Lenguajes y Compiladores - Trabajo práctico 4 - Año 2006

(1) Demostrar que el funcional F en la definición de la semántica denotacional del **while** para el lenguaje con **fail** es continua.

(2) Demostrar o refutar las siguientes equivalencias usando semántica denotacional:
   (a) **while true do fail** $\equiv$ **skip**.
   (b) c; **fail** $\equiv$ **fail**.
   (c) **newvar** $v := e$ **in** $v := v + 1$; **fail** $\equiv$ **newvar** $w := e$ **in** $w := w + 1$; **fail**.
   (d) **while** b **do fail** $\equiv$ **if** b **then fail else skip**.

(3) Para el lenguaje con **fail** enunciar los Teoremas de Coincidencia, de Sustitución y de Renombre.

(4) Demostrar la equivalencia entre la semántica denotacional y la operacional incluyendo **fail** demostrando simultáneamente
   (a) $(c, \sigma) \rightarrow^+ \sigma' \Longrightarrow [\![c]\!]\sigma = \sigma'$.
   (b) $(c, \sigma) \rightarrow^+ (\mathbf{abort}, \sigma') \Longrightarrow [\![c]\!]\sigma = (\mathbf{abort}, \sigma')$.
   (c) $(c, \sigma) \uparrow \Longrightarrow [\![c]\!]\sigma = \perp$.
   por inducción en la estructura de c, utilizando que la semántica operacional es dirigida por sintaxis)

(5) Resolver los Ejercicios 5.3 (a), 5.4 (a) y 5.5 (usando la semántica directa) del libro de Reynolds.

(6) Demostrar las Proposiciones 5.1 y 5.2 del libro de Reynolds.

(7) Dado el programa $P \equiv$

```
newvar x := y + x in
        while x > 0 do
                ?x;
                y:= x+y;
                !(y-1);
                if x > 0 then fail else skip
```

   (a) Determinar estados σ y σ' tales que P equivalente a **skip** en σ y diferente a **skip** en σ'.
   (b) Computar la semántica operacional de P en ambos estados.
   (c) Calcular la semántica denotacional de P en ambos estados.

(8) Demostrar o refutar las siguientes equivalencias usando semántica denotacional:
   (a) $?x$; **while** b **do** $!x; ?x$ **od**; $!x \equiv$ **while** b **do** $?x; !$ **od**.
   (b) $?x$; **while** b **do** $!x; ?x$ **od**; $!x \equiv ?x; !x$; **while** b **do** $?x; !x$ **od**.
   (c) $?x; ?y \equiv ?y; ?x$.
   (d) $?x; z := x \equiv ?z$.

(e) **newvar** $x := e$ **in** $(?x; z := x) \equiv ?z$.
(f) **newvar** $x := e$ **in** $(?x; z := x; !e) \equiv ?z; !e$.

(9) Considere la representación gráfica de Ω dada al principio de la sección 5.5 del Reynolds. "Despliegue" el gráfico recursivo para poder señalar el nodo correspondiente a:
(a) $[\![!x; !(x+1); !(x+2)]\!]\sigma_0$
(b) $[\![!x; !(x+1); !(x+2);$ **while true do skip**$]\!]\sigma_0$
(c) $[\![!x; !(x+1); !(x+2);$ **fail**$]\!]\sigma_0$

(10) Definir la semántica denotacional de continuaciones para un lenguaje con **while**, input, output y **fail**.

(11) Calcule de manera detallada la semántica de continuaciones de $x := 0;$ **while** $x < 2$ **do** $?y; x := x + 1; !y$ **od**.

(12) Para el lenguaje imperativo simple sin **while** demostrar que para todo c, σ, k, $[\![c]\!]^c k\sigma = k([\![c]\!]^d\sigma)$.

(13) Para el lenguaje imperativo simple con **while** demostrar que para todo c, σ, k, $[\![c]\!]^c k\sigma = k_{\perp\!\!\perp}([\![c]\!]^d\sigma)$.

(14) (a) Para el lenguaje imperativo con input y output demostrar que para todo c, σ, k, $[\![c]\!]^c k\sigma = k_*([\![c]\!]^d\sigma)$.
(b) Vale la igualdad anterior si agregamos **fail**?
(c) Considere el lenguage de (a) con **fail**, y considere la semántica directa de variables locales que no recupera el estado original en caso de abortar. O sea, $(\ )_*$ en lugar de $(\ )_\dagger$:

$[\![$**newvar** $v := e$ **in** $c]\!]\sigma = (\lambda\sigma' \in \Sigma.[\sigma'|v : \sigma v])_* \ ([\![c]\!][\sigma|v : [\![e]\!]\sigma])$

Pruebe la igualdad de (a).

(15) Para el lenguaje imperativo con **fail** demostrar que para todo c, σ, k, k_f, $[\![c]\!]^c k k_f \sigma = (k, k_f)_*([\![c]\!]^d\sigma)$.

(16) Dado el programa $P \equiv$
while $x \neq 0$ **do if** $x = 1$ **then fail else** $x := x - 2$
(a) Calcular la semántica directa de P en un estado σ tal que $\sigma\ x = n$.
(b) Calcular la semántica de continuaciones de P en un estado σ tal que $\sigma\ x = n$, con las continuaciones k y k_f.

(17) Resolver los Ejercicios 5.3 (b), 5.4 (b) y 5.5 (usando la semántica de continuaciones) del libro de Reynolds.

(18) Para el lenguaje con **while**, input, output, **fail** l y **catch** l **in** c_0 **with** c_1 enunciar la correspondencia entre la semántica directa y de continuaciones.

(19) Para este lenguaje, dar la semántica de transiciones.

(20) Agregar a este lenguaje el comando **repeat** definiendo su semántica directa, de continuaciones y de transiciones.

(21) Agregar a este lenguaje el comando **for** y dar su semántica directa, de continuaciones y de transiciones.

(22) Para este lenguaje, agregar un comando **break** que cuando se ejecuta fuerza la salida del cuerpo del **while** más próximo en que se encuentre (no sale de varios anidados). Dar la semántica de continuaciones.

(23) Modificar la semántica de **break** para que:
   (a) fuerce la salida del **repeat** también.
   (b) fuerce la salida del **for** también.
   (c) fuerce la salida del **newvar** también.

(24) Probar o refutar usando semántica de continuaciones la equivalencia:

$$\textbf{repeat}\ c\ \textbf{until}\ b \equiv_{\textbf{comm}} c; \textbf{while not}\ b\ \textbf{do}\ c.$$