Ejercicios

I.3. MACROPROGRAMAS

PRERREQUISITOS:

- ❑ Entender los concepto de macro y macroprograma
- ❑ Distinguir claramente entre macroprogramas y programas-while
- ❑ Conocer los distintos tipos de macros y su uso

PROBLEMAS:

A. PARA ENTENDER LA SINTAXIS DE LAS MACROS Y EL CONCEPTO DE EXPANSIÓN DE UNA MACRO

1. Señala las distintas macros que aparecen en los siguientes macroprogramas, el tipo al que pertenecen, e indica someramente en qué orden debería producirse su expansión para obtener el programa while equivalente:

a)
```
CONT := cons_a (X1);
while nonem?(CONT) loop
    CONT := sig(CONT & X3);
    X1 := cdr(X1);
end loop;
```

b)
```
if nonem?(AUX) then
    LAG := cdr(cons_a(AUX));
end if;
```

c)
```
AUX := 'abab'
X0 := CONT^R & AUX;
```

d)
```
AUX := 'abbab';  X0 := ε;
while prefijo?(X0, cdr(X1)&B)
    loop
        X1 := cdr(LAG);
        X0 := cdr(X0);
    end loop;
```

2. Escribe la expansión de los siguientes macroprogramas sobre el alfabeto {a, b}

a)
```
X1 := cdr(X1);
if  car_a?(cdr(X1))  then
    X0 := cons_b(X1);
else
    X1 := cdr(X1);
    X0 := cons_a(X1);
end if;
```

b)
```
AUX:= 'abb';
X0 := cons_a(cdr(AUX));
```

c)
```
X3:= primeros3símbolos(X1);
X0 := X3^R;
```

d)
```
SIM := primero(X1);
if nonem?(SIM) then
    X0:= cons_a(cons_b(SIM));
end if;
```

3. Definimos la macro-instrucción `break` que nos permite abandonar el bucle en el que aparece cuando se verifica una condición B al alcanzar dicha instrucción. La sintaxis de la macro sería la siguiente

```
while nonem?(XI) loop
P
if B(V1, ..., VK) then break; end if;
Q
end loop;
```

donde P y Q son programas-while (por simplificar), B(V1, ..., VK) es una macrocondición y V1, …, VK macrovariables. Escribe la expansión de la macroinstrucción `break`, utilizando el alfabeto $\Sigma = \{a, b, c\}$

B. PARA ACOSTUMBRARSE A PROGRAMAR CON MACROS

4. Demuestra que las siguientes funciones y predicados sobre el alfabeto $\Sigma = \{a,b,c\}$ son while-computables o while-decidibles, diseñando el macroprograma correspondiente. Dicho macroprograma sólo podrá contener macrovariables, macroexpresiones y macrocondiciones de un solo argumento:

   a) $\psi(x) \cong \begin{cases} u & \exists v \in L((bbb \cup c)^*)\,(x = u \bullet a \bullet v) \\ \perp & \text{c.c.} \end{cases}$

   b) $R(x,y,z) \Leftrightarrow u \bullet y = z^R$ donde u se obtiene eliminando las a's de x.

   c) $f(x,y) = \begin{cases} u & \exists z \in \Sigma^+ \,(x = y^R \bullet z \wedge \text{"u es el resultado de eliminar las c's de z e intercambiar sus a's por b's"}) \\ \varepsilon & \text{c.c.} \end{cases}$

   d) $\chi(x) \cong \begin{cases} u^3 & |u| = 1 \wedge \exists y,z \in \Sigma^* \,(x = y \bullet u \bullet z \wedge |y| = |z|) \\ \perp & \text{c.c.} \end{cases}$

   e) $g(x,y) = a^{|x|_{aaa}+|y|_b}$

   f) $Q(x) \Leftrightarrow x \in \{a^i b^j a^k : j = \max(i,k)\}$

   g) $P(x) \Leftrightarrow x \in L((aaa \cup bcb)^*)$

   h) $S(x,y) \Leftrightarrow |x|$ es múltiplo de $|y|_b$

5. Demuestra que las siguientes funciones y predicados sobre el alfabeto $\Sigma = \{a,b,c\}$ son while-computables o while-decidibles, diseñando el macroprograma correspondiente:

   a) $f(x) = \begin{cases} u \bullet v^R & u \in L((ab)^* \cup (ba)^*) \wedge x = u \bullet c \bullet v \\ \varepsilon & \text{c.c.} \end{cases}$

   b) $Q(x,y) \Leftrightarrow \exists z \in \Sigma^+ \,(x = y \bullet y^R \bullet z)$

c) $R(x,y) \Leftrightarrow \exists z \in L(b)^+ \exists u,v \in \Sigma^+ (x = u \bullet z \bullet v \wedge |u| < 2 * |v|)$

d) $g(x,y) = \begin{cases} x & |x| \leq |y| \leq 3 * |x| \\ y & \text{c.c.} \end{cases}$

e) $h(x) = \begin{cases} z & \exists u\,(x = z \bullet u \wedge u = u^R \wedge \text{"no existe ninguna palabra más larga que u que satisfaga la condición"}) \\ aba & \text{c.c.} \end{cases}$

6. Demuestra que las siguientes funciones son while-computables, incluyendo las explicaciones oportunas para documentar los macroprogramas construidos:

   a) Dado el alfabeto $\Sigma = \{a\}$, la función $f : \Sigma^* \rightarrow \Sigma^*$ definida como $f(a^n) = a^{n^2}$

   b) Dado el alfabeto $\Sigma = \{a,b\}$, la función *sin_repes*: $\Sigma^* \rightarrow \Sigma^*$, que reduce todas las cadenas formadas por un solo símbolo repetido, a una única aparición de dicho símbolo. Por ejemplo *sin_repes*(aabaaabbbbb) = abab, y *sin_repes*(ab) = ab.

   c) Dado el alfabeto {a,b}, la función *quita_aba*: $\Sigma^* \rightarrow \Sigma^*$ que elimina de una palabra todos los símbolos que intervienen en apariciones de la subpalabra aba. Así, *quita_aba*(bbabbbbaabbba) = bbbbbabba, *quita_aba*(bababa) = b, y *quita_aba*(bbba) = bbba.

   d) Dado el alfabeto {a,b}, la función que reemplaza la primera aparición de una cadena en una palabra por otra cadena, es decir:

   $$\text{reemplazar}(x,y,z) \cong \begin{cases} \perp & \neg\text{subpalabra?}(y,x) \\ u \bullet z \bullet v & x = u \bullet y \bullet v \wedge |u \bullet y|_y = 1 \end{cases}$$

7. Sea el alfabeto {0,1}. Construye macroprogramas para computar las siguientes funciones:

   a) La función f(x) que obtiene a partir de una palabra x otra cadena que expresa la longitud de x en binario.

   b) La función g(x) que, dado un número en forma binaria, calcula el cociente de dividirlo por 3, también en forma binaria. Considera que la palabra vacía también representa el valor cero, y no te preocupes si en el resultado aparecen ceros a la izquierda.

   c) La función h(x,y) que obtiene la cadena binaria que representa la suma de los números binarios x e y.

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

a) $f(x) = \begin{cases} cons_b(f(cdr(x))) & car_a?(x) \\ cons_a(f(cdr(x))) & car_b?(x) \\ \varepsilon & c.c. \end{cases}$

b) $g(x,y) = \begin{cases} y & x = \varepsilon \\ g(cdr(x), cdr(y)) & c.c. \end{cases}$

C. APRENDER A UTILIZAR LAS LEYES DE LAS PARTICULARIZACIONES, DE LA COMPOSICIÓN LOCAL Y DE LA DEFINICIÓN TOTAL POR CASOS PARA EVITAR TENER QUE ESCRIBIR ALGUNOS PROGRAMAS.

9. Demuestra, sin hacer el programa while o macroprograma correspondiente, que las siguientes funciones son while-computables. Puedes utilizar las leyes de formación de funciones while-computables para hacerlo.

a) cuatriplica(x) = x•x•x•x

b) añade_aba(x) = aba•x

c) $f(x, y) = x^R \bullet y^R$

d) $g(x, y) = x \bullet y^R \bullet x$

e) $rdc(x) = \begin{cases} u & \exists s \in \Sigma\,(x = u \bullet s) \\ \varepsilon & c.c. \end{cases}$

f) $h(x) = \begin{cases} u & |x| \geq 2 \wedge \text{"u es el resultado de quitar el segundo símbolo de x"} \\ \varepsilon & |x| \leq 1 \end{cases}$

10. Demuestra, sin hacer el programa-while o macroprograma correspondiente, que las siguientes funciones son while-computables. Utiliza los resultados del ejercicio 6.

a) $k(x,y) = \begin{cases} quita\_aba(y) \bullet x & subpalabra?(aba, y) \\ x^R & c.c. \end{cases}$

b) $\psi(x,y,z) \cong \begin{cases} \perp & \neg subpalabra?(y, x) \\ w & x = u \bullet y \bullet v \wedge |u \bullet y|_y = 1 \wedge \text{"w es el resultado de quitar las apariciones de aba en } u \bullet z \bullet v\text{"} \end{cases}$

c) $\chi(x,y,z) \cong \begin{cases} \perp & \neg subpalabra?(y, x) \\ u \bullet w \bullet v & x = u \bullet y \bullet v \wedge |u \bullet y|_y = 1 \wedge \text{"w es el resultado de quitar las apariciones de aba en z"} \end{cases}$