Paradigmas de Programación
Parcial recuperatorio

Paradigmas 2007

20 de junio de 2007

Parte 1

1. Supongamos que (en el modelo declarativo), la ejecución de un programa llevo a la máquina abstracta al siguiente estado:

   $([(\langle s_1\rangle\ \langle s_2\rangle, E_2), (\langle s_1\rangle\ \langle s_2\rangle, E_1)], \{p = (\alpha, Q \rightarrow q_1), q_1 = (\beta, \emptyset), q_2 = (\gamma, \emptyset), x_1, x_2\})$

   Donde:
   $\langle s_1\rangle \equiv$ `X = 42`
   $\langle s_2\rangle \equiv$ `{P X}`
   $E_1 = \{P \rightarrow p, Q \rightarrow q_1, X \rightarrow x_1\}$
   $E_2 = \{P \rightarrow p, Q \rightarrow q_2, X \rightarrow x_2\}$
   $\alpha =$ **proc** $\{\$\ X\}\{Q\ X\}$ **end**
   $\beta =$ **proc** $\{\$\ X\}$ **local** A **in** $A = a(X)$ $\{Browse\ A\}$ **end end**
   $\gamma =$ **proc** $\{\$\ X\}$ **local** A **in** $A = b(X)$ $\{Browse\ A\}$ **end end**

   a) Continuar la ejecución de la máquina abstracta paso a paso, hasta que la primera instrucción en el stack sea una llamada a `Browse`. Mostrar el estado completo de la máquina abstracta en cada uno de los pasos.

   b) Dar un programa en lenguaje kernel cuya ejecución pueda llevar la máquina abstracta al estado mostrado.

2. Traducir a lenguaje kernel el siguiente fragmento de programa:

```
fun {F Xs}
   if Xs==nil then 0
   else local Xh|Xt=X in
      if Xh==0 orelse Xh < 0 then {F Xt} else 1+{F Xt}
   end
end
```

3. Recuerde la definición de las siguientes funciones.

```
fun {Iterate S IsDone Transform}
   if {IsDone S} then S
   else S1 in
      S1 = {Transform S}
      {Iterate S1 IsDone Transform}
   end
end
```

```
fun {FoldR L F U}
   case L
   of nil then U
   [] X|L1 then {F X {FoldR L1 F U}}
   end
end
```

a) Implemente `FoldR` en términos de `Iterate`.

b) ¿Para que casos se puede implementar `Iterate` en términos de `FoldR`?

4. Redefinamos la sentencia `try` $\langle S \rangle_1$ `catch` $\langle x \rangle_2$ `then` $\langle S \rangle_2$ en dos instrucciones distintas:

   - `try` $\langle S \rangle_1$
   - `catch` $\langle x \rangle_2$ `then` $\langle S \rangle_2$

   ¿Se puede dar una semántica para estas operaciones de manera que esta sea igual a la original? De la semántica o argumente en contra.

Parte 2

1. Considere la siguiente definición de colas usando diferencia de listas:

```
fun {NewQueue} X in q(X X) end

fun {Insert Q X}
   case Q of q(S E) then E1 in E=X|E1 q(S E1) end
end

fun {Delete Q X}
   case Q of q(S E) then S1 in S=X|S1 q(S1 E) end
end
```

   Ahora supongamos que se corre el siguiente fragmento de programa:

```
local Q A1 B1 A2 B2 A3 B3 in
   Q = {NewQueue}
   A1 = {Insert Q elemento}
   B1 = {Insert Q elemento}
   A2 = {Insert A1 otro}
   B2 = {Delete Q _}
   A3 = {Insert A2 otromas}
   B3 = {Insert B2 yotro}
end
```

   Describa en detalle como procede la ejecución; no hace falta hacerlo con la máquina abstracta, pero si que va pasando con las variables en cada paso.

2. La sucesión de Fibonacci aparece en varios ejemplos del libro y de los prácticos. Supongamos que escribimos el siguiente programa para calcular el elemento número 50 de esta sucesión:

```
local SumList Fib in
   fun {SumList Xs Ys}
      case Xs of nil then nil
      [] X|Xr then Y|Yr=Ys in
         (X+Y)|{SumList Xr Yr}
      end
   end

   Fib=0|1|{SumList Fib Fib.2}

   {Browse {List.nth Fib 50}}
end
```

   a) ¿Este programa funciona? ¿Por que? Si funciona, indicar la eficiencia (en notación $O(\ldots)$) del programa en función del índice de elemento que se calcula.

   b) Ahora modifiquemos la función `SumList` para que sea lazy. ¿Este nuevo programa funciona? ¿Por que? Nuevamente, si el programa funciona, indicar la eficiencia.

3. Observe el siguiente código:

```
thread X = {ByNeed fun {$} 3 end} end
thread X = Y end
thread if X==Y then Z=10 end end
```

a) al final de la ejecución esta Z determinado?

b) se dispara el trigger asociado a X?

c) si el trigger se dispara, elimine la instruccion que lo hace disparar.

d) si el trigger no se dispara, agregue una instruccion que lo haga disparar.

justifique las respuestas utilizando la semámtica de las intrucciones.

4. Describa una pila que donde el constructor de la pila tome un booleano que indique si la pila resultante tiene que ser segura o insegura respectivamente. Elija el resto de los atributos.