Estructuras Discretas 2017-1
Proyecto 8: Números Binarios Positivos

Emiliano Galeana Araujo 314032324
Márquez Castillo Irene 314318424

Facultad de ciencias, UNAM

Fecha de entrega: Viernes 25 de octubre del 2016

1. Números Binarios Positivos

1.1. Universo

Empecemos dando la definición de los números binarios positivos:

Definición 1 *Dado un tipo BinarioPos, esta gramática representa números binarios mayores o iguales a uno, donde:*

$$BinarioPos ::= UjCeroBinarioPosjUnoBinarioPos$$

U: Representa al dígito uno.
Cero x: Representa al binario x0 donde x es un binario.
Uno x: Representa al binario x1 donde x representa un binario.

Definición 2 *Queremos implementar un número bonario a través de la expresión donde dado un elemento de binario suponemos que es una lista que consiste en ceros y unos.*

```
type Binario = [Int]
```

2. Implementación

1. BinPos :: U — Cero(BinPos) — Uno (BinPos)

2. Las siguientes son funciones recursivas para el tipo de dato BinPos.

   -- | sucesor. Función que devuelve el sucesor de un elemento de Binario.
   sucesor :: BinarioPos →BinarioPos

   -- | suma. Función recursiva que devuelve la suma de dos elementos de BinarioPos.
   suma :: BinarioPos →BinarioPos →BinarioPos

   -- | resta. Función recursiva que devuelve la resta positiva de dos elementos de BinarioPos
   resta :: BinarioPos →BinarioPos →BinarioPos

   -- | producto. Función recursiva que devuelve el producto de dos elementos de BinarioPos.
   producto :: BinarioPos →BinarioPos →BinarioPos

   -- | binPosToInt. Función recursiva que dado un elemento de BinarioPos devuelve el número natural representante bajo el tipo Int.
   binPosToInt :: BinarioPos →Int

   -- | intToBinPos. Función recursiva que dado un elemento que represente a un número natural de tipo Int devuelva su representante bajo el tipo BinarioPos.
   intToBinPos :: Int →BinarioPos

3. Teoría

Teorema 3.1 *La función* ***`intToBinPos`*** *definida en* ***`Haskell`*** *es la inversa izquierda de la función* ***`binToIntPos`****, es decir, para toda lista que representa un número binario b se cumple que:*

$$intToBinPos(binPosToIntb) = b$$

Demostración 3.1 P.D `intToBinPos(BinPosToInt b)=b`

Paso Base b = U

$$intToBinPos(binPosToIntU) = U$$

$$intToBinPOs(1) = U$$

$$U = U$$

Hipótesis de Inducción intToBinPos(binPosToInt n) = n Paso Inductivo

$$intToBinPos(binPosToIntsucesorb) = sucesorb$$

$$intToBinPos(binPosToIntS(b)) = S(b)$$

$$sucesorb = sucesorb$$

$\therefore intToBinPos(binPosToIntsucesorU) = sucesorU$
$\therefore intToBinPos(binPosToIntb) = b$

Teorema 3.2 *La función* ***`intToBinPos`****definida en Haskell es la inversa derecha de la función binPosToInt, es decir, para todo número natural n represen- tado bajo el tipo Int se cumple que:*

$$binPosToInt(intToBinPosn) = n$$

Demostración 3.2 P.D `binPosToInt(intToBinPos n)=n`

Paso Base n = 1

$$binPosToInt(intToBinPos1) = 1$$

$$intToBinPos(U) = 1$$

$$Pordefiniciónde binToIntPosU = 1$$

$$(U) = 1$$

$$1 = 1$$

Hipótesis de Inducción binPosToInt(intToBinPos n)=n
Paso Inductivo sea una n $\geq$ 1

`Por Demostrar`$binPosToInt(intToBinPosS(n)) = S(n)$ *1°sea n par, entonces n = 2k, su sucesor es de la forma 2k + 1 para S(n)impar*

$$binPosToInt(intToBinPosS(n)) = S(n)$$

$$binPosToInt(intToBinPos(2k+1))$$

$$binPosToInt(intToBinPos((2k+1)/2))$$

$$binPosToInt(Uno(intToBinPos((2k+1)/2+1/2))))$$

La función divisora vuelve el 1/2 a cero y divide al 2 de 2k, dejando a k sola

$$binPosToInt(Uno(intToBinPosk))$$

Por definición de binPosToInt llamamos a la función binAux y queda de la forma

$$binAux(Uno(intToBinPosk))0 = S(n)$$

$$(binAux(intToBinPosk)) + 2^0$$

]

$$(binAux(intToBinPosk)1) + 1$$

$$S(binAux(intToBinPosk))1$$

$$S(2k) = 2k + 1$$

$\therefore binPosToInt(intToBinPosS(n)) = S(n)\ \forall n(n \geq 1 \wedge n = 2k+1)$
2° sea n una n impar, entonces n = 2k + 1, su sucesor es de la forma 2k $binPosToInt(intToBin$
$S(n)$

$$binPosToInt(intToBinPos(2k)$$

$$binPosToInt(intToBinPos((2k)/2))$$

$$binPosToInt(Uno(intToBinPos((2k)/2)))$$

La función divisora divide al 2 de 2k, dejando a k sola

$$binPosToInt(Cero(intToBinPosk))$$

Por definición de binPosToInt llamamos a la función binAux y queda de la forma

$$binAux((intToBinPosk)0) = S(n)$$

$$(binAux(intToBinPosk)0)$$

$$(binAux(intToBinPosk)1)$$

$$S(binAux(intToBinPos2k))$$

$$S(2k + 1) = 2k$$

$\therefore binPosToInt(intToBinPosS(n)) = S(n)$ ∀ *n (n* $\geq 1 \wedge n = 2k)$

$\therefore binPosToInt(intToBinPosS(n)) = S(n)$ ∀ *n ((binPosToInt(intToBinPos n))=n)*

La función ***suma****definida en Haskell es correcta, es decir para cualquiera b1, b2 número binario po* $b1 + b2$*Donde b1 + b2 es la suma usual de números binarios.*

P.D `suma b1 b2 = b1 + b2`

Paso Base b1 = U y b2 = U

$$sumaUU = U + U$$

$$(CeroU) = sucesorU$$

$$(CeroU) = (CeroU)$$

Hipótesis de Inducción suma b1 b2 = b1 + b2

Paso Inductivo Sea b1 un binario

Caso 1: P.d. suma (Cero b1) b2 = (Cero b1) + b2

caso 1.1: b2 = U

$$suma(Cerob1)U$$

$$= sucesor(Cerob1)$$

$$= (Cerob1) + 1$$

caso 1.2: b2 = (Cero b2)

$$suma(Cerob1)(Cerob2)$$

$$= Cero(sumab1b2)$$
$$= Cero(b1 + b2)$$
$$= 0(b1 + b2)$$
$$= (b1, 0) + (b2, 0)$$

caso 1.3: b2 = (Uno b2)

$$suma(Cerob1)(Unob2)$$
$$= Uno(sumab1b2)$$
$$= Uno(b1 + b2)$$
$$= 1(b1 + b2)$$
$$= (b1, 1) + (b2, 1)$$

caso 2: P.d. suma(Uno b1) b2 = (Uno b1) + b2
caso 2.1: b2 = U

$$suma(Unob1)U$$
$$= sucesor(Unob1) + 1$$
$$= Cero(sucesorb1) + 1$$

caso 2.2: b2 = (Cero b2)

$$suma(Unob1)(Cerob2)$$
$$= Uno(sumab1b2)$$
$$= Uno(b1 + b2)$$
$$= 1(b1 + b2)$$
$$= (b1, 1) + (b2, 1)$$

caso 2.3: b2 = (Uno b2)

$$suma(Unob1)(Unob2)$$
$$= Cero(sumaU(sumab1b2))$$
$$= Cero(sumaU(b1 + b2))$$

$\therefore sumab1b2 = b1 + b2$