Paradigmas de la Programación
Primer Parcial

29 de abril de 2008

1. Considere el siguiente programa:

```
local
   X=2
   fun {XTimesY Y}
      X*Y
   end
in
   {Browse {XTimesY 3}}
end
```

   Traduzca el programa al lenguaje del kernel y ejecute manualm
   guiendo las reglas semánticas. ¿Qué resultados muestra el *browser*

2. Considere la función `filter` de Haskell:

```
filter :: (a -> Bool) -> [a] -> [a]
filter f [] = []
filter f (x:xs) = if f(x) then x : (filter f xs)
                  else filter f xs
```

   Implemente la función `{Filter F Xs}` en Oz utilizando recursi
   cola.

3. Suponga que modificamos la semántica del lenguaje del núcleo p
   tenga *scoping* dinámico. Para eso sólo hace falta modificar la se
   de la aplicación de procedimientos (págs. 66 y 67 del libro). La s
   semántica es:

$$(\{\langle x\rangle\ \langle y\rangle_1 \cdots \langle y\rangle_n\}, E)$$

   Si $E(\langle x\rangle)$ tiene la forma

$$(\texttt{proc}\ \{\$\ \langle z\rangle_1 \cdots \langle z\rangle_n\}\ \langle s\rangle\ \texttt{end}, CE)$$

   hay que poner en el stack

$$(\langle s \rangle, E + \{\langle z \rangle_1 \rightarrow E(\langle y \rangle_1), \ldots, \langle z \rangle_n \rightarrow E(\langle y \rangle_n)\})$$

Esto es igual que la semántica original pero cambiando "$CE+$" por "$E+$".

Ahora considere el siguiente código:

```
local Map F in
   fun {Map Xs}
      case Xs
      of nil then nil
      [] Y|Ys then {F Y} | {Map F Ys}
      end
   end
   fun {F X} X+2 end
   local F = fun {$ X} 2*X end in {Browse {Map [1 3 0]}} end
   local F = fun {$ X} 3*X end in {Browse {Map [1 3 0]}} end
end
```

Traduzca la definición al lenguaje del núcleo. Escriba una ejecución manual salteando los detalles. ¿Qué resultados muestra el *browser*?