Universidad Nacional Autónoma de México

Facultad de Ciencias

Tarea Examen II

Lenguajes de Programación

Dafne Bonilla Reyes
José Camilo García Ponce
Rodrigo Aldair Ortega Venegas

Profesor: Javier Enríquez Mendoza

Ayudante: Kevin Axel Prestegui Ramos
Ayudante: Karla Denia Salas Jiménez
Ayudante Laboratorio: Ramón Arenas Ayala
Ayudante Laboratorio: Oscar Fernando Millán Pimentel

Octubre 2023

Lenguajes de Programación

Tarea Examen 2

1. (3 pts) Chon Hacker quiere quiere desarrollar un lenguaje calculador para expresiones aritméticas en notación polaca. La notación polaca es un antiguo sistema que no requiere de paréntesis, para lograrlo mueve los operadores al final de la expresión, es decir, utiliza notación posfija. Por ejemplo, la expresión $1 + 2$ se convierte en $2\,1\,+$ y la expresión $1 - (3 + 2)$ corresponde a $1\,3\,2\,+\,-$.

   Este lenguaje evalúa las expresiones metiendo cada símbolo a una pila hasta que se encuentra un operador, al ver un operador saca los dos símbolos que se encuentran en el tope de la pila, evalúa la operación con ellos y agrega el resultado a la pila.

   Una gramática con la que se puede definir este lenguaje es la siguiente:

$$\begin{array}{lcl} \texttt{sym} & ::= & n \mid + \mid - \mid * \\ \texttt{rpn} & ::= & \varepsilon \mid \texttt{sym rpn} \end{array}$$

   Responde a los siguientes incisos:

   a) Con la gramática anterior se pueden construir programas que no pertenecen al lenguaje como $+1\,2$ o $1 + 2$. Para tratar con esto Chon Hacker te pide definir mediante reglas de inferencia una semántica estática que verifique que la expresión es correcta según la definición de la notación polaca. Para esto se define el juicio e correct que indica que la expresión e está correctamente en notación polaca.

$$\frac{}{\varepsilon \text{ correct}}\ \varepsilon \qquad \frac{}{\text{num}[n]}\ \text{correctN}$$

$$\frac{\text{correct(e1)} \qquad \text{correct(e2)}}{e1 \quad e2 \quad +}\ \text{correctSum}$$

$$\frac{\text{correct(e1)} \qquad \text{correct(e2)}}{e1 \quad e2 \quad -}\ \text{correctRes}$$

$$\frac{\text{correct(e1)} \qquad \text{correct(e2)}}{e1 \quad e2 \quad *}\ \text{correctMult}$$

   b) Usando la definición del inciso anterior verifica que $1\,2\,4\,3\,+\,*\,-$ correct Aplicamos las reglas de inferencia:

$$\dfrac{\dfrac{\dfrac{}{1}\text{ correctN} \qquad \dfrac{\dfrac{}{2}\text{ correctN} \qquad \dfrac{\dfrac{}{4}\text{ correctN} \qquad \dfrac{}{3}\text{ correctN}}{4 \quad 3 \quad +}\ \text{correctSum}}{2 \quad 4 \quad 3 \quad + \quad *}\ \text{correctMult}}{1 \quad 2 \quad 4 \quad 3 \quad + \quad * \quad -}\ \text{correctRes}$$

   Como podemos ver, basados en las reglas definidas en el inciso A, la expresión "1 2 4 3 + * -" es correcta.

   c) Define una semántica operacional de paso grande para este lenguaje calculador mediante la relación $\Downarrow$ que evalúa los programas del lenguaje. Puede ser de ayuda leer el programa de derecha a izquierda.

$$\frac{}{\text{num}[n] \quad \Downarrow \quad \text{num}[n]}\ \text{bsnum}$$

$$\frac{e_1 \Downarrow \text{num}[n] \qquad e_2 \Downarrow \text{num}[m]}{e_1 \quad e_2 \quad + \quad \Downarrow \quad \text{num}[n +_{\mathbb{N}} m]}\ \text{bssum}$$

$$\frac{e_1 \Downarrow \text{num}[n] \qquad e_2 \Downarrow \text{num}[m]}{e_1 \quad e_2 \quad - \quad \Downarrow \quad \text{num}[n -_{\mathbb{N}} m]}\ \text{bsres}$$

$$\frac{e_1 \Downarrow \text{num}[n] \qquad e_2 \Downarrow \text{num}[m]}{e_1 \quad e_2 \quad * \quad \Downarrow \quad \text{num}[n *_{\mathbb{N}} m]}\ \text{bsmult}$$

d) Define una semántica operacional de paso pequeño con estados de la forma $s \models e$ en donde s es la pila que almacena los símbolos (la pila vacía se denota como $\circ$) y e es la expresión del lenguaje. Para esto indica claramente cuáles son los estados iniciales y finales y define la relación de transición $\to_{\texttt{rpn}}$ que modela la evaluación del programa.
Estados iniciales:

$$\frac{e\ \text{expr} \qquad \circ\ \text{emptyStack}}{\circ \models e \quad \text{inicial}}\ \text{inicial}$$

Estados finales:

$$\frac{s\ \text{stack}}{s \models \varepsilon \quad \text{final}}\ \text{final} \qquad \frac{}{\text{num}[n] \models \varepsilon \quad \text{final}}\ \text{final}$$

Transiciones:

$$\frac{}{s \models \text{num}[n] \quad \to_{rpn} \quad \text{num}[n] : s \models e}\ \text{ssnum}$$

$$\frac{}{s \models \text{num}[n]\ \text{num}[m]\ + \quad \to_{rpn} \quad \text{num}[n +_{\mathbb{N}} m] : s \models e}\ \text{sssnumf}$$

$$\frac{e_1 \to_{rpn} e_1'}{s \models e_1\ e_2\ + \quad \to_{rpn} \quad (e_1' +_{\mathbb{N}} e_2) : s \models e}\ \text{sssnum1}$$

$$\frac{e_2 \to_{rpn} e_2'}{s \models \text{num}[n]\ e_2\ + \quad \to_{rpn} \quad (\text{num}[n] +_{\mathbb{N}} e_2') : s \models e}\ \text{sssnum2}$$

$$\frac{}{s \models \text{num}[n]\ \text{num}[m]\ - \quad \to_{rpn} \quad \text{num}[n -_{\mathbb{N}} m] : s \models e}\ \text{ssresf}$$

$$\frac{e_1 \to_{rpn} e_1'}{s \models e_1\ e_2\ - \quad \to_{rpn} \quad (e_1' -_{\mathbb{N}} e_2) : s \models e}\ \text{ssres1}$$

$$\frac{e_2 \to_{rpn} e_2'}{s \models \text{num}[n]\ e_2\ - \quad \to_{rpn} \quad (\text{num}[n] -_{\mathbb{N}} e_2') : s \models e}\ \text{ssres2}$$

$$\frac{}{s \models \text{num}[n]\ \text{num}[m]\ * \quad \to_{rpn} \quad \text{num}[n *_{\mathbb{N}} m] : s \models e}\ \text{ssmultf}$$

$$\frac{e_1 \to_{rpn} e_1'}{s \models e_1\ e_2\ * \quad \to_{rpn} \quad (e_1' *_{\mathbb{N}} e_2) : s \models e}\ \text{ssmult1}$$

$$\frac{e_2 \to_{rpn} e_2'}{s \models \text{num}[n]\ e_2\ + \quad \to_{rpn} \quad (\text{num}[n] +_{\mathbb{N}} e_2') : s \models e}\ \text{ssmult2}$$

e) Prueba que las semánticas definidas en los incisos anteriores son equivalentes, para esto:

- Demuestra que para cualquier expresión correcta e del lenguaje, si $e \Downarrow v$ entonces $\circ \models e \to^\star_{\texttt{rpn}} \circ \models v$:
  - Caso base (num[n]):
    Paso pequeño:
    num[n] $\to^\star_{\texttt{rpn}}$ num [n]
    Paso grande:
    num[n] $\Downarrow$ num[n]
  - Hipótesis de inducción: Sean e_1 y e_2 expresiones correctas del lenguaje:
    Si $e_1 \Downarrow v_1$ entonces $\circ \models e_1 \to^\star_{\texttt{rpn}} \circ \models v_1$ y
    Si $e_2 \Downarrow v_2$ entonces $\circ \models e_2 \to^\star_{\texttt{rpn}} \circ \models v_2$
  - Paso inductivo:
    Suma paso grande:
    Si $e_1 \Downarrow v_1 = num[n]$ y $e_2 \Downarrow v_2 = num[m]$, evaluamos $e_1\ e_2\ +$:
    Nos sale $e_1\ e_2\ + \Downarrow num[n+m]$
    Suma paso pequeño:
    Por HI. sabemos que si $e_1 \Downarrow v_1$ y $e_2 \Downarrow v_2$ entonces $\circ \models e_1 \to^\star_{\texttt{rpn}} \circ \models v_1 = num[n]$ y $\circ \models e_2 \to^\star_{\texttt{rpn}} \circ \models v_2 = num[m]$, evaluamos $e_1\ e_2\ +$:
    Nos sale $e_1\ e_2\ + \to^\star_{\texttt{rpn}} v1\ v2\ + \to num[n+m]$
    Como podemos ver, las semánticas nos dan el mismo resultado
    La resta y multiplicación son casos análogos, por lo tanto la hipótesis es correcta
- Prueba que para cualquier expresión correcta e del lenguaje $\circ \models e \to^\star_{\texttt{rpn}} \circ \models v$ de tal forma que $\circ \models v$ es final, entonces $e \Downarrow v$:
  - Caso base (num[n]):
    Paso pequeño:
    num[n] $\to^\star_{\texttt{rpn}}$ num [n]
    Paso grande:
    num[n] $\Downarrow$ num[n]
  - Hipótesis de inducción: Sean e_1 y e_2 expresiones correctas del lenguaje:
    Si $\circ \models e_1 \to^\star_{\texttt{rpn}} \circ \models v_1$ entonces $e_1 \Downarrow v_1$ y
    Si $\circ \models e_2 \to^\star_{\texttt{rpn}} \circ \models v_2$ entonces $e_2 \Downarrow v_2$
  - Paso inductivo :
    Pd. $e_1\ e_2\ + \to^\star_{\texttt{rpn}} v$ entonces $e_1\ e_2\ + \Downarrow v$
    Suma paso pequeño:
    Si $e_1 \to^\star_{\texttt{rpn}} v_1$ =num[n] y $e_2 \to^\star_{\texttt{rpn}} v_2$ =num[m], entonces:
    $s \models e_1\ e_2\ + \to^\star_{\texttt{rpn}} s \models v_1\ v_2\ + \to num[n+m] : s \models \varepsilon = v$
    Suma paso grande:
    $e_1\ e_2\ +$: por HI. $e_1 \Downarrow v_1$ = num [n] y $e_2 \Downarrow v_2$ = num [m]
    Entonces: $e_1\ e_2\ + \Downarrow$ num[$n+m$] = v
    Como podemos ver si $e_1\ e_2\ + \to^\star_{\texttt{rpn}} v$ entonces $e_1\ e_2\ + \Downarrow v$
    Para la resta y multiplicación son casos análogos, por lo tanto la hipótesis es correcta.

2. (3 pts) Los numerales de *Barendregt*, denotados $\widehat{n}$, se definen como sigue:

$$\begin{array}{rcl} \widehat{0} & =_{def} & \lambda x.x \\ \widehat{n+1} & =_{def} & \texttt{false}, \widehat{n} \end{array}$$

En esta definición recuerde que tanto $false$ como la operación de par ordenado $\langle \cdot, \cdot \rangle$ corresponden a ciertas abstracciones lambda definidas en clase.

Se realizó lo siguiente para los numerales de Barendregt:

a) Demuestre que $\widehat{n+1}\,\widehat{0} \to^\star_\beta \widehat{0}$ y que $\widehat{n+1}\ \widehat{m+1} \to^\star_\beta \widehat{m}\ \widehat{n}$

Primero veremos que $< false, \widehat{n} > \to^*_\beta (\lambda b.b\ false\ \widehat{n})$
$< false, \widehat{n} > =_{def} pair\ false\ \widehat{n} =_{def} (\lambda f.\lambda s.\lambda b.b\ f\ s)\ false\ \widehat{n}$
$\to_\beta (\lambda s.\lambda b.b\ false\ s)\ \widehat{n} \to_\beta (\lambda b.b\ false\ \widehat{n})$

Ahora veamos que $\widehat{n+1}\ \widehat{0} \to^*_\beta \widehat{0}$
$\widehat{n+1}\ \widehat{0} =_{def} < false, \widehat{n} > \widehat{0} \to^*_\beta (\lambda b.b\ false\ \widehat{n})\ \widehat{0} \to_\beta \widehat{0}\ false\ \widehat{n}$
$=_{def} (\lambda x.x)\ false\ \widehat{n} \to_\beta false\ \widehat{n} =_{def} (\lambda x.\lambda y.y)\ \widehat{n} \to_\beta \lambda y.y$
$=_{def} \widehat{0}$

Por ultimo veamos que $\widehat{n+1}\ \widehat{m+1} \to^*_\beta \widehat{m}\ \widehat{n}$
$\widehat{n+1}\ \widehat{m+1} =_{def} < false, \widehat{n} >< false, \widehat{m} >$
$\to^*_\beta (\lambda b.b\ false\ \widehat{n})\ (\lambda c.c\ false\ \widehat{m}) \to_\beta (\lambda c.c\ false\ \widehat{m})\ false\ \widehat{n}$
$\to_\beta (false\ false\ \widehat{m})\ \widehat{n} =_{def} ((\lambda x.\lambda y.y)\ false\ \widehat{m})\ \widehat{n}$
$\to_\beta ((\lambda y.y)\ \widehat{m})\ \widehat{n} \to_\beta \widehat{m}\ \widehat{n}$

b) Defina la función sucesor S y verifique con su definición que $S\,\widehat{n} \to^\star \widehat{n+1}$.

La definicion es $S =_{def} \lambda x. < false, x >$
Ahora veamos si $S\ \widehat{n} \to^*_\beta \widehat{n+1}$
$S\ \widehat{n} =_{def} (\lambda x. < false, x >)\ \widehat{n} \to_\beta < false, \widehat{n} > =_{def} \widehat{n+1}$

c) Defina la función predecesor P y verifique con su definición que $P\,\widehat{n+1} \to^\star \widehat{n}$. ¿Cúal es la forma normal de $P\widehat{0}$?

La definicion es $P =_{def} \lambda x.x\ false$
Ahora veamos si $P\ \widehat{n+1} \to^*_\beta \widehat{n}$
$P\ \widehat{n+1} =_{def} (\lambda x.x\ false)\ \widehat{n+1} \to_\beta \widehat{n+1}\ false =_{def} < false, \widehat{n} > false$
$\to^*_\beta (\lambda b.b\ false\ \widehat{n})\ false \to_\beta false\ false\ \widehat{n} =_{def} (\lambda x.\lambda y.y)\ false\ \widehat{n}$
$\to_\beta (\lambda y.y)\ \widehat{n} \to_\beta \widehat{n}$

Y la forma normal de $P\ \widehat{0}$ es $\lambda x.\lambda y.y$ o $false$
$P\ \widehat{0} =_{def} (\lambda x.x\ false)\ \widehat{0} \to_\beta \widehat{0}\ false =_{def} (\lambda x.x)\ false \to_\beta false$
$=_{def} \lambda x.\lambda y.y$

d) Defina la función test de cero Z y verifique con su definición que $Z\,\widehat{n+1} \to^\star false$ y que $Z\,\widehat{0} \to^\star true$

La definicion es $Z =_{def} \lambda x.x\ true$
Ahora veamos si $Z\ \widehat{n+1} \to^*_\beta false$
$Z\ \widehat{n+1} =_{def} (\lambda x.x\ true)\ \widehat{n+1} \to_\beta \widehat{n+1}\ true =_{def} < false, \widehat{n} > true$
$\to^*_\beta (\lambda b.b\ false\ \widehat{n})\ true \to_\beta true\ false\ \widehat{n} =_{def} (\lambda x.\lambda y.x)\ false\ \widehat{n}$
$\to_\beta (\lambda y.false)\ \widehat{n} \to_\beta false$
Ahora veamos si $Z\ \widehat{0} \to^*_\beta true$
$Z\ \widehat{0} =_{def} (\lambda x.x\ true)\ \widehat{0} \to_\beta \widehat{0}\ true =_{def} (\lambda x.x)\ true \to_\beta true$

3. (3 pts) Defina utilizando combinadores de punto fijo la función `reverse` que encuentre la reversa de una lista. Utilice su definición para mostrar que

$$\texttt{reverse}\ (\texttt{cons}\ 3\ (\texttt{cons}\ 2\ (\texttt{cons}\ 1\ \texttt{nil}))) \to^\star_\beta (\texttt{cons}\ 1\ (\texttt{cons}\ 2\ (\texttt{cons}\ 3\ \texttt{nil}))).$$

Puede suponer definidas y correctas todas las funciones de listas que requiera, salvo `reverse` por supuesto.

La definición de la función será la siguiente:

$$reverse =_{def} \lambda x.reverse'\ x\ nil$$
$$reverse' =_{def} F\ reverse''$$
$$reverse'' =_{def} (\lambda f.\lambda x.\lambda a.if\ (isnil\ x)\ a\ (f\ (tail\ x)\ (cons\ (head\ x)\ a)))$$

Ahora mostremos que:

$$\texttt{reverse}\ (\texttt{cons}\ 3\ (\texttt{cons}\ 2\ (\texttt{cons}\ 1\ \texttt{nil}))) \to^{\star}_{\beta} (\texttt{cons}\ 1\ (\texttt{cons}\ 2\ (\texttt{cons}\ 3\ \texttt{nil}))).$$

Esto es:

$$reverse\ (cons\ 3\ (cons\ 2\ (cons\ 1\ nil)))$$
$$=_{def} (\lambda x.reverse'\ x\ nil)\ (cons\ 3\ (cons\ 2\ (cons\ 1\ nil)))$$
$$\to_{\beta} reverse'\ (cons\ 3\ (cons\ 2\ (cons\ 1\ nil)))\ nil$$
$$=_{def} (F\ reverse'')\ (cons\ 3\ (cons\ 2\ (cons\ 1\ nil)))\ nil$$
$$\to^{*}_{\beta} reverse''\ (F\ reverse'')\ (cons\ 3\ (cons\ 2\ (cons\ 1\ nil)))\ nil$$
$$=_{def} (\lambda f.\lambda x.\lambda a.if\ (isnil\ x)\ a\ (f\ (tail\ x)\ (cons\ (head\ x)\ a)))\ (F\ reverse'')\ (cons\ 3\ (cons\ 2\ (cons\ 1\ nil)))\ nil$$
$$\to_{\beta} (\lambda x.\lambda a.if\ (isnil\ x)\ a\ ((F\ reverse'')\ (tail\ x)\ (cons\ (head\ x)\ a)))\ (cons\ 3\ (cons\ 2\ (cons\ 1\ nil)))\ nil$$
$$\to_{\beta} (\lambda a.if\ (isnil\ (cons\ 3\ (cons\ 2\ (cons\ 1\ nil))))\ a\ ((F\ reverse'')\ (tail\ (cons\ 3\ (cons\ 2\ (cons\ 1\ nil))))\ (cons\ (head\ (cons\ 3\ (cons\ 2\ (cons\ 1\ nil))))\ a)))\ nil$$
$$\to_{\beta} if\ (isnil\ (cons\ 3\ (cons\ 2\ (cons\ 1\ nil))))\ nil\ ((F\ reverse'')\ (tail\ (cons\ 3\ (cons\ 2\ (cons\ 1\ nil))))\ (cons\ (head\ (cons\ 3\ (cons\ 2\ (cons\ 1\ nil))))\ nil))$$
$$\to^{*}_{\beta} (F\ reverse'')\ (tail\ (cons\ 3\ (cons\ 2\ (cons\ 1\ nil))))\ (cons\ (head\ (cons\ 3\ (cons\ 2\ (cons\ 1\ nil))))\ nil)$$
$$\to^{*}_{\beta} (F\ reverse'')\ (tail\ (cons\ 3\ (cons\ 2\ (cons\ 1\ nil))))\ (cons\ (head\ (cons\ 3\ (cons\ 2\ (cons\ 1\ nil))))\ nil) \to^{*}_{\beta} (F\ reverse'')\ (cons\ 2\ (cons\ 1\ nil))\ (cons\ (head\ (cons\ 3\ (cons\ 2\ (cons\ 1\ nil))))\ nil)$$
$$\to^{*}_{\beta} (F\ reverse'')\ (cons\ 2\ (cons\ 1\ nil)\ (cons\ 3\ nil)$$
$$\to^{*}_{\beta} reverse''\ (F\ reverse'')\ (cons\ 2\ (cons\ 1\ nil)\ (cons\ 3\ nil)$$
$$=_{def} (\lambda f.\lambda x.\lambda a.if\ (isnil\ x)\ a\ (f\ (tail\ x)\ (cons\ (head\ x)\ a)))\ (F\ reverse'')\ (cons\ 2\ (cons\ 1\ nil)\ (cons\ 3\ nil)$$
$$\to_{\beta} (\lambda x.\lambda a.if\ (isnil\ x)\ a\ ((F\ reverse'')\ (tail\ x)\ (cons\ (head\ x)\ a)))\ (cons\ 2\ (cons\ 1\ nil)\ (cons\ 3\ nil)$$
$$\to_{\beta} (\lambda a.if\ (isnil\ (cons\ 2\ (cons\ 1\ nil))\ a\ ((F\ reverse'')\ (tail\ (cons\ 2\ (cons\ 1\ nil)))\ (cons\ (head\ (cons\ 2\ (cons\ 1\ nil)))\ a)))\ (cons\ 3\ nil)$$
$$\to_{\beta} if\ (isnil\ (cons\ 2\ (cons\ 1\ nil))\ (cons\ 3\ nil)\ ((F\ reverse'')\ (tail\ (cons\ 2\ (cons\ 1\ nil)))\ (cons\ (head\ (cons\ 2\ (cons\ 1\ nil)))\ (cons\ 3\ nil)))$$
$$\to^{*}_{\beta} (F\ reverse'')\ (tail\ (cons\ 2\ (cons\ 1\ nil)))\ (cons\ (head\ (cons\ 2\ (cons\ 1\ nil)))\ (cons\ 3\ nil))$$
$$\to^{*}_{\beta} (F\ reverse'')\ (cons\ 1\ nil)\ (cons\ (head\ (cons\ 2\ (cons\ 1\ nil)))\ (cons\ 3\ nil))$$
$$\to^{*}_{\beta} (F\ reverse'')\ (cons\ 1\ nil)\ (cons\ 2\ (cons\ 3\ nil))$$
$$\to^{*}_{\beta} reverse''\ (F\ reverse'')\ (cons\ 1\ nil)\ (cons\ 2\ (cons\ 3\ nil))$$
$$=_{def} (\lambda f.\lambda x.\lambda a.if\ (isnil\ x)\ a\ (f\ (tail\ x)\ (cons\ (head\ x)\ a)))\ (F\ reverse'')\ (cons\ 1\ nil)\ (cons\ 2\ (cons\ 3\ nil))$$
$$\to_{\beta} (\lambda x.\lambda a.if\ (isnil\ x)\ a\ ((F\ reverse'')\ (tail\ x)\ (cons\ (head\ x)\ a)))\ (cons\ 1\ nil)\ (cons\ 2\ (cons\ 3\ nil))$$

$$\rightarrow_\beta (\lambda a.if\ (isnil\ (cons\ 1\ nil))\ a\ ((F\ reverse'')\ (tail\ (cons\ 1\ nil)))\ (cons\ (head\ (cons\ 1\ nil))\ a)))\ (cons\ 2\ (cons\ 3\ nil))$$

$$\rightarrow_\beta if\ (isnil\ (cons\ 1\ nil))\ (cons\ 2\ (cons\ 3\ nil))\ ((F\ reverse'')\ (tail\ (cons\ 1\ nil)))\ (cons\ (head\ (cons\ 1\ nil))\ (cons\ 2\ (cons\ 3\ nil))))$$

$$\rightarrow^*_\beta (F\ reverse'')\ (tail\ (cons\ 1\ nil))\ (cons\ (head\ (cons\ 1\ nil))\ (cons\ 2\ (cons\ 3\ nil)))$$

$$\rightarrow^*_\beta (F\ reverse'')\ (nil)\ (cons\ (head\ (cons\ 1\ nil))\ (cons\ 2\ (cons\ 3\ nil)))$$

$$\rightarrow^*_\beta (F\ reverse'')\ (nil)\ (cons\ 1\ (cons\ 2\ (cons\ 3\ nil)))$$

$$\rightarrow^*_\beta reverse''\ (F\ reverse'')\ (nil)\ (cons\ 1\ (cons\ 2\ (cons\ 3\ nil)))$$

$$=_{def} (\lambda f.\lambda x.\lambda a.if\ (isnil\ x)\ a\ (f\ (tail\ x)\ (cons\ (head\ x)\ a)))\ (F\ reverse'')\ (nil)\ (cons\ 1\ (cons\ 2\ (cons\ 3\ nil)))$$

$$\rightarrow_\beta (\lambda x.\lambda a.if\ (isnil\ x)\ a\ ((F\ reverse'')\ (tail\ x)\ (cons\ (head\ x)\ a)))\ (nil)\ (cons\ 1\ (cons\ 2\ (cons\ 3\ nil)))$$

$$\rightarrow_\beta (\lambda a.if\ (isnil\ (nil))\ a\ ((F\ reverse'')\ (tail\ (nil))\ (cons\ (head\ (nil))\ a)))\ (cons\ 1\ (cons\ 2\ (cons\ 3\ nil)))$$

$$\rightarrow_\beta if\ (isnil\ (nil))\ (cons\ 1\ (cons\ 2\ (cons\ 3\ nil)))\ ((F\ reverse'')\ (tail\ (nil))\ (cons\ (head\ (nil))\ (cons\ 1\ (cons\ 2\ (cons\ 3\ nil)))))$$

$$\rightarrow^*_\beta if\ (true)\ (cons\ 1\ (cons\ 2\ (cons\ 3\ nil)))\ ((F\ reverse'')\ (tail\ (nil))\ (cons\ (head\ (nil))\ (cons\ 1\ (cons\ 2\ (cons\ 3\ nil)))))$$

$$\rightarrow^*_\beta (cons\ 1\ (cons\ 2\ (cons\ 3\ nil)))$$

4. (3 pts) Considera la siguiente definición de un lenguaje de números naturales, cuya sintaxis se define con las siguientes reglas:

$$\frac{}{0\ \mathsf{Nat}} \qquad \frac{n\ \mathsf{Nat}}{(s\ n)\ \mathsf{Nat}} \qquad \frac{n\ \mathsf{Nat}}{(p\ n)\ \mathsf{Nat}}$$

en donde s y p definen las funciones sucesor y predecesor respectivamente. La semántica operacional del lenguaje se define como sigue:

$$\frac{}{(p\ (s\ n)) \rightarrow n} \qquad \frac{n \rightarrow n'}{(s\ n) \rightarrow (s\ n')} \qquad \frac{}{(p\ 0) \rightarrow 0}$$

a) Diseña una semántica estática para el lenguaje que defina un sistema de tipos para los naturales con los tipos Even y Odd, estos tipos nos indican si una expresión n del lenguaje define un número natural par o impar.

A continuación definimos la semántica estática para el lenguaje:

$$\frac{}{0 : \mathsf{Even}} \qquad \frac{\varnothing \vdash n : \mathsf{Even}}{\varnothing \vdash (s\ n) : \mathsf{Odd}} \qquad \frac{\varnothing \vdash n : \mathsf{Odd}}{\varnothing \vdash (s\ n) : \mathsf{Even}}$$

$$\frac{\varnothing \vdash n : \mathsf{Even}}{\varnothing \vdash (p\ n) : \mathsf{Odd}} \qquad \frac{\varnothing \vdash n : \mathsf{Odd}}{\varnothing \vdash (p\ n) : \mathsf{Even}} \qquad \frac{}{\varnothing \vdash (p\ 0) : \mathsf{Even}}$$

b) Con la semántica definida en el inciso anterior y la semántica operacional dada en el ejercicio, indica si el lenguaje cumple con ser seguro, es decir, si cumple las propiedades de: *unicidad*, *preservación* y *progreso*. No es necesario dar una demostración formal de las propiedades, sino una justificación del porqué se cumplen o no cada una de estas propiedades.

(1) *Unicidad:* Esta semántica nos dice que todos tienen un valor y es único. Por otro lado, la semántica dada no genera bucles, en la definición recursiva de números pares e impares, al seguir el proceso de obtención del predecesor, llegamos finalmente al cero, el cual sabemos que es un número par.

(2) *Preservación:* La semántica no causa ninguna modificación en los tipos, las acciones realizadas en números naturales siempre conducen a otro número natural como resultado, por lo que el tipo original se conserva, esto es, cada regla dada no cambia el tipo. Ya que si $(p\,(s\,n)) : Odd$ entonces por nuestras reglas $n : Odd$, de manera análoga para $Even$, el $0 : Even$ y $(p\ 0) : Even$ cumplen y por lo tanto con $n \to n'$ si $n : Odd$ entonces por lo visto antes $n' : Odd$, análogo para $Even$.

(3) *Progreso:* La mayoría son valores que ya tienen un tipo, en este caso el 0, que podemos ir procesando y por esto, llegar a su valor. Está claro que el cero es de tipo par. Si un número no es cero, su tipo se puede determinar de manera recursiva, y esto se aplica a los sucesores de un número, los cuales también tendrán un tipo definido. Por lo tanto, ningún número quedará en un estado indefinido, lo que confirma que la propiedad se cumple.