Hoja II.6

SOLUCIÓN DEL EJERCICIO 1

El dominio de la función universal para las funciones while-computables unarias es $\{ (\mathbf{x},\mathbf{y}): \varphi_{\mathbf{x}}(\mathbf{y})\downarrow \}$

SOLUCION DEL EJERCICIO 2

a) $\{ \mathbf{x}: \Phi(0,\mathbf{x})\downarrow \} = \sum^*$

b) $\{ \mathbf{x}: \exists \mathbf{y}\ \Phi(\mathbf{x},\mathbf{y})\downarrow \} = \overline{\mathrm{VAC}}$

c) $\{ \mathbf{x}: \Phi(\mathbf{x},\mathbf{x}) = \varphi_{\mathbf{x}}(\mathbf{x}) \} = \mathbf{K}$

d) $\{ \mathbf{x}: \forall \mathbf{y}\ \Phi(\mathbf{y},\mathbf{x})\downarrow \} = \mathrm{TOT}$

e) $\{ \mathbf{x}: \exists \mathbf{y}\ \Phi(\mathbf{y},\mathbf{x})\downarrow \} = \sum^*$

f) $\{ \mathbf{x}: \exists \mathbf{z}\ \forall \mathbf{y}\ \Phi(\mathbf{x},\mathbf{y}) = \mathbf{z} \} = \mathrm{CONS}$

g) $\{ \mathbf{x}: \forall \mathbf{y}\ \Phi(\mathbf{y},\mathbf{x})\downarrow \} = \varnothing$

h) $\{ \mathbf{x}: \forall \mathbf{y}\ \Phi(\mathbf{y},\mathbf{x})\uparrow \} = \varnothing$

ALGUNAS SOLUCIONES DEL EJERCICIO 4

a) $f(\mathbf{x}) = c_K(\mathbf{x})$

b) $g(\mathbf{x}) = \mathrm{false}$

e) $h(\mathbf{x}) = \mathrm{true}$

Hoja II.6

SOLUCIÓN DEL EJERCICIO 5a

Para computar ψ simularemos la ejecución del programa $\mathbf{P_x}$ sobre el dato **y**, comprobando tras cada paso si el valor de la variable XJ ha sido alterado. Utilizamos el mismo método de simulación que el de la función universal, con el vector de estado VARS y la pila de procesos en ejecución PILA. La variable VALOR_XJ nos sirve para comparar el valor de la variable XJ antes y después de la ejecución de cada instrucción:

```
VARS := (ε);  VARS(ult_variable(X1)) := ε;  VARS (1) := X2;
X0 := false; PILA := empilar(X1, <]);   VALOR_XJ := VARS (X3);
while  not X0 loop
      PROG := cima(PILA); PILA := desempilar(PILA);
      if asig_vacía?(PROG) then  VARS (ind_var(PROG)) := ε;
      elsif asig_cons?( PROG) then
            VARS (ind_var(PROG)) :=
                  ind_sim(PROG) & VARS (ind_var2(PROG));
      elsif asig_cdr?( PROG) then
            VARS (ind_var(PROG)) := cdr(VARS (ind_var2(PROG)));
      elsif composición?( PROG) then
            PILA := empilar(inst_int(PROG),
                        empilar(inst_int2(PROG), PILA));
      elsif condición?( PROG) then
            if ind_sim(PROG) = primero(VARS(ind_var(PROG))) then
                  PILA := empilar(inst_int(PROG), PILA);
            end if;
      else
            if nonem?(VARS(ind_var(PROG)))  then
                  PILA := empilar(PROG, PILA);
                  PILA := empilar(inst_int(PROG), PILA);
            end if;
      end if;
      X0 := VARS (X3) /= VALOR_XJ;
end loop;
```

Nótese que si PILA se termina sin que X0 se haya convertido en true el programa también cicla (como es debido) al intentar calcular la cima de una pila vacía.

Hoja II.6

SOLUCIÓN DEL EJERCICIO 5b

Para computar χ simularemos la ejecución del programa $\mathbf{P_x}$ sobre el dato **y**, comprobando si XJ aparece en el lado derecho de una asignación o en una condición. Utilizamos el mismo método de simulación que el de la función universal, con el vector de estado VARS y la pila de procesos en ejecución PILA.

```
VARS := (ε);  VARS(ult_variable(X1)) := ε;  VARS (1) := X2;
X0 := false; PILA := empilar(X1, <]);
while  not X0 loop
      PROG := cima(PILA); PILA := desempilar(PILA);
      if asig_vacía?(PROG) then  VARS (ind_var(PROG)) := ε;
      elsif asig_cons?( PROG) then
            VARS (ind_var(PROG)) :=
                  ind_sim(PROG) & VARS (ind_var2(PROG));
            X0:=ind_var2(PROG) = X3;
      elsif asig_cdr?( PROG) then
            VARS (ind_var(PROG)) := cdr(VARS (ind_var2(PROG)));
            X0:=ind_var2(PROG) = X3;
      elsif composición?( PROG) then
            PILA := empilar(inst_int(PROG),
                        empilar(inst_int2(PROG), PILA));
      elsif condición?( PROG) then
            if ind_sim(PROG) = primero(VARS(ind_var(PROG))) then
                  PILA := empilar(inst_int(PROG), PILA);
            end if;
            X0:=ind_var(PROG) = X3;
      else
            if nonem?(VARS(ind_var(PROG)))  then
                  PILA := empilar(PROG, PILA);
                  PILA := empilar(inst_int(PROG), PILA);
            end if;
            X0:= ind_var(PROG) = X3;
      end if;
end loop;
```

Nótese que si PILA se termina sin que X0 se haya convertido en true el programa también cicla (como es debido) al intentar calcular la cima de una pila vacía.

Hoja II.6

SOLUCIÓN DEL EJERCICIO 5c

Simularemos la ejecución de $\mathbf{P_x}$ intentando verificar la condición pedida sobre XJ. Si esta se cumple, abortamos la ejecución y devolvemos true. Si la ejecución termina sin que **XJ** haya contenido nunca los valores indicados, ciclaremos voluntariamente. Por último, si el programa cicla sin cumplirse nunca la condición no tendremos que hacer nada especial.Utilizamos el mismo método de simulación que el de la función universal, con el vector de estado VARS y la pila de procesos en ejecución PILA.

```
VARS:= ε; VARS(ult_variable(X1)):= ε; VARS(1):= X2;
PILA:= <]; PILA:= empilar (X1, PILA); X0:= false;
if VARS(X3) = X4 then X0:= true;                    -- Por si la condición se cumple al principio
else while not X0 loop
     PROG:= cima(PILA); PILA:= desempilar(PILA);
     if asig_vacía?(PROG) then VARS(ind_var(PROG)):= ε;
     elsif asig_cons?(PROG) then
          VARS(ind_var(PROG)):= ind_simb(PROG)•VARS(ind_var_2(PROG));
     elsif asig_cdr?(PROG) then
          VARS(ind_var(PROG)):=cdr(VARS(ind_var_2(PROG)));
     elsif composición?(PROG) then
          PILA := empilar(inst_int(PROG), empilar(inst_int_2(PROG), PILA));
     elsif condición?(PROG) then
          if ind_simb(PROG) = primero(VARS(ind_var(PROG))) then
          PILA := empilar(inst_int(PROG), PILA); end if;
     else
          if nonem?(VARS(ind_var(PROG))) then
          PILA := empilar(inst_int(PROG), empilar(PROG, PILA)); end if;
     end if;
  -- Pero comprobando si XJ llega a contener z
     X0:= VARS(X3)=X4;

     end loop;

end if;
```

Hoja II.6

SOLUCIÓN DEL EJERCICIO 5d

Simulamos la ejecución de $\mathbf{P_x}$ contando el nº de veces que se ejecuta cualquiera de sus *whiles*. Como puede contener un nº arbitrario de iteraciones anidadas, deberemos utilizar un nº indeterminado de contadores para controlar cada uno de los bucles abiertos en un momento dado. Otro problema es cómo asociar un *while* con un contador concreto, teniendo en cuenta que puede haber varios bucles abiertos que afecten a la misma variable. Lo resolvemos enriqueciendo nuestro esquema de simulación de ejecuciones: la información asociada a un *while* se almacenará junto con él en la pila, para que esté disponible cuando sea necesario. Tendremos las estructuras VARS y PILA, pero esta última contendrá pares de la forma (**P**, **n**), donde **P** es un programa-while que se debe ejecutar y **n** representa, cuando **P** es un *while*, el número de iteraciones de su instrucción interna que ya se habrán ejecutado cuando P sea desempilado. Si **P** no es un *while*, su valor será siempre 0.

```
VARS := (ℰ); VARS(ult_variable(X1)) := ℰ; VARS (1) := X2;
PILA := empilar(cod_2(X1, 0), <])); X0 := 0;
while not P_vacía?(PILA) loop
      PAR := cima(PILA); PILA := desempilar(PILA);
      PROG := decod_2_1(PAR); NITER := decod_2_2(PAR);
      if asig_vacía?(PROG) then VARS (ind_var(PROG)) := ℰ;
      elsif asig_cons?(PROG) then
            VARS (ind_var(PROG)) := ind_sim(PROG) & VARS (ind_var2 (PROG));
      elsif asig_cdr?(PROG) then
            VARS (ind_var(PROG)) := cdr(VARS (ind_var2(PROG)));
      elsif composición?(PROG) then
            PILA := empilar(cod_2(inst_int(PROG), 0),
                              empilar(cod_2(inst_int_2(PROG), 0), PILA));
      elsif condición?(PROG) then
            SIMB := ind_sim(PROG); X_I:= VARS(ind_var(PROG));
            if SIMB = primero(X_I) then
                  PILA := empilar(cod_2(inst_int(PROG), 0), PILA);
            end if;
      else X0 := max (NITER, X0);   X_I:= VARS(ind_var(PROG))
            if nonem?(X_I) then
                  PILA := empilar(cod_2(PROG, succ(NITER)), PILA);
                  PILA := empilar(cod_2(inst_int(PROG), 0), PILA);
            end if;
      end if;
end loop;
```

Obsérvese que cuando la ejecución del programa $\mathbf{P_x}$ sobre el dato **y** no termine nuestro programa tampoco terminará, pero ello es coherente con la definición de τ, ya que para que el programa cicle es imprescindible que un bucle se ejecute un número infinito de veces. Como parece lo más razonable, no hemos considerado conjuntamente las iteraciones no consecutivas de un bucle.

Hoja II.6

SOLUCIÓN DEL EJERCICIO 5e

Para computar ξ simularemos la ejecución del programa $\mathbf{P_x}$ sobre el dato **y**, contabilizando en X0 cada vez que ejecutamos el cuerpo de un bucle. Utilizamos el mismo método de simulación que el de la función universal, con el vector de estado VARS y la pila de procesos en ejecución PILA.

```
VARS := (ε);  VARS(ult_variable(X1)) := ε;  VARS (1) := X2;
X0 := 0; PILA := empilar(X1, <]);
while  not X0 loop
      PROG := cima(PILA); PILA := desempilar(PILA);
      if asig_vacía?(PROG) then  VARS (ind_var(PROG)) := ε;
      elsif asig_cons?( PROG) then
            VARS (ind_var(PROG)) :=
                  ind_sim(PROG) & VARS (ind_var2(PROG));
      elsif asig_cdr?( PROG) then
            VARS (ind_var(PROG)) := cdr(VARS (ind_var2(PROG)));
      elsif composición?( PROG) then
            PILA := empilar(inst_int(PROG),
                        empilar(inst_int2(PROG), PILA));
      elsif condición?( PROG) then
            if ind_sim(PROG) = primero(VARS(ind_var(PROG))) then
                  PILA := empilar(inst_int(PROG), PILA);
            end if;
      else
            if nonem?(VARS(ind_var(PROG)))  then
                  PILA := empilar(PROG, PILA);
                  PILA := empilar(inst_int(PROG), PILA);
                  X0:=succ(X0);
            end if;
      end if;
end loop;
```

Hoja II.6

SOLUCIÓN DEL EJERCICIO 5f

Para computar σ simularemos la ejecución del programa $\mathbf{P_x}$ sobre el dato **y**, comprobando tras cada paso si la variable XJ ha alcanzado un valor mayor al acumulado en X0. Utilizamos el mismo método de simulación que el de la función universal, con el vector de estado VARS y la pila de procesos en ejecución PILA pero comprobamos primero que la variable XJ interviene en el programa X1:

```
VARS:= ε; VARS(ult_variable(X1)):= ε; VARS(1):= X2;
PILA:= <]; PILA:= empilar (X1, PILA); X0:= ε;
if X3> ult_variable(X1) then X0:= ⊥⊥;
else while not p_vacia?(PILA) loop
      PROG:= cima(PILA); PILA:= desempilar(PILA);
      if asig_vacía?(PROG) then VARS(ind_var(PROG)):= ε;
      elsif asig_cons?(PROG) then
            VARS(ind_var(PROG)):= ind_simb(PROG)•VARS(ind_var_2(PROG));
      elsif asig_cdr?(PROG) then
            VARS(ind_var(PROG)):=cdr(VARS(ind_var_2(PROG)));
      elsif composición?(PROG) then
            PILA := empilar(inst_int(PROG), empilar(inst_int_2(PROG), PILA));
      elsif condición?(PROG) then
            if ind_simb(PROG) = primero(VARS(ind_var(PROG))) then
                  PILA := empilar(inst_int(PROG), PILA); end if;
      else
            if nonem?(VARS(ind_var(PROG))) then
            PILA := empilar(inst_int(PROG), empilar(PROG, PILA)); end if;
      end if;
      if X0 > VARS(X3) then X0:= VARS(X3); end if;
      end loop;
end if;
```

Hoja II.6

SOLUCIÓN DEL EJERCICIO 5g

Para computar η simularemos la ejecución del programa $\mathbf{P_x}$ sobre el dato **y**, apuntando en un vector CONT las veces que se actualiza cada variable XI. Con cada modificación de XI sumamos 1 en CONT(I) . Utilizamos el mismo método de simulación que el de la función universal, con el vector de estado VARS y la pila de procesos en ejecución:

```
VARS:= ε; VARS(ult_variable(X1)):= ε; VARS(1):= X2;
CONT:= ε; CONT(ult_variable(X1)):= ε;
PILA:= <]; PILA:= empilar (X1, PILA);
while not p_vacía?(PILA) loop
      PROG:= cima(PILA); PILA:= desempilar(PILA);
      if asig_vacía?(PROG) then VARS(ind_var(PROG)):= ε;
      elsif asig_cons?(PROG) then
            VARS(ind_var(PROG)):= ind_simb(PROG)•VARS(ind_var_2(PROG));
      elsif asig_cdr?(PROG) then
            VARS(ind_var(PROG)):=cdr(VARS(ind_var_2(PROG)));
      elsif composición?(PROG) then
            PILA := empilar(inst_int(PROG), empilar(inst_int_2(PROG), PILA));
      elsif condición?(PROG) then
            if ind_simb(PROG) = primero(VARS(ind_var(PROG))) then
                  PILA := empilar(inst_int(PROG), PILA); end if;
      else
            if nonem?(VARS(ind_var(PROG))) then
                  PILA := empilar(inst_int(PROG), empilar(PROG, PILA)); end if;
      end if;
      if asig_vacía?(PROG) or asig_cons?(PROG) or asig_cdr?(PROG) then
            CONT(ind_var(PROG)):= CONT(ind_var(PROG))+1;
      end if;
      end loop;
end if;
X0:=0;
for I in 0..ult_indice(CONT) loop
      if CONT(I) > X0 then X0:= I; end if;
end loop;
```

Hoja II.6

SOLUCIÓN DEL EJERCICIO 5h

```
VARS := (ε);  VARS(ult_variable(X1)) := ε;  VARS (1) := X2;
FIN := false; PILA := empilar(X1, <]); X0=:false;
while  not FIN loop
      PROG := cima(PILA); PILA := desempilar(PILA);
      if asig_vacía?(PROG) then
            VARS (ind_var(PROG)) := ε;
            FIN:= ind_var(PROG) = X3; X0:= true;
      elsif asig_cons?( PROG) then
            VARS (ind_var(PROG)) :=
                  ind_sim(PROG) & VARS (ind_var2(PROG));
            FIN:=ind_var2(PROG) = X3 or ind_var(PROG) = X3;
            X0:=ind_var(PROG) = X3 and not ind_var2(PROG) = X3;
      elsif asig_cdr?( PROG) then
            VARS (ind_var(PROG)) := cdr(VARS (ind_var2(PROG)));
            X0:=ind_var2(PROG) = X3;
            FIN:=ind_var2(PROG) = X3 or ind_var(PROG) = X3;
            X0:=ind_var(PROG) = X3 and not ind_var2(PROG) = X3;
      elsif composición?( PROG) then
            PILA := empilar(inst_int(PROG),
                        empilar(inst_int2(PROG), PILA));
      elsif condición?( PROG) then
            if ind_sim(PROG) = primero(VARS(ind_var(PROG))) then
                  PILA := empilar(inst_int(PROG), PILA);
            end if;
            FIN:=ind_var(PROG) = X3;
      else
            if nonem?(VARS(ind_var(PROG)))  then
                  PILA := empilar(PROG, PILA);
                  PILA := empilar(inst_int(PROG), PILA);
            end if;
            FIN:=ind_var(PROG) = X3;
      end if;
end loop;
```

Hoja II.6

SOLUCIÓN DEL EJERCICIO 5i

Dado que tendremos que simular la ejecución del programa $\mathbf{P_x}$ sobre el dato **y**, utilizaremos una técnica similar a la del programa universal, con una pila PILA para los procesos pendientes de ejecución y un vector VARS para almacenar el contenido de las variables de $\mathbf{P_x}$.

Naturalmente, será necesario variar tanto la condición de terminación (usaremos una variable FIN para salir del bucle de ejecución) como el resultado proporcionado (que será siempre true). Si durante la ejecución de $\mathbf{P_x}$ sobre el dato **y** en algún momento se alcanza el valor '**abb**' para la variable X**N**, el resultado debe ser true, independientemente de si $\mathbf{P_x}$ termina o no. Lo conseguiremos comprobando si X**N**='**abb**' al final de cada iteración (es decir, tras la ejecución de cada instrucción individual), y forzaremos la terminación mediante FIN en cuanto este hecho se produzca. El programa será:

```
VARS(ult_variable (X1)) := ε; VARS(1) := X2;
PILA := empilar (X1, <]); FIN := false;
if  X3 <> 1  or  X2<>'abb'  then
      while  not FIN  loop
            PROG := cima (PILA); PILA  := desempilar (PILA);
            if  asig_vacía? (PROG)  then
                  VARS(ind_var(PROG)) := ε;
            elsif  asig_cons? (PROG)  then
                  VARS(ind_var(PROG)) :=
                        ind_simb (PROG) & VARS(ind_var2 (PROG));
            elsif  asig_cdr? (PROG)  then
                  VARS(ind_var(PROG)) := cdr (VARS(ind_var2 (PROG)));
            elsif  composición?(PROG)  then
                  PILA := empilar (inst_int (PROG), empilar (inst_int2 (PROG), PILA));
            elsif  condición? (PROG)  then
                  if  ind_simb(PROG) = primero (VARS(ind_var (PROG)))  then
                        PILA := empilar (inst_int (PROG), PILA);
                  end if;
            else
                  if  nonem? (VARS(ind_var (PROG)))  then
                        PILA := empilar (inst_int (PROG), empilar (PROG, PILA));
                  end if;
            end if;
            FIN := VARS(X3) = 'abb';
      end loop;
end if;
X0 := true;
```

El primer condicional tiene como función detectar el que la condición se cumpla antes de empezar a ejecutar (es decir, si **n**=1 y el dato de entrada **y** es justamente '**abb**').

Comprobemos, además, que el programa se comporta correctamente también en los casos en que el resultado debe quedar indefinido:

Si **n**>ult_variable(**P_x**) X**N** nunca alcanzará el valor '**abb**'. La primera vez que se entra en el while la evaluación de la expresión *VARS(X3)* supondrá acceder a una posición inexistente del vector.

La ejecución de **P_x** termina sin que X**N** haya tomado nunca el valor '**abb**'. Entonces PILA se vaciará, pero FIN no cambiará y no se provocará la salida del bucle, por lo que al intentar calcular *cima(PILA)* en la siguiente iteración se producirá el ciclado deseado.

La ejecución de **P_x** se prolonga indefinidamente sin que X**N** tome nunca el valor '**abb**'. Entonces nunca se sale del bucle.