Hoja I.3

SOLUCIONES DE ALGÚN APARTADO DEL EJERCICIO 1

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

CONT es una macrovariable
CONT := sig(CONT & X3); es una macroexpresión

ORDEN DE EXPANSIÓN:

- ✓ Macroexpresiones (
  - o 1º Z:=CONT&X3
  - o 2º CONT:=sig(Z))
- ✓ Macrovariables

SOLUCIONES DE ALGÚN APARTADO DEL EJERCICIO 2

```
X1 := cdr(X1);
if cara?(cdr(X1)) then –Macro de control, macroexpresión
   X0 := consb(X1);
else
   X1 := cdr(X1);
   X0 := consa(X1);
end if;
```

Expansión →

```
X3:=ε;
X1:= cdr(X1);
X1:= cdr(X1);
if cara?(X1) then
   X0:=consb(X1);
   X3:= consb(X3);
end if;
if carb?(X3) then
   X1:=cdr(X1);
   X0:= consa(X1);
end if;
```

Hoja I.3

SOLUCIONES DE ALGUNOS APARTADOS DEL EJERCICIO 4

Planteamos una posible solución a modo de ejemplo para demostrar la computabilidad de algunas funciones y la decibilidad de algunos predicados. Por supuesto no es la única solución. Además la estrategia a la hora de programar podría cambiar según las macros que permitamos. No todas las soluciones aquí presentadas utilizan las macros al mismo nivel.

$$\mathbf{4a}\ \psi(\mathbf{x}) \cong \begin{cases} \mathbf{u} & \exists \mathbf{v} \in L((\mathbf{bbb} \cup \mathbf{c})^*)\ (\mathbf{x} = \mathbf{u} \bullet \mathbf{a} \bullet \mathbf{v}) \\ \perp & \text{c.c.} \end{cases}$$

PRIMERA ESTRATEGIA: Invertimos X1 en una variable auxiliar AUX. Vamos eliminando **c** y **bbb** mientras los vayamos encontrando. Comprobamos si el siguiente símbolo es **a** y de ser así devolvemos lo que ha quedado en AUX, en caso contrario el programa ha de ciclar.

PROGRAMA

```
AUX:=X1^R;
- - - ∃v(X1=AUX^R•v ∧v∈L(c∪bbb)*
- - - -buscamos v de la mayor longitud posible
while prefijo?('bbb', AUX) or car_c?(AUX) loop
        if car_c?(AUX) then
                AUX:=cdr(AUX);
        end if;
        if car_b?(AUX) then
                AUX:=cdr(cdr(cdr(AUX)));
        end if;
end loop;
- - - - - - en AUX tenemos X1^R tras eliminar el mayor sufijo v con v∈L(c∪bbb)*
- - - - - - solo si AUX comienza por a es de la forma prevista
while nonem?(AUX) loop
        if car_a?(AUX) then
                X0:=(cdr(AUX))^R;
                AUX:= ε;
        end if;
end loop;
```

Hoja I.3

4a $\psi(\mathbf{x}) \cong \begin{cases} \mathbf{u} & \exists \mathbf{v} \in L((\mathbf{bbb} \cup \mathbf{c})^*)\ (\mathbf{x} = \mathbf{u} \bullet \mathbf{a} \bullet \mathbf{v}) \\ \perp & \text{c.c.} \end{cases}$

SEGUNDA ESTRATEGIA: Buscamos U y V tales que X1 = U•a•V localizando la primera **a** de AUX (copia de la inversa de X1) y después comprobamos si V tiene la estructura adecuada

PROGRAMA

```
AUX:=X1^R;
V:= ε;
- - - X1^R=AUX•V
while not car_a?(AUX) loop
        V:=primero(AUX)&V;
        AUX:=cdr(AUX);
end loop;
- - - X1^R=AUX•a•V o bien el bucle no termina
X0:= (cdr(AUX))^R;
- - - comprobamos si V es de la forma pedida, si no lo es ciclará
while nonem?(V) loop
        if prefijo?('bbb', V) then
                V:=cdr(cdr(cdr(V)));
        end if;
        if car_c?(V) then
                V:=cdr(V);
        end if;
end loop;
```

Hoja I.3

4b R(**x**,**y**,**z**) ⇔ **u•y** = **z**^{R} donde **u** se obtiene eliminando las **a**'s de **x**.

ESTRATEGIA: Construimos **u**, lo concatenamos con **y** y lo comparamos con **z**^{R}. Como no podemos usar la macro-condición = por ser de dos argumentos comprobamos si la igualdad es cierta símbolo a símbolo.

PROGRAMA

```
AUX1:=X1;
AUX2:=X2;
AUX3:=X3^R;
U:= ε;
- - - construcción de u^R
while nonem?(AUX1) loop
        if car_b?(AUX1) then U:= cons_b?(U); end if;
        if car_c?(AUX1) then U:= cons_c?(U); end if;
        AUX1:=cdr(AUX1);
end loop;
- - - construcción de u•y
while nonem?(U) loop
        if car_b?(AUX1) then AUX2:= cons_b?(AUX2); end if;
        if car_c?(AUX1) then AUX2:= cons_c?(AUX2); end if;
        U:=cdr(U);
end loop;
- - - comparación de u•y y z^R
while car_a?(AUX2) and car_a?(AUX3) or
        car_b?(AUX2) and car_b?(AUX3) or
        car_c?(AUX2) and car_c?(AUX3) or loop
        AUX1:=cdr(AUX1);
        AUX3:=cdr(AUX3);
end loop;
X0:= 'a';
if nonem?(AUX1) or nonem?(AUX2)  then
        X0:= ε
end if;
```

Hoja I.3

$$\mathbf{4c}\quad \mathbf{f(x,y)} = \begin{cases} \mathbf{u} & \exists \mathbf{z} \in \Sigma^+ \ (\mathbf{x} = \mathbf{y}^R \bullet \mathbf{z} \wedge \text{"}\mathbf{u} \text{ es el resultado de eliminar las } \mathbf{c}\text{'s de } \mathbf{z} \text{ e intercambiar sus } \mathbf{a}\text{'s por } \mathbf{b}\text{'s"}) \\ \varepsilon & \text{c.c.} \end{cases}$$

ESTRATEGIA: Comprobamos si $\mathbf{y}^R$ es prefijo de **x** sin utilizar la macro-condición prefijo comprobándolo símbolo a símbolo

PROGRAMA

```
AUX1:=X1;
AUX2:=X2^R;
while car_a?(AUX1) and car_a?(AUX2) or
      car_b?(AUX1) and car_b?(AUX2) or
      car_c?(AUX1) and car_c?(AUX2) loop
  - - ∃V X1= V•AUX1 ∧ X2^R= V•AUX2
      AUX1:=cdr(AUX1);
      AUX2:=cdr(AUX2);
end loop;
FALLO:= 'a';
if nonem?(AUX2) then
      X0:=ε;
      FALLO:=ε;
end if;
   - - (∃V∈∑* (X1= V•AUX1 ∧ X2^R= V•AUX2 ∧ AUX2 = ε) →(prefijo?( X2^R,X1) ∧ FALLO = a)) ∨
   - - (∃V∈∑* (X1= V•AUX1 ∧ X2^R= V•AUX2 ∧ AUX2 ≠ ε) →(¬prefijo?( X2^R,X1) ∧ FALLO = ε))
if car_a?(FALLO) then
      U:=Ɛ;
      while nonem?(AUX1) loop
            if car_a?(AUX1) then U:= cons_b(U); end if;
            if car_b?(AUX1) then U:= cons_a(U); end if;
            AUX1:=cdr(AUX1);
      end loop;
      X0:=U^R;
end if;
```

Hoja I.3

4f $Q(x) \Leftrightarrow x \in \{a^i b^j a^k : j=\max(i,k)\}$

ESTRATEGIA: Intentaremos hacerlo en una sola pasada. Determinaremos los valores **i**, **j** y **k** tales que $\mathbf{x}=\mathbf{a}^i\mathbf{b}^j\mathbf{a}^k\mathbf{z}$, y para ello usaremos variables homónimas. Primero contamos las **a**'s iniciales de **x** en la variable I. Cuando se terminan, contamos en la variable J las **b**'s que figuren seguidamente en **x**, al mismo tiempo que comparamos dicho número de **b**'s con el contenido de I. Para terminar, extraemos las **a**'s que vayan a continuación y las comparamos con el contenido de J. Los casos posibles son, a medida que vamos comparando:

1. $\mathbf{z} \neq \mathcal{E} \Rightarrow$ el predicado es falso
2. $\mathbf{i} > \mathbf{j} \Rightarrow$ el predicado es falso
3. $\mathbf{i} \leq \mathbf{j}$
    a. $\mathbf{j} < \mathbf{k} \Rightarrow$ el predicado es falso
    b. $\mathbf{j} \geq \mathbf{k}$
        i. $\mathbf{i} < \mathbf{j} > \mathbf{k} \Rightarrow$ el predicado es falso
        ii. resto de casos $\Rightarrow$ el predicado es cierto

La estructura de casos nos sugiere que empecemos suponiendo que el resultado es verdadero, y que lo cambiemos a falso al ir detectando los casos correspondientes.

PROGRAMA

```
X0:= 'a';  AUX1:= X1;  I:= ε;
AUX2:= ε;
while car_a?(AUX1) loop
        I:= cons_a(I);
        AUX1:= cdr(AUX1);
end loop;
```

-- I contiene a^i, AUX1 contiene el resto de X1 (ba^k y lo que siga)

```
J:= ε;
while car_b?(AUX1) loop
        J:= cons_b(J);
        AUX1:= cdr(AUX1);
        IMENORJ:= 'a';
        if car_a?(I) then
                I:= cdr(I);
                IMENORJ:= ε;
        end if;
end loop;
```

-- J contiene b^j, AUX1 contiene el resto de X1 (a^k y lo que siga),I contiene a^{i-j}

-- IMENORJ indica si i<j (puede ser importante para el caso 3-b-i)

```
if car_a?(I) then
```

-- Estamos en el caso 2

```
        X0:= ε;
end if;
while nonem?(J) loop
        J:= cdr(J);
        KMENORJ:= 'a';
        if car_a?(AUX1) then
                AUX1:= cdr(AUX1);
                KMENORJ:= ε;
        end if;
end loop;
```

-- AUX1 contiene el resto de X1 tras quitarle $a^ibja^{min(j,k)}$

-- KMENORJ indica si k<j (puede ser importante para el caso 3-b-i)

```
while nonem?(AUX1) loop
```

-- Estamos en el caso 1 (si AUX1 empieza por b o c) o en el caso 3-a (si AUX1 empieza por a)

```
        X0:= ε;
end loop;
if car_a?(IMENORJ) then
        if car_a?(KMENORJ) then
                X0:= ε;
        end if;
end if;
```

Hoja I.3

SOLUCIONES DE ALGUNOS APARTADOS DEL EJERCICIO 5

$$\mathbf{5d}\quad g(\mathbf{x},\mathbf{y}) = \begin{cases} \mathbf{x} & |\mathbf{x}| \leq |\mathbf{y}| \leq 3*|\mathbf{x}| \\ \mathbf{y} & \text{c.c.} \end{cases}$$

ESTRATEGIA: Tendremos dos variables MAX y MIN que contendrán palabras de longitud $|3*\mathbf{x}|$ y $|\mathbf{x}|$ respectivamente, y las iremos reduciendo al ritmo del dato **y**. Si MAX se acaba antes de tiempo, o en MIN queda al final algún símbolo, devolveremos **y**. En caso contrario, **x**.

PROGRAMA

```
AUX:= X2;   X0:= X1;
MAX:= X1&X1&X1;    MIN:= X1;
while  nonem?(AUX)  loop
      if  not nonem?(MAX)  then
            X0:= X2;
            AUX:= ε;
      end if;
      if  nonem?(MAX)  then
            MIN:= cdr(MIN);
            MAX:= cdr(MAX);
            AUX:= cdr(AUX);
      end if;
end loop;
if  nonem?(MIN)  then
      X0:= X2;
end if;
```

Hoja I.3

5e $h(\mathbf{x}) = \begin{cases} \mathbf{z} & \exists \mathbf{u}\,(\mathbf{x} = \mathbf{z} \bullet \mathbf{u} \wedge \mathbf{u} = \mathbf{u}^R \wedge \text{"no existe ninguna palabra más larga que } \mathbf{u} \text{ que satisfaga la condición"}) \\ \mathbf{aba} & \text{c.c.} \end{cases}$

ESTRATEGIA: Tenemos en cuenta que siempre encontraremos la palabra **u** ya que al menos ε lo cumple. La palabra más larga podría ser la propia **x**, por lo que empezamos comprobando si es capicua. De no ser así la siguiente más larga podría ser la que obtenemos eliminando el último símbolo y así sucesivamente. El programa se simplifica con el uso de AUX<>AUX^R como macrocondición.

PROGRAMA

```
X0:=ε; AUX:=X1; SEGUIR:='a';
while AUX/=AUX^R loop
        X0:=primero(AUX^R) & X0;
        AUX:=cdr(AUX^R);
end loop;
```

Hoja I.3

SOLUCIONES DE ALGUNOS APARTADOS DEL EJERCICIO 6

6b *sin_repes*: $\sum^* \rightarrow \sum^*$

```
X0:= ε;
AUX:= X1;
while nonem?(AUX) loop
        if car_a?(AUX) then
                X0:= cons_a(X0);
                AUX:=cdr(AUX);
                while car_a?(AUX) loop
                        AUX:=cdr(AUX);
                end loop;
         else
                X0:= cons_b(X0);
                AUX:=cdr(AUX);
                while car_b?(AUX) loop
                        AUX:=cdr(AUX);
                end loop;
        end if;
end loop;
X0:= X0^R;
```

6c *quita_aba*: $\sum^* \rightarrow \sum^*$

```
X0:= ε;
AUX:= X1;
while nonem?(AUX) loop
      if prefijo?(aba, AUX) then
            ---- comprobamos si hay solapamiento
            if prefijo?(aba, cdr(cdr(AUX)) then
                  AUX:=cdr(cdr(AUX));
            else AUX:=cdr(cdr(cdr(AUX)));
            end if;
      else
            X0:= X0&primero(AUX);
            AUX:=cdr(AUX);
      end if;
end loop;
```

Hoja I.3

6d $\text{reemplazar}(\mathbf{x},\mathbf{y},\mathbf{z}) \cong \begin{cases} \bot & \neg\text{subpalabra?}(\mathbf{y},\mathbf{x}) \\ \mathbf{u}\bullet\mathbf{z}\bullet\mathbf{v} & \mathbf{x}=\mathbf{u}\bullet\mathbf{y}\bullet\mathbf{v} \wedge |\mathbf{u}\bullet\mathbf{y}|_{\mathbf{y}}=1 \end{cases}$

ESTRATEGIA: Primero buscamos la primera aparición de **y** en **x**. Si esta no se da, el bucle de búsqueda ciclará, tal y como se ha especificado. Los símbolos desechados (cadena **u**) pasan directamente al resultado, al que añadiremos después la cadena **z**. Para terminar, eliminamos la aparición de **y** y también incorporamos la cadena resultante **v** al resultado.

PROGRAMA

```
X0:= ε;
AUX:= X1;
while not prefijo?(X2, AUX) loop
      X0:= X0 & primero(AUX);
      AUX:= cdr(AUX);
end loop;
-- X0=u ∧ AUX=y•v
X0:= X0&X3;
-- X0=u•z
SUBPALABRA:= X2;
while nonem?(SUBPALABRA) loop
      SUBPALABRA:= cdr(SUBPALABRA);
      AUX:= cdr(AUX);
end loop;
-- AUX=v
X0:= X0 & AUX;
-- X0=u•z•v
```

Hoja I.3

SOLUCIONES DEL EJERCICIO 7

7a Longitud en binario

ESTRATEGIA: Vamos quitando símbolos de la entrada y sumando **1** al contenido de X0

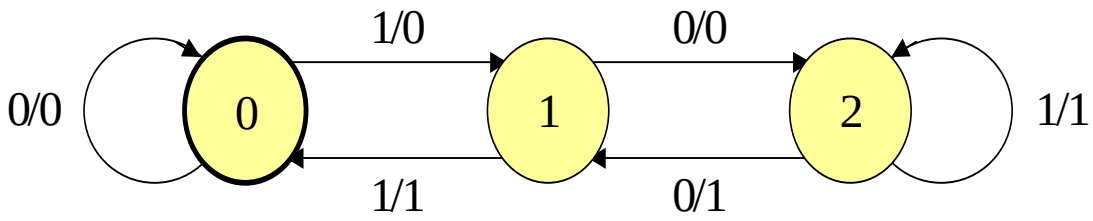
PROGRAMA

```
X0:='0'; AUX:= X1;
while nonem?(AUX) loop
        AUX:=cdr(AUX);
        LONG:=X0; X0:=ε;
        while car1?(LONGR) loop
                X0:= cons0(X0);
                LONG:=rdc(LONG);
        end loop;
        X0:=rdc(LONG)&cons1(X0);
end loop;
```

Hoja I.3

7b Dividir por tres en binario

ESTRATEGIA: No es difícil dividir por tres en binario. A medida que vamos avanzando en las cifras del dividendo, iremos produciendo las del cociente, pero también es importante llevar la cuenta del resto. Inicialmente el resto será cero, y en cada iteración el resto de la anterior se duplica y se suma a la nueva cifra del dividendo.
Podemos representar la solución mediante un traductor finito en el que cada estado representa un resto:



PROGRAMA

```
X0:= ε;
AUX:= X1;   ESTADO:= '0';
while  nonem?(AUX)  loop
      if  ESTADO =  '0'  then
            if  car0?(AUX)  then  X0:= snoc0(X0);  end if;
            if  car1?(AUX)  then  X0:= snoc0(X0);   ESTADO:= '1';  end if;
      elsif  ESTADO = '1'  then
            if  car0?(AUX)  then  X0:= snoc0(X0);   ESTADO:= '10';  end if;
            if  car1?(AUX)  then  X0:= snoc1(X0);   ESTADO:= '0';  end if;
      else
            if  car0?(AUX)  then  X0:= snoc1(X0);   ESTADO:= '1';   end if;
            if  car1?(AUX)  then  X0:= snoc1(X0);  end if;
      end if;
      AUX:=cdr(AUX);
end loop;
```

Hoja I.3

7c Sumar en binario

ESTRATEGIA: Hacer la suma de la manera habitual desde la derecha, por lo que comenzamos invirtiendo las palabras a sumar. Una variable auxiliar nos informa si llevamos o no.

```
AUX1:= X1^R; AUX2:= X2^R; LLEV0:='0';
while nonem?(AUX1) and nonem?(AUX2) loop
        if car1?(LLEVO) then
                if car0?(AUX1) and car0?(AUX2) then
                        X0:= cons1(X0); LLEVO:= '0';
                elsif   (car0?(AUX1) and car1?(AUX2)) or
                        (car1?(AUX1) and car0?(AUX2))  then
                        X0:= cons0(X0);
                else    X0:= cons1(X0);
                end if;
        elsif car0?(AUX1) and car0?(AUX2) then
                        X0:= cons0(X0);
        elsif   (car0?(AUX1) and car1?(AUX2)) or
                (car1?(AUX1) and car0?(AUX2))  then
                        X0:= cons1(X0);
        else    X0:= cons0(X0); LLEV0:='1';
        end if;
        AUX1:= cdr(AUX1); AUX2:= cdr(AUX2);
end loop;
if nonem?(AUX1) then Z:= AUX1;  else Z:= AUX2;  end if;
if car1?(LLEVO) then
        while car1?(Z) then
                X0:= cons0(X0);
                Z:=cdr(Z);
        end loop;
        X0:= cons1(X0);
        Z:=cdr(Z);
end if;
X0:= Z^R&X0;
```

Hoja I.3

SOLUCIONES DEL EJERCICIO 8

8a $$f(\mathbf{x}) = \begin{cases} cons_{\mathbf{b}}(f(cdr(\mathbf{x}))) & car_{\mathbf{a}}?(\mathbf{x}) \\ cons_{\mathbf{a}}(f(cdr(\mathbf{x}))) & car_{\mathbf{b}}?(\mathbf{x}) \\ \varepsilon & \text{c.c.} \end{cases}$$

ESTRATEGIA: Según la definición lo que hace la función es intercambiar **a**'s y **b**'s

```
X0:= ε; AUX:= X1^R;
while nonem?(AUX) loop
        if car_a?(AUX) then
                X0:= cons_b(X0);
        else    X0:= cons_a(X0);
        end if;
        AUX:= cdr(AUX);
end loop;
```

8b $$g(\mathbf{x},\mathbf{y}) = \begin{cases} \mathbf{y} & \mathbf{x} = \varepsilon \\ g(cdr(\mathbf{x}), cdr(\mathbf{y})) & \text{c.c.} \end{cases}$$

ESTRATEGIA: Según la definición lo que hace la función es eliminar de **y** tantos símbolos como tiene **x** y devolver el resto

```
X0:= X2; AUX:= X1;
while nonem?(AUX) loop
        AUX:= cdr(AUX);
        X0:= cdr(X0);
end loop;
```

Hoja I.3

SOLUCIONES DE ALGUNOS APARTADOS DEL EJERCICIO 9

Podemos utilizar funciones computables y predicados decidibles para definir las funciones "por partes"

$$\mathbf{9e}\quad \mathrm{rdc}(\mathbf{x}) = \begin{cases} \mathbf{u} & \exists \mathbf{s} \in \sum (\mathbf{x} = \mathbf{u} \bullet \mathbf{s}) \\ \varepsilon & \text{c.c.} \end{cases}$$

$$\mathrm{rdc}(\mathbf{x}) = \begin{cases} (\mathrm{cdr}(x^R))^R & \mathrm{nonem?}(x) \\ \mathcal{E} & \text{c.c.} \end{cases}$$

$$\mathbf{9f}\quad \mathrm{h}(\mathbf{x}) = \begin{cases} \mathbf{u} & |\mathbf{x}| \geq 2 \ \wedge \text{"}\mathbf{u} \text{ es el resultado de quitar el segundo símbolo de } \mathbf{x}\text{"} \\ \varepsilon & \text{c.c.} \end{cases}$$

$$\mathrm{h}(\mathbf{x}) = \begin{cases} \mathrm{primero}(x) \bullet \mathrm{cdr}(\mathrm{cdr}(x)) & \mathrm{nonem?}(\mathrm{cdr}(x)) \text{ and } \mathrm{nonem?}(x) \\ \varepsilon & |x| \leq 1 \end{cases}$$