\s → val s) (\r → ¬ r)
                       (\r1 r2 → r1 && r2) (&&)      not
                       (\r1 r2 → r1 || r2) (||)
                       (\r1 r2 → (¬ r1) || r2)
```

d) estáEnFNN :: Prop → Bool

```
estáEnFNN = recprop (const True)
                    (faux              ¿QUÉ ES ESTE NOMBRE?
                    (\_ _ r1 r2 → r1 && r2)
                    (\_ _ r1 r2 → r1 && r2)
                    (\_ _ _ _ → False)

where
faux :: Prop → Bool → Bool
faux (Var _) _ = True
faux _       _ = False.
```

② Valentín Fialkowski
N.O = 82.

a) demostrar:

$\forall t :: AIH\ a .\ \forall u :: AIH\ a.$ mismaEstructura $t\ u$ = mismaEstructura $u\ t$

Empiezo haciendo inducción sobre t.

· El ppo. de inducción estructural me dice:

Sea $P(t) = \forall u :: AIH\ a.$ mismaEstructura $t\ u$
= misma Estructura $u\ t$.

· si se cumple

$\forall n :: a .\ P(\text{Hoja } n)$ y

$\forall i :: AIH\ a .\ \forall d .\ AIH\ a .\ P(i) \wedge P(d) \rightarrow P(\text{Bin } i\ d)$.

entonces $\forall t :: AIH\ a .\ P(t)$.

· P(Hoja n) (no escribo los ∀ pero están presentes)

: mismaEstructura (Hoja n) u = mismaEstructura u (Hoja n)

(· Defino la macro ME = misma Estructura).

. esHoja u = misma Estructura u (Hoja n) {M0}.

Ahora utilizo extensionalidad sobre u: $(\exists m :: a)$.
· si $u :: AIH\ a$, entonces, o bien u = Hoja m
o bien u = Bin l r
$(\exists l :: AIH\ a .\ \exists r :: AIH\ a)$.

• caso u = Hoja m.

• esHoja (Hoja m) = ME (Hoja m) (Hoja n)

$\downarrow$ {E0} $\qquad\qquad$ $\downarrow$ {M0}

True = esHoja (Hoja n)

$\downarrow$ {E0}

~~True =~~

True = True ✓.

P(Hoja n) vale para u = Hoja (m).

caso u = Bin l r

• esHoja (Bin l r) = ME (Bin l r) (Hoja n)

$\downarrow$ {E1} $\qquad\qquad$ $\downarrow$ {M1} $\quad$ {BETA}

False = not (EsHoja (Hoja n)) &&
ME l (izq (Hoja n)) &&
ME r (der (Hoja n)).

$\neq$

como, por {E0}: EsHoja (Hoja n) = True

entonces not (esHoja (Hoja n)) = False

reemplazando:

False = False && ME l (izq (Hoja n)) &&
ME l (der (Hoja n)).

Por prop. de booleanos False && X = False para cualquier X :: Bool.

entonces:

Ejercicio 2. Valentín Fiolkowski N.O = 81

$False = False$ ✓.

Queda demostrado $P(Hoja\ n)$ para $u = (Bin\ l\ r)$.

• Ahora $P(Bin\ i\ d) =$ (está presente el $\forall u :: AIH\ a$).

~~~~ • $ME\ (Bin\ i\ d)\ u = ME\ u\ (Bin\ i\ d)$.

[Aparte: tengo ~~la~~ la siguiente Hipótesis inductiva:
$P(i) \wedge P(d)$: $(ME\ i\ u = ME\ u\ i \ \wedge$
$(\forall u :: AIH\ a)$ $ME\ d\ u = ME\ u\ d.)$]

sigo con $P(Bin\ i\ d)$:
Reemplazando por $\{M\emptyset_1\}$: {BETA}

(not (esHoja u) && ME i (izq u) && ME d (der u).
= ME u (Bin i d).)

• Ahora uso extensionalidad ~~de~~ sobre u de nuevo.
si $u :: AIH\ a$, entonces

o bien ~~la~~ ~~grava~~ $\exists m :: a .\ u = Hoja\ m$,

o bien $(\exists t :: AIH\ a .\ \exists r :: AIH\ a)$. $u = (Bin\ l\ r)$.

y separo en casos para $u = Hoja\ m$ y $u = (Bin\ l\ r)$
~~~~

· caso u = Hoja m. Primero ~~Primero~~ Analizo la parte izquierda de la igualdad:

$\underbrace{\text{not (EsHoja (Hoja m))}}_{\text{False (por \{E0\} y \{not\})}}$ && $\underbrace{\text{ME i izq (Hoja m) \&\& ME d der (Hoja m)}}_{*}$

⇒: False && * = False. {&&}

P queda:

False = ME (Hoja m) (Bin i d)

False = esHoja (Bin i d) {Mo}

False = False ✓ {E1}.

queda demostrado P (Bin i d) para u = Hoja m. ✓

· caso u = (Bin l r). Analizo la parte izq del '='. (en P)

not (EsHoja (Bin l r)) && ~~[illegible]~~

ME i izq (Bin l r) && ME d der (Bin l r).

~~A~~ reemplazo not (EsHoja (Bin l r)) por True (por {E2} y not).

True && ME i izq (Bin l r) && ME d der (Bin l r).

(por prop. de booleanos)

ME i izq (Bin l r) && ME d der (Bin l r)

Ahora, por {I} y por {D}

~~P~~ queda: (ME i l) && (ME d r) = ME (Bin l r) (Bin i d)
P

Valentín Frolkowski
N.O. = 81
Ejercicio 2.

Ahora analizo la parte derecha de la igualdad.

ME (Bon l r) (Bon i d):

not EsHoja (Bon i d) && ME l (izq (Bon i d))
&& ME r (der (Bon i d)).

{ME1}
{BOTH}

(procedimiento similar a la parte izquierda de la igualdad).

la propiedad queda: (junto ahora los dos lados de la igualdad)
(P)

~~ME l i && ME r d.~~

FALTAN PASOS

ME i l && ME d r = ME l i && ME r d.

Ahora podemos aplicar la hipótesis inductiva $P(i) \land P(d)$.

~~como P(i): ME ∀u :: AHa. ME i u = ME~~

como $P(i) \land P(d)$: $\forall u :: AHa. \begin{pmatrix} ME\ i\ u = ME\ u\ i & \land \\ ME\ d\ u = ME\ u\ d & \end{pmatrix}$

en particular: $\begin{pmatrix} ME\ i\ L = ME\ L\ i \\ \land\ ME\ d\ r = ME\ r\ d \end{pmatrix}$

y luego: ME i L && ME d r = ME L i && ME r d

$\leftrightarrow$ ME i L = ME L i $\land$ ME d r = ME r d.

lo considero una propiedad de booleanos conocida.

. ~~Aplicando~~ lo cual vale por H.I. ✓

queda demostrado P(Bon i d) para u = (Bon L r)

Por lo tanto. vale $\forall t :: AHa.\ P(t)$.

NO HACE FALTA, HACIENDO LOS REEMPLAZOS YA TE QUEDA LO QUE QUERÉS

Valentín Fialkowski

N.O. = 81.

Ejercicio 2) b)

b) ⑧ $(\lambda x.\ x\ (\lambda x.\ succ(x)))\ (\lambda x.\ x).$

Aplicación

⑥ $(\lambda x.\ x\ (\lambda x.\ succ(x)))$ ⑦ $(\lambda x.\ x)$

Abs | Abs.

⑤ $x\ (\lambda x.\ succ(x))$ ⑨ x

Ap.

① x ④ $\lambda x.\ suc(x)$

Abs.

③. $suc(x)$

suc.

②. x

ups. me olvidé de ese término. (me di cuenta cuando terminé ⑥) (lo llamo ⑨).

①. $W(x) \rightsquigarrow x: X_1 \vdash x: X_1$ X_1: ~~variable~~ incógnita.

②. $W(x) \rightsquigarrow x: X_2 \vdash x: X_2$ X_2: incógnita.

③ ~~W(succ)~~

$S = mgu(X_2 \stackrel{?}{=} Nat)$

$S = \{X_2 := Nat\}.$

$$\frac{②.\ W(x) \rightsquigarrow x: X_2 \vdash x: X_2}{W(succ(x)) \rightsquigarrow x: Nat \vdash succ(x): Nat.}$$

④

$$\frac{③\ W(succ(x)) \rightsquigarrow x: Nat \vdash succ(x): Nat}{W(\lambda x.\ succ(x)) \rightsquigarrow \emptyset \vdash \lambda x{:}Nat.\ succ(x): Nat \to Nat.}$$

⑤.

$$S = mgu(\{X_1 \overset{?}{=} (\mathrm{Nat} \to \mathrm{Nat}) \to X_5\})$$

$$S = \{X_1 := (\mathrm{Nat} \to \mathrm{Nat}) \to X_5\}.$$

① : $W(x) \rightsquigarrow x : X_1 \vdash x : X_1$

④ : $W(\lambda x.\, succ(x)) \rightsquigarrow$
$\emptyset \vdash \lambda x : \mathrm{Nat}.\, succ(x) : \mathrm{Nat} \to \mathrm{Nat}.$

$$\frac{}{W(x\ (\lambda x.\ succ(x))) \rightsquigarrow x : (\mathrm{Nat} \to \mathrm{Nat}) \to X_5 \vdash x\ (\lambda x : \mathrm{Nat}.\ succ(x)) : X_5}$$

⑥

$$\Gamma_1(x) = \mathrm{Nat} \to \mathrm{Nat} \to X_5$$

$$\frac{⑤\ W(x\ \lambda x.\ succ(x)) \rightsquigarrow x : (\mathrm{Nat} \to \mathrm{Nat}) \to X_5 \vdash x\ (\lambda x : \mathrm{Nat}.\ succ(x)) : X_5}{W(\lambda x.\ x\ (\lambda x.\ succ(x))) \rightsquigarrow \emptyset \vdash (\lambda x : (\mathrm{Nat} \to \mathrm{Nat}) \to X_5.\ x\ (\lambda x : \mathrm{Nat}.\ suc(x)))}$$

Γ_1:

$: ((\mathrm{Nat} \to \mathrm{Nat}) \to X_5) \to X_5$

⑨

⑨ $W(x) \rightsquigarrow x : X_9 \vdash x : X_9.$ ✓

⑦.

$$\frac{⑨\ W(x) \rightsquigarrow x : X_9 \vdash x : X_9.}{W(\lambda x.\, x) \rightsquigarrow \emptyset \vdash (\lambda x : X_9 .\, x) : X_9 \to X_9.}$$ ✓

⑧.

⑥ $W(\lambda x. x(\lambda x. succ(x))) \rightsquigarrow$

$\emptyset \vdash (\lambda x: (Nat \to Nat) \to x_5 .$
$x\ (\lambda x: Nat . succ(x)))$
$: ((Nat \to Nat) \to x_5) \to x_5$

⑦ $W(\lambda x. x)$
$\rightsquigarrow \emptyset \vdash (\lambda x: x_9 . x) : x_9 \to x_9$

$S = mgu\{((Nat \to Nat) \to x_5) \to x_5 \overset{?}{=} (x_9 \to x_9) \to x_8\}$

x_8: incógnita.

$W((\lambda x. \underbrace{x(\lambda x. succ(x)))}_{⑥}\ \underbrace{(\lambda x. x)}_{⑦}) \rightsquigarrow \emptyset \vdash$
$((\lambda x: (Nat \to Nat) \to x_5 .$
$x\ (\lambda x: Nat . succ(x)))$
$(\lambda x: Nat \to Nat . x))$
$: x_5 \to x_5$

---

con $S = \{ x_9 := Nat \to Nat,\ x_8 := x_5 \to x_5 \}$

¡ojo con los paréntesis!

Este es el juicio de tipado válido para la expresión.

Ehw

2) b)

II. ⑧ $\lambda x.$ if $isZero(x)$ then x else $x\ zero$

Abs

⑦ if $isZero(x)$ then x else $x\ zero$

② $isZero(x)$ ③ x ⑥. $x\ zero$

① x ④. x ⑤ zero.

✓

Valentin Tidlkowski

N.O. = 81

Ejercicio 2)b).

①. $W(x) \rightsquigarrow x : X_1 \vdash x : X_1$ ✓

②. ~~W(iszero(x))~~

$$\frac{W(x) \rightsquigarrow x : X_1 \vdash x : X_1 \qquad S = mgu(X_1 \stackrel{?}{=} Nat) = \{X_1 := Nat\}}{W(iszero(x)) \rightsquigarrow x : Nat \vdash iszero(x) : Bool.}$$ ✓

③. $W(x) \rightsquigarrow x : X_3 \vdash x : X_3.$ ✓

④. $W(x) \rightsquigarrow x : X_4 \vdash x : X_4$ ✓

⑤. $W(zero) \rightsquigarrow \phi \vdash zero : Nat.$ ✓

⑥.

$$\frac{\begin{array}{l} ④\ W(x) \rightsquigarrow x : X_4 \vdash x : X_4 \\ W(zero) \rightsquigarrow \phi \vdash zero : Nat \end{array} \qquad \begin{array}{l} S = mgu\{X_4 \stackrel{?}{=} Nat \to X_6\} \\ = \{X_4 := Nat \to X_6\}. \end{array}}{W(x\ zero) \rightsquigarrow x : Nat \to X_6 \vdash x.zero : X_6.}$$ ✓

⑦.

$$\frac{\begin{array}{l} ②.\ W(iszero(x)) \rightsquigarrow x : Nat \vdash iszero(x) : Bool \\ ③.\ W(x) \rightsquigarrow x : X_3 \vdash x : X_3 \\ ⑥\ W(x\ zero) \rightarrow x : Nat \to X_6 \vdash x\ zero : X_6 \end{array} \qquad \begin{array}{l} S = mgu(\\ \{Bool \stackrel{?}{=} Bool,\ X_3 \stackrel{?}{=} X_6\} \\ \cup \{Nat \stackrel{?}{=} X_3,\ \cancel{X_3} \stackrel{?}{=} \\ X_3 \stackrel{?}{=} Nat \to X_6\}. \\ Nat \stackrel{?}{=} Nat \to X_6 \end{array}}{W(if\ \underbrace{iszero(x)}_{②}\ then\ \underbrace{x}_{③}\ else\ \underbrace{x\ zero}_{⑥}) \rightsquigarrow falla.}$$

· ~~S/~~ falla porque el algoritmo para encontrar a S.

(Martelli - Montanari) falla

· Justificación.

$mgu(\{Bool \overset{?}{=} Bool,\ x_3 \overset{?}{=} x_6,\ \cancel{Nat}\ x_3 \overset{?}{=} Nat,\ x_3 \overset{?}{=} Nat \to x_6\})$

$\xrightarrow{delete}\ x_3 \overset{?}{=} x_6,\ x_3 \overset{?}{=} Nat,\ x_3 \overset{?}{=} Nat \to x_6$

$\xrightarrow[x_3 := x_6]{elim}\ x_6 \overset{?}{=} Nat,\ x_6 \overset{?}{=} Nat \to x_6.$

$\xrightarrow[x_6 := Nat]{elim}\ Nat \overset{?}{=} Nat \to Nat \ \longrightarrow\ falla$

por clash.

✓

Valentín Fialkowski

N.O. = 81.

Ejercicio 2)b).

Valentín Fiałkowski
N.O. = 81.

Ejercicio 3

3) a) reglas de tipado.

$$\frac{\Gamma \vdash M : \tau}{\Gamma \vdash \mathrm{Hoja}(M) : \mathrm{AIH}(\tau)}\ \mathrm{hoja}_M$$

$$\frac{\Gamma \vdash M_1 : \mathrm{AIH}(\tau) \qquad \Gamma \vdash M_2 : \mathrm{AIH}(\tau)}{\Gamma \vdash \mathrm{Bin}(M_1, M_2) : \mathrm{AIH}(\tau)}\ \mathrm{Bin}$$

$$\frac{\Gamma \vdash M : \mathrm{AIH}(\tau) \qquad \Gamma, x:t \vdash M_1 : \sigma \qquad \Gamma, i : \mathrm{AIH}(\tau), d : \mathrm{AIH}(\tau) \vdash M_2 : \sigma}{\Gamma \vdash \text{case } M \text{ of } \mathrm{Hoja}\ x \rightsquigarrow M_1 ;\ \mathrm{Bin}(i,d) \rightsquigarrow M_2 : \sigma}$$

b)
$V ::= \ldots \mid \mathrm{Hoja}(V) \mid \mathrm{Bin}(V, V)$

nuevas
reglas de semántica operacional: (cómputo)

• $\{case_1\}$: ~~case(Hoja(v))~~

~~case ... of~~

case (Hoja v) of Hoja $x \rightsquigarrow M_1$; Bin $(i,d) \rightsquigarrow M_2$

$\to M_1 \{x := v\}$.

{case2}

case $\mathrm{Bin}(v_1, v_2)$ of $\mathrm{Hoja}\ x \rightsquigarrow M_1$; $\mathrm{Bin}(i,d) \rightsquigarrow M_2$

$\to M_2 \{i := v_1\} \{d := v_2\}$.

reglas de congruencia: si $M \to M'$, $M_1 \to M_1'$, $M_2 \to M_2'$

{hoja1} $\mathrm{Hoja}(M) \to \mathrm{Hoja}(M')$

{Bin1} $\mathrm{Bin}(M_1, M_2) \to \mathrm{Bin}(M_1', M_2)$.

{Bin2} $\mathrm{Bin}(V, M_2) \to \mathrm{Bin}(V, M_2')$.

{case3} case M of $\mathrm{Hoja}\ x \rightsquigarrow M_1$; $\mathrm{Bin}(i,d) \rightsquigarrow M_2$

$\to$ case M' of $\mathrm{Hoja}\ x \rightsquigarrow M_1$; $\mathrm{Bin}(i,d) \rightsquigarrow M_2$.

c) case $[(\lambda n: Nat, \mathrm{Hoja}(n))\ succ(zero)]$

of $\mathrm{Hoja}\ x \rightsquigarrow succ(pred(x))$; $\mathrm{Bin}(i,d) \rightsquigarrow zero$

Por {case3} tengo que reducir:

$(\lambda n: Nat . \mathrm{Hoja}(n))\ succ(zero)$

$\xrightarrow{\beta} \mathrm{Hoja}(n)\{n := succ(zero)\} = \mathrm{Hoja}(succ(zero))$

case $\mathrm{Hoja}(suc(zero))$ of $\mathrm{Hoja}\ x \rightsquigarrow succ(pred\ x))$;

$\mathrm{Bin}(i,d) \rightsquigarrow zero$.

Usando {zero,}

Valentin Fiałkowski

N.O. = 81.

{Serwo}

$\rightarrow$ succ (pred x)) } x := succ (zero) {

= succ (pred (succ (zero)))

{pred} $\rightarrow$ {succ + pred} succ (zero) ✓.

[redacted]

[redacted]