Ejercicios

I.2.LOS PROGRAMAS-WHILE

PRERREQUISITOS:

- Conocer la definición y entender el funcionamiento de los programas-while a través de cómputos
- Entender la relación entre un programa y la función computada por el mismo (o funciones, según el número de argumentos)
- Conocer la definición de función while-computable y predicado while-decidible

PROBLEMAS:

A. PARA FAMILIARIZARSE CON LAS PARTICULARIDADES SINTÁCTICAS DE LOS PROGRAMAS-WHILE

1* Estos programas-while no son correctos sintácticamente. Señala los errores sintácticos en cada caso.

a)
X3 := ε;
while X2 $\neq \varepsilon$ **loop**
    X2 := cdr(X2);
end loop;

b)
X3 := '**abc**';
while nonem?(X3) **loop**
    X3 := cdr(X3);
end loop;
X2 := $cons_a(\varepsilon)$;

c)
X3 := cdr($cons_a$(X2));
if car_ε?(X3) **then**
    X2 := X5;
    X2 := $cons_a$(X1);
    X1 := $cons_a$(X2);
end if;

d)
X3 := $cons_a$(X1);
if car_a?(X3) **then** **end if**;

e)
X3 := $cons_a$(X3);
if car_a?(X4)
    { X4 := cdr(X3);
      X2 := $cons_a$(X1); }
else { X4 := cdr(X1);
      X2 := $cons_b$(X0); }

f)
X3 := $cons_a$(X1);
if nonem?(X3) **then**
    X0 := $cons_a$(X3);
end if;
X3 := car_a(X1);

g)
X4 := '**abc**';
AUX := $cons_a$(X4)•X2;

h)
```
X2 := cons_a(X1);
while X2 ≠ 'abc' loop
    X0 := car_a(X5);
    X2 := cdr(X2);
    X3 := cons_a(X2);
end loop;
```

i)
```
X3 := ε;
X3 := cons_a(X3);
if X1 = a•x then
    X3 := cons_b(X3);
end if;
```

j)
```
X2 :=ε;
X2 := cons_a(X2);
X3 := X2;
```

k)
```
while car_a?(X4) loop
    X3 := cons_a(X3);
    if car_b?(X2) then
        X2 := cdr(X2);
        X3 := cons_a(X5, X3);
        X4 := cdr(X4);
    end if;
end loop;
```

B. PARA FAMILIARIZARSE CON LA NOCIÓN DE CÓMPUTO Y COMPUTABILIDAD

2. Sea el siguiente programa while:

```
X0 := cons_a(X1);
while nonem?(X1) loop
    X1 := cdr(X1);
    X4 := cons_a(X4);
    if car_a?(X2) then
        X2 := cdr(X2);
        X4 := ε;
        while nonem?(X3) loop
            X3 := cons_a(X1);
        end loop;
    end if;
    if car_a?(X4) then
        X0 := cons_b(X0);
    end if;
end loop;
```

a) Con cada uno de estos estados de cómputo inicial V_0, ¿cuál sería el estado de cómputo final?

- V_0 = (**aaa**, $\mathcal{E}$, **bb**, $\mathcal{E}$, $\mathcal{E}$)
- V_0 = (**abba**, **b**, $\mathcal{E}$, $\mathcal{E}$, $\mathcal{E}$)
- V_0 = (**b**, **bb**, **baa**, **b**, **a**)
- V_0 = ($\mathcal{E}$, **bbba**, **aaa**, **b**, **ba**)

b) Para que el cómputo sea divergente o infinito, ¿qué condiciones tiene que cumplir el estado de cómputo inicial?

3. Razona sobre la veracidad o falsedad de los siguientes enunciados:

a) Sea ψ una función de un argumento calculada por un programa-while de la forma: **if** $car_C?(X0)$ **then** P **end if**; . Entonces la función ψ es total.

b) Sea **f** una función constante. Dado que **f** es while-computable, su inversa $\mathbf{f^{-1}}$ también es while-computable.

c) Sea ψ cualquier función finita. Entonces ψ es while-computable

C. PARA APRENDER A OBTENER LA FUNCIÓN COMPUTADA POR UN PROGRAMA-WHILE

4. Sea al siguiente programa-while **P** sobre el alfabeto $\Sigma = \{\mathbf{a},\mathbf{b}\}$:

```
while nonem?(X2) loop
    while nonem?(X3) loop
        if car_a?(X2) then X2 := ε; end if;
        if car_b?(X2) then X1 := cons_b(X1); end if;
        X3 := cdr(X3);
    end loop;
end loop;
X0 := cons_a(X1);
X0 := cons_a(X1);
```

Comprueba que las funciones computadas por **P** para una o dos entradas son efectivamente las siguientes:

$\varphi_{\mathbf{P}}^{1}(\mathbf{x}) = \mathbf{a}\bullet\mathbf{x}$

$$\varphi_{\mathbf{P}}^{2}(\mathbf{x},\mathbf{y}) \cong \begin{cases} \mathbf{a}\bullet\mathbf{x} & \mathbf{y} = \varepsilon \\ \perp & \text{c.c.} \end{cases}$$

Calcula la función computada por P para tres entradas.

¿En qué varía la función computada si se le suministran a **P** más de tres entradas?

5. ¿Cuál sería la definición de las funciones $\varphi_{\mathbf{P}}^{1}$, $\varphi_{\mathbf{P}}^{2}$ y $\varphi_{\mathbf{P}}^{3}$ computadas por el programa-while del ejercicio **2**?

D. PARA ACOSTUMBRARSE A PROGRAMAR MEDIANTE LOS PROGRAMAS-WHILE

6. Demuestra que las siguientes funciones sobre el alfabeto $\Sigma = \{\mathbf{a},\mathbf{b},\mathbf{c}\}$ son while-computables:

a) $\mathbf{f}(\mathbf{x})$ = la palabra que resulta de rotar los símbolos de **x** (**a**'s por **b**'s, **b**'s por **c**'s y **c**'s por **a**'s)

b) $\mathbf{g}(\mathbf{x},\mathbf{y}) = \mathbf{x}\bullet\mathbf{y}\bullet\mathbf{x}$

c) $\mathbf{h}(\mathbf{x}) = \mathbf{a}^{|\mathbf{x}|_\mathbf{b}+2*|\mathbf{x}|_\mathbf{c}}$

d) $\mathbf{k}(\mathbf{x})$ = la palabra que resulta de sustituir en $\mathbf{x}$ cada aparición de **cba** por **c**

e) $\mathbf{f'}(\mathbf{x})$ = la palabra que resulta de eliminar de $\mathbf{x}$ todos los símbolos aislados (es decir, que no tienen otro igual a la derecha ni a la izquierda)

f) $\psi(\mathbf{x},\mathbf{y}) \cong \begin{cases} \mathbf{x} & \text{nonem?}(\mathbf{x}) \\ \mathbf{y} & \mathbf{x} = \varepsilon \wedge \text{nonem?}(\mathbf{y}) \\ \perp & \text{c.c.} \end{cases}$

g) $\chi(\mathbf{x},\mathbf{y}) \cong \begin{cases} \mathbf{x} & \text{prefijo?}(\mathbf{abb}, \mathbf{x}) \\ \mathbf{x}^R & \text{prefijo?}(\mathbf{abc}, \mathbf{x}) \\ \perp & \text{c.c.} \end{cases}$

h) $f(\mathbf{x})$ = palabra que resulta de duplicar uno a uno todos los símbolos de $\mathbf{x}$

i) $\xi(\mathbf{x},\mathbf{y}) \cong \begin{cases} \mathbf{a} & |\mathbf{x}| \bmod 2 = 0 \wedge |\mathbf{y}| \bmod 2 = 0 \\ \mathbf{y} & |\mathbf{x}| \bmod 2 \neq 0 \\ \perp & \text{c.c.} \end{cases}$

7. Demuestra que los siguientes predicados, definidos sobre el alfabeto $\Sigma = \{\mathbf{a},\mathbf{b}\}$, son while-decidibles:

a) $P(\mathbf{x}) \Leftrightarrow \mathbf{x}$ contiene cuatro símbolos repetidos seguidos

b) $Q(\mathbf{x}, \mathbf{y}) \Leftrightarrow |\mathbf{x}|_\mathbf{a} \leq |\mathbf{x}|_\mathbf{b}$

c) $R(\mathbf{x}) \Leftrightarrow \mathbf{x} \in \{ \mathbf{a}^\mathbf{n}\mathbf{b}^\mathbf{m} : \mathbf{n}, \mathbf{m} \geq 2 \}$

d) $S(\mathbf{x}) \Leftrightarrow \mathbf{x} = \mathbf{x}^R$

e) $T(\mathbf{x}, \mathbf{y}, \mathbf{z}) \Leftrightarrow \mathbf{x} = \mathbf{y}\bullet\mathbf{z}$

f) $P'(\mathbf{x}) \Leftrightarrow \exists\mathbf{n}\exists\mathbf{m}\ (\mathbf{m},\mathbf{n}>\mathbf{0} \wedge (\mathbf{x}=(\mathbf{ab})^\mathbf{n}\bullet\mathbf{aaa}\bullet(\mathbf{ab})^\mathbf{m})$

g) $Q'(\mathbf{x}) \Leftrightarrow \exists\mathbf{y}\ (\mathbf{x}=\mathbf{y}\bullet\mathbf{y})$

h) $R'(\mathbf{x}, \mathbf{y}) \Leftrightarrow |\mathbf{x}|=3*|\mathbf{y}|_\mathbf{a}$

8. Sea el alfabeto $\Sigma = \{\mathbf{0},\mathbf{1}\}$. Definimos la función $f(\mathbf{x})$, que devuelve la palabra que representa el número binario anterior al representado por $\mathbf{x}$. Demuestra que es while-computable

E. PARA RAZONAR SOBRE LA MINIMALIDAD DE LOS PROGRAMAS-WHILE

9. Aunque hemos dicho que queríamos manejar un lenguaje de programación lo más sencillo posible, en realidad el lenguaje de los programas-while se puede simplificar aún más. Demuestra que:

a) Aunque eliminásemos de la definición las instrucciones de la forma **XI:=$\mathcal{E}$;** podríamos computar exactamente las mismas funciones.

b) Análogamente, que se podría prescindir de la instrucción **if_then** si se permitiera a cambio el uso de los predicados $\mathbf{car_s?}$ en los bucles **while_loop**