WUOLAH

MAC2-3-4.pdf

patriciacorhid

Modelos Avanzados de Computacion

4º Doble Grado en Ingeniería Informática y Matemáticas

Facultad de Ciencias
Universidad de Granada

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Estudiar sin publi es posible.

Compra Wuolah Coins y que nada te distraiga durante el estudio

GET IT ON Google Play

Download on the App Store

Relación 2

7. Dado un conjunto finito de lenguajes que recubran el alfabeto con intersección vacía y todos r.e., demostrar que son recursivos.

Tomo Li fijo, que sabemos que es r.e. por hipotesis. Su complementario es la unión de los Lj con j != i. Si probamos que la unión finita de r.e es r.e, Li complementario sería r.e, luego Li sería recursivo.

Creamos una MT que, dada una entrada w, en cada cinta ejecute el comportamiento de la MT que acepta Lj con la entrada w. Si uno de los lenguajes Lj con j ≠ i acepta w, la MT acepta w. En caso contrario, rechaza.

Hemos construido una MT que acepta todos los casos positivos, luego la unión finita es r.e, por tanto Li es recursivo. Como el i era arbitrario, todos los Li son recursivos.

9. Estudiar si las clases de lenguajes recursivos y r.e son cerradas para:

Recursivos:

a) **Unión:** La unión finita de lenguajes recursivos es recursiva, basta con construir una MT que dada una entrada w simule el comportamiento de todas las MT que aceptan los lenguajes de la familia y aceptar la entrada si una la acepta y rechazar cuando todas la rechazan.
La unión numerable no es recursiva, pues en los casos negativos tendría que simular el comportamiento en infinitas MT, ya que no puede rechazarla hasta que todas las MT la rechacen.

b) **Intersección:** La intersección finita de lenguajes recursivos es recursiva, basta con construir una MT que dada una entrada w simule el comportamiento de todas las MT que aceptan los lenguajes de la familia y aceptar w sólo cuando todas esas MT acepten w, y rechazar en caso contrario.
La intersección numerable no es recursiva, ya que en los casos positivos tendría que comprobar que se acepta la palabra para infinitas MT.

c) **Concatenación:** La concatenación de n lenguajes recursivos es recursiva. Construimos una MT que ante una entrada w añada n-1 separadores de forma no determinista entre sus caracteres (algunos trozos pueden quedar vacíos) y ejecute la MT del lenguaje j-ésimo sobre el trozo j-ésimo. Acepta sólo si aceptan las n, rechaza en caso contrario. Como todas terminan, ésta termina.

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Compra Coins y descarga sin publicidad.

d) Clausura: (Unión de concatenaciones finitas de un lenguaje consigo mismo)
Dado L, construimos una MT que acepte L^+, ante una entrada w:
- Decide en cada paso de forma no determinista si insertar un separador antes de un símbolo o se mueve a la derecha (empieza poniendo separadores al principio de la palabra y luego entre sus símbolos).
- Cuando llegue al blanco después de la palabra, decide de forma no determinista si insertar un separador antes del blanco (al final de la palabra) o ejecutar.
- Cuando decida ejecutar, comprueba si cada uno de los trozos (puede haber vacíos) es aceptado por el lenguaje L.

e) Homomorfismo: Dada una entrada w, se selecciona de forma no determinista una palabra u, se comprueba que $u \in L$ y que $f(u)=w$, sólo en este caso se acepta.
$(w \in f(L) \Leftrightarrow \exists u \in L$ tal que $f(u)=w)$

f) Homomorfismo Inverso: Se comprueba que f(w) pertenezca a L.
$(w \in f^{-1}(L) \Leftrightarrow f(w) \in L)$

Recursivamente Enumerables:

a) Unión: La unión numerable de lenguajes r.e es r.e. Construimos una MT que acepte la entrada w si una de las MT que acepta un lenguaje de la familia acepta la palabra w.
Primero ejecuta la primera máquina 1 paso, luego las dos primeras 2 pasos, etc. (todo de forma secuencial). Hasta que una acepte.

b) Intersección: La intersección finita de lenguajes r.e es r.e. Basta ejecutar todas las máquinas en paralelo y aceptar cuando todas ellas acepten.
La intersección numerable no es r.e., habría que comprobar infinitas MT.

El resto se hacen igual.

	Recursivo	**R. Enumerable**
a) Unión	Finita	Numerable
b) Intersección	Finita	Finita
c) Concatenación	Sí	Sí
d) Clausura	Sí	Sí
e) Homomorfismo	Sí	Sí
f) Homomorfismo Inverso	Sí	Sí

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.
WUOLAH
Compra Coins y descarga sin publicidad.

11. Dada una MT que tenga dos estados: campana y silbato, determinar que saber si una MT entra alguna vez en estado silbato es indecidible.

Reducimos el problema C-EMPTY al problema de ver si una MT entrará en estado silbato. Como C-EMPTY es semidecidible, pero no decidible, nuestro problema es no decidible.

Dada como entrada una máquina de Turing M, construimos una máquina de Turing F(M) en la que etiquetamos todos los estados finales como estados silbato y los no finales como estados campana.

Si M acepta un lenguaje no vacío ⇒ F(M) entra en estado silbato
Si M acepta el lenguaje vacío ⇒ F(M) no entra en estado silbato

Por tanto, hemos reducido C-EMPTY a nuestro problema, luego nuestro problema es no decidible.

16. Determinar si los siguientes problemas son decidibles, semidecidibles o no semidecidibles.

b) Determinar si una MT no se para para ninguna palabra:
Reducimos al problema complementario a la parada. Como el problema de la parada es semidecidible pero no decidible, este problema es no semidecidible.

Dada la entrada (M, w) del problema complementario a la parada, construimos f(M) una MT que ignore su entrada y simule el comportamiento de M ejecutando w.

Si M no para con w -> f(M) no para para ninguna palabra.
Si M para con w -> f(M) para todas sus entradas.

Hemos reducido el problema complementario a la parada al nuestro, por tanto, el nuestro es no semidecidible.

c) Determinar si una MT se para, al menos, para alguna entrada.
Su complementario es el problema del apartado b), que es no semidecidible, luego éste problema no es decidible. Veamos que es semidecidible.
Construimos una MT que dada como entrada una máquina de Turing M escoja de forma no determinista una palabra w y simule el comportamiento de M ejecutando w.

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Compra Coins y descarga sin publicidad.

Si M para con w -> acepta
En caso contrario -> rechaza

d) Determinar si una MT no se para, al menos, para una entrada.
Reducimos el problema complementario a la parada a éste. Como el problema de la parada es semidecidible pero no decidible, el complementario es no semidecidible.

Dada una entrada (M,w) del problema complementario a la parada, construimos una máquina f(M) que ignore su entrada y simule el comportamiento de M ejecutando w.

Si M no para con w -> f(M) no para para ninguna entrada -> f(M) cicla para alguna.
Si M para con w -> f(M) se para con todas las entradas.

Hemos reducido el problema complementario a la parada al nuestro, luego el nuestro es no semidecidible.

18. Sea el problema de determinar si una MT acepta a lo más 100 palabras. Determinar si es decidible, semidecidible o no semidecidible.
Reducimos el problema EMPTY a nuestro problema. Como el EMPTY es no semidecidible, el nuestro es no semidecidible.

Dada una entrada M, una máquina de Turing, del problema EMPTY, construimos una máquina de Turing, f(M), que acepte 100 palabras más que M. Esto lo hacemos añadiendo un nuevo símbolo $ al alfabeto de f(M) y nuevos estados y transiciones que acepten las palabras $\$^n$ para $1 \leq n \leq 100$.

Si M acepta el lenguaje vacio -> f(M) acepta 100 palabras.
Si M no acepta el lenguaje vacio -> f(M) acepta más de 100 palabras.

Hemos reducido el problema EMPTY al nuestro, luego el nuestro es no semidecidible.

19. Determinar si el siguientes problemas son decidibles, semidecidibles o no semidecidibles.

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Compra Coins y descarga sin publicidad.

Estudiar sin publi es posible.
Compra Wuolah Coins y que nada te distraiga durante el estudio.

e) Dada una MT determinar si no acepta la palabra vacía.
Veamos que el complementario es semidecidible pero no decidible, por tanto, el nuestro tiene que ser no semidecidible.
El complementario es: “Dada una MT determinar si acepta la palabra vacía”.
Por el Teorema de Rice, no es decidible. Veamos que es semidecidible:

Construimos una máquina de Turing que, dada como entrada una máquina de Turing M, simule su comportamiento ejecutando la palabra vacía.
Si M acepta la palabra vacía -> acepto.
En caso contrario -> rechazo.

24. Determinar si el siguientes problemas son decidibles, semidecidibles o no semidecidibles.

c) Dadas dos Máquinas de Turing, determinar si el conjunto de las palabras aceptadas a la vez por ambas máquinas de Turing es finito.

Por el ejercicio 13 b) ya sabemos que determinar si una máquina de Turing acepta un lenguaje finito es un problema no semidecidible. Reduciremos este problema al nuestro.

Dada una entrada M, una máquina de Turing, del problema de determinar si el lenguaje aceptado por esta máquina es finito, construimos la pareja (M, Mu), donde Mu es una máquina de Turing que acepta todas las palabras del alfabeto, luego las palabras aceptadas por ambas máquinas de Turing son las palabras de L(M).

Si M acepta un lenguaje finito -> $L(M) \cap L(Mu)$ es un lenguaje finito.
Si M acepta un lenguaje infinito -> $L(M) \cap L(Mu)$ es un lenguaje infinito.

Hemos reducido el problema de determinar si el lenguaje aceptado por una MT es finito al nuestro, luego el nuestro es no semidecidible.

27. Demuestra que el Problema de las Correspondencias de Post con un alfabeto A que tiene un sólo elemento es decidible.
Usamos la siguiente notación: A cada bloque le asignamos un número entero que representa la resta del número de símbolos de arriba menos el número de símbolos de abajo.
Por ejemplo:

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

GET IT ON Google Play
Download on the App Store
WUOLAH

aa
aaa

2-3 = -1

Buscamos una combinación que de 0. Esto se consigue si uno de los bloques tiene asignado el 0 o hay un bloque positivo y otro negativo. Por ejemplo, si tenemos -13 y 37 basta con poner 37 bloques identificados por el -13 seguidos de 13 bloques identificados por el 37.

Construimos la siguiente MT:
La máquina de Turing acepta como entrada la lista de números enteros que representa cada bloque.

Si uno de los números es 0 -> Se acepta
Si hay uno negativo y otro positivo -> Se acepta
En caso contrario (todos negativos o todos positivos) -> Se rechaza

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Relación 3

Compra Coins y descarga sin publicidad.

Ej 1: Programa Post-Turing que calcule f(u) = u^{-1} con u ∈ {0,1}*

Dada una palabra u, aprovecha el primer símbolo de ésta y escribe a su izquierda poco a poco el resto de símbolos. Ej:
0101 ⟶ 0X01 ⟶ 10X01 ⟶ 10XX1⟶ 010XX1 ⟶ 010XXX ⟶ 1010XXX ⟶ 1010

```
[A]    RIGHT            Mira el símbolo a la derecha
       IF X  GOTO A     Recorre las X hasta que se acaben
       IF # GOTO H      He terminado de leer la palabra
       IF 0 GOTO B      Leí un 0
       IF 1 GOTO D      Leí un 1

[B]    PRINT X          Tacho el símbolo leído y salto a C
       IF X GOTO C

[C]    LEFT             Si leí un 0 voy al principio de la palabra a escribir un 0
       IF X GOTO C
       IF 0 GOTO C
       IF 1 GOTO C
       PRINT 0
       IF 0 GOTO F

[D]    PRINT X          Tacho el símbolo leído y salto a E
       IF X GOTO E

[E]    LEFT             Si leí un 1 voy al principio de la palabra a escribir un 1
       IF X GOTO E
       IF 0 GOTO E
       IF 1 GOTO E
       PRINT 1
       IF 1 GOTO F

[F]    RIGHT            Me muevo a la derecha hasta encontrar la primera X
       IF 0 GOTO F
       IF 1 GOTO F
       IF X GOTO A          Cuando la encuentro, repito el proceso
[H]    LEFT             Al terminar de leer la palabra borra las X
       IF X GOTO G
```

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Compra Coins y descarga sin publicidad.

```
        HALT

[G]     PRINT #            Borro las X
        IF # GOTO H
```

Ej 2: Programa Post-Turing que calcule f(u) = u + 1 con u un número binario

```
IF # GOTO S

[A]     RIGHT              Voy al final de la palabra
        IF 0 GOTO A
        IF 1 GOTO A
        LEFT               Leo el último símbolo de ésta
        IF 0 GOTO B
        IF 1 GOTO C

[B]     PRINT 1            Cambio el 0 (o el #) por un 1 y termino
        HALT

[C]     PRINT 0            Cambio el 1 por un 0 y busco el próximo 0 para
        LEFT               cambiarlo por un 1 o en su defecto, poner un 1 al inicio de
        IF 0 GOTO B        la palabra
        IF # GOTO B
        IF 1 GOTO C

[S]     La palabra vacía no es una entrada válida, el programa no puede terminar con ella
```

Ej 3: Programa Post-Turing que calcule f(ucv) = si u subcadena de v

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Compra Coins y descarga sin publicidad.

Estudiar sin publi es posible.
Compra Wuolah Coins y que nada te distraiga durante el estudio.

$i, j \in \{0, 1\}$ i' denota 0 si i=1 y 1 si i=0

Xi se usa para tachar el símbolo i en u
Yi se usa para tachar el símbolo i en v
Z se usa para tachar un símbolo de v que hemos probado si la copia de u empieza en él. Si en algún momento se acaba v, no tiene sentido probar a empezar más adelante.

101c001001	
$X_1$01cZZZ001	Marca primer 1 de u y lo empareja con el primer 1 que aparece en v
$X_1X_0$1cZZZ$Y_0$01	Marca siguiente símbolo de u y lo empareja con el siguiente de v
$X_1X_0X_1$cZZZ$Y_0$01	Marca siguiente símbolo de u, como no empareja con el siguiente de v, hay que restaurar
101cZZZ001	Seguimos buscando u en v
$X_1$01cZZZZZZ	
$X_1X_0$1cZZZZZZ	Marca siguiente símbolo de u, pero se le acaba v buscando compañero

```
        IF c GOTO H         La palabra vacía está contenida en cualquiera
        IF i GOTO Ai

[Ai]    PRINT Xi
[Bi]    RIGHT
        IF j GOTO Bi        Recorre u hasta que llegue a c
[Pi]    RIGHT               Primer símbolo de v
        IF Z GOTO Pi        Saltamos todas las Zs que llevábamos puestas
        IF i' GOTO Ci       Encuentro un símbolo que no es el que busco
        IF i GOTO D         Encuentro el símbolo que busco
        IF # GOTO S         No quedan símbolos de v por probar

[Ci]    PRINT Z             Si no es el símbolo que esperaba imprimo Z
        RIGHT               y sigo buscando el símbolo bueno
        IF i' GOTO Ci
        IF # GOTO S         u no es subcadena de v (Terminamos de leer v y no estaba u)
        IF i GOTO D
[D]     PRINT Z             Probamos a empezar la copia de u en este símbolo
[E]     LEFT
        IF 0, 1, c, Yi, Z GOTO E    A Xi en u
        RIGHT                       Siguiente símbolo de u
        IF c GOTO H                 Se acabó u
        IF i GOTO Fi                Tenemos que comprobar que el siguiente
```

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.
GET IT ON Google Play
Download on the App Store
WUOLAH

símbolo de u es el siguiente de v

[Fi]	PRINT Xi	Marcamos el siguiente símbolo de u
[Gi]	RIGHT	
	IF j GOTO Gi	Recorre u hasta que llegue a c
[Ji]	RIGHT	Llegamos a la c y vamos avanzando hasta encontrar
	IF Z, Yj GOTO Ji	el primer símbolo de v sin comprobar
	IF i’ GOTO L	Intento fallido, tenemos que restaurar u y v (salvo Zs)
	IF # GOTO S	No vamos a encontrar nunca el símbolo que buscamos
	IF i GOTO Ki	Encontramos el símbolo que buscamos en v
[Ki]	PRINT Yi	Marcamos el siguiente símbolo de v
	IF Yi GOTO E	Volvemos al siguiente símbolo de u
[L]	LEFT	Recorremos la palabra cambiando los Xi y Yi por i
	IF Yi GOTO Li	
	IF Xi GOTO Li	
	IF c, Z, i GOTO L	
	RIGHT	Primer símbolo de u
	IF i GOTO Ai	Empezamos de nuevo (ahora v es más corta por los Zs)
[Li]	PRINT i	
	IF i GOTO L	
[H]	HALT	Si
[S]	NO	

Ej 4: Programa con variables para concatenar las cadenas que están en X1 y X2 y devolverlas en Y.

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Compra Coins y descarga sin publicidad.

[A]	IF X2 ENDS i GOTO Ai	Hasta que X2 esté copiado en Y
	IF X1 ENDS i GOTO Bi	Luego copio X2 en Y
	HALT	

[Ai]	Y ← iY
	X2 ← X2-
	GOTO A

[Bi]	Y ← iY
	X1 ← X1-
	GOTO A

Ej 5: Programa con variables que calcule el número de apariciones de X1 en X2, salida en Y

En X2 quitamos los números que hemos comprobado que pueden ser el final de X1

X1	100
U1	
X2	10001
U2	
Y	0

X1	10	Quita el 0 de X1
U1	0	Lo guarda en U1
X2	100	Quita el 0 de X2 (se lleva por delante el 1)
U2		
Y	0	

X1	1	Quita el 0 de X1
U1	00	Lo guarda en U1
X2	10	Quita el 0 de X2 (se lleva por delante el 1)
U2	0	Lo guarda en U2
Y	0	

X1		Quita el 1 de X1
U1	100	Lo guarda en U1
X2	10	No encuentra un 1 en X2, falla y hay que restaurar

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Compra Coins y descarga sin publicidad.

U2	0	
Y	0	

X1	100	X1=X1.U1
U1		Vacio U1
X2	100	X2=X2.U2
U2		Vacio U2
Y	0	

[A]	IF X1 ENDS i GOTO Ai	
[Ai]	U1 ← iU1	
	X1 ← X1-	Quitamos i de X1
[Bi]	IF X2 ENDS i' GOTO Ci	
	IF X2 ENDS i GOTO Di	
	HALT	Se acaba X2, ya no habrá más ocurrencias
[Ci]	X2 ← X2-	Quitamos del final de X2 hasta encontrar i
	GOTO Bi	
[Di]	X2 ← X2-	Quitamos i, ahora hay que ver que el resto de
	GOTO E	símbolos de X1 coinciden
[E]	IF X1 ENDS i GOTO Ei	
	GOTO S	No quedan más símbolos de X1, luego hemos encontrado una ocurrencia suya en X2
[Ei]	U1 ← iU1	
	X1 ← X1-	Quitamos i de X1
[Fi]	IF X2 ENDS i GOTO Gi	Es el símbolo correcto
	IF X2 ENDS i' GOTO L	No es el símbolo que esperamos y hay que restaurar
	HALT	Se acaba X2, ya no habrá más ocurrencias
[Gi]	U2 ← iU2	Lo guardamos en U2
	X2 ← X2-	Lo quitamos de X2
	GOTO E	
[L]	X1 ← X1U1	Restauramos X1 y X2
	X2 ← X2U2	En el ejercicio hacemos un programa concatena
	GOTO A	

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Compra Coins y descarga sin publicidad.

Estudiar sin publi es posible.
Compra Wuolah Coins y que nada te distraiga durante el estudio.

```
[S]     Y ← Y+1                 Hay que incrementar Y y restaurar X1 y X2
        GOTO L

MACRO: Y ← X+1          (Donde X representa un número en binario)

U ← X
Y ← ε

[L]     IF U ENDS 0 GOTO A
        IF U ENDS 1 GOTO B
[A]     U ← U-          Le quitamos el último símbolo a U y concatenamos U con 1
        Y ← U.1.Y       La concatenación es asociativa
        HALT            NOTA: Cuando U se acabe (era X=1...1), le ponemos
                        un 1 delante a la Y (que será 0...0)

[B]     U ← U-          Si toca un 1, se lleva una y continúa
        Y ← Y.0
        GOTO L
```

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

GET IT ON Google Play
Download on the App Store
WUOLAH

Ej 6: Programa con variables que acepte palíndromos, entrada X

```
        U ← ε
        V ← ε

[A]     IF X ENDS i GOTO Ai
[B]     IF U ENDS i GOTO Bi      Comprobamos si U = V
        HALT                     Ambos son iguales

[Ai]    U ← U.i     Metemos en U el contenido de X al revés
        V ← i.V     Metemos en V el contenido de X
        X ← X-
        GOTO A

[Bi]    IF V ENDS i GOTO C       Ambos símbolos coinciden
        GOTO N                   No coinciden, rechazamos

[C]     U ← U-
        V ← V-
        GOTO B

[N]                 Rechazamos
```

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Ej 7: Programa con variables que dada una cadena u calcule una cadena con los símbolos de las posiciones impares de u. Entrada X, salida Y.

WUOLAH

Compra Coins y descarga sin publicidad.

IF X ENDS i GOTO Ai
HALT

[Ai] V ← iV Guardo posiciones impares empezando por el final
X ← X-
IF X ENDS j GOTO Bj
Y ← V La longitud de X es impar, los impares son los impares por el final
HALT

[Bj] U ← jU Guardo posiciones pares empezando por el final
X ← X-
IF X ENDS i GOTO Ai
Y ← U La longitud de X es par, los impares son los pares por el final
HALT

X=cdefg
U=
V=

X=cde Una pasada por Ag y otra por Bf
U=f
V=g

X=c Una pasada por Ae y otra por Bd
U=df
V=eg

X= Una pasada por Ac y sale escribiendo V en Y
U=df
V=ceg

11. Dado el siguiente programa Post-Turing:

LEFT
[C] RIGHT
IF # GOTO E
IF 0 GOTO A

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Compra Coins y descarga sin publicidad.

```
        IF 1 GOTO C
[A]     PRINT #
        IF # GOTO C
[E]     HALT
```

construir una MT equivalente.

El programa cambia los 0 de la palabra por #.

δ(q0, 0) = (q0, #, D)
δ(q0, 1) = (q0, 1, D)
δ(q0, #) = (qf, #, D)

Ej 12: 12. Dada la MT M = ({q 0 , q 1 , q 2 , q 3 , q 4 }, {0, 1}, {0, 1, X, Y, #}, δ, q 0 , #, {q 4 }) donde las transiciones no nulas son las siguientes:

δ(q0 , 0) = (q1 , X, D)	**δ(q0 , Y) = (q3 , Y, D)**
δ(q1 , 0) = (q1 , 0, D)	**δ(q1 , 1) = (q2 , Y, I)**
δ(q1 , Y) = (q1 , Y, D)	**δ(q2 , 0) = (q2 , 0, I)**
δ(q2 , X) = (q0 , X, D)	**δ(q2 , Y) = (q2 , Y, I)**
δ(q3 , Y) = (q3 , Y, D)	**δ(q3 , #) = (q4 , #, D)**

construir un programa con variables equivalente (se pueden usar macros).

Esta máquina acepta las palabras de la forma 0^n1^n, con n>0.

- q0 busca el primer 0 y lo marca con una X, cuando encuentra una Y interpreta que ya no quedan más 0s y pasa a q3.
- q1 busca el siguiente 1 y lo marca con Y, luego pasa a q2.
- q2 salta las Ys y los 0s, hasta encontrar la última X puesta y pasa a q0.
- q3 comprueba que sólo queden Y.

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Compra Coins y descarga sin publicidad.

Estudiar sin publi es posible.
Compra Wuolah Coins y que nada te distraiga durante el estudio.

El programa con variables que acepta el lenguaje es:

```
U ← ε
V ← X

[L]     IF X ENDS 1 GOTO A
        GOTO S                  No termina en 1, entrada inválida

[A]     V ← V-
        U ← 0                   Guardo en U tantos 0s como 1s tenga X al final
        IF V ENDS 1 GOTO A      Sigo comprobando si hay 1s al final
        IF V ENDS 0 GOTO B      Voy a comprobar los 0s
        GOTO S                  Sólo había 1s, entrada inválida

[B]     V ← V-
        U ← U-                  Quito de U tantos 0s como 0s tenga X al final
        IF U ≠ ε GOTO C         Si debería haber más 0s sigo comprobando lo que hay en
V
        GOTO D                  Si no, compruebo que V sea vacío

[C]     IF V ENDS 0 GOTO B

[D]     IF V ≠ ε GOTO S         Sobran dígitos
        IF U ≠ ε GOTO S         Faltan ceros
        HALT

[S]
```

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

GET IT ON Google Play
Download on the App Store
WUOLAH

Relación 4

1.Programa con variables numéricas que calcule $f(x_1, x_2) = x_1+x_2$ y otro que calcule $f(x_1, x_2) = x_1 \cdot x_2$

- $f(x_1, x_2) = x_1+x_2$:

```
U ← X1
V ← X2
IF X2 ≠ 0 GOTO A          Si X2 es 0, devuelvo X1
GOTO H

[A]   U ← U+1             Sumo X1 a X2
      V ← V-1
      IF V ≠ 0 GOTO A
      GOTO H

[H]   Y ← U               Devuelvo el resultado
      HALT
```

- $f(x_1, x_2) = x_1 \cdot x_2$:

```
U ← 0
V ← X2
IF V ≠ 0 GOTO A    Si X2 no es 0, sumo X1 consigo mismo tantas veces como el valor de X2
GOTO H             Si X2 es 0, devuelvo 0

[A]   U ← U+X1     Sumo X1 consigo mismo tantas veces como el valor de X2
      V ← V-1
      IF V ≠ 0 GOTO A

[H]   Y ← U
      HALT
```

3. Programa con variables numéricas donde f(x) = 1 si x es primo y 0 en caso contrario.

MACRO salto si menor que: IF X1 < X2 GOTO E

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Compra Coins y descarga sin publicidad.

U ← X1
V ← X2

[A] U ← U-1 Resto X2 - X1
V ← V-1
IF U != 0 GOTO A
IF V ≠ 0 GOTO E V vale 0 si X2 era igual o menor que X1

MACRO resto de la división: Y ← X1%X2 (Supongo X2>0)

U ← X2
V ← X1

IF V < X2 GOTO H Si X1 < X2, el resto es X1

[A] U ← U-1 Resto V - X2
V ← V-1
IF U != 0 GOTO A
IF V < X2 GOTO H Si V es menor que X2, el módulo es V
U ← X2 Si V es mayor o igual que X2, vuelvo a restarle X2
GOTO A

[H] Y ← V

SOLUCIÓN: Variable de entrada es X, Y la de salida.

V ← X

W ← 0
W ←W + 1 Guardo un 1 en W

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Compra Coins y descarga sin publicidad.

IF W < V GOTO A		Si V=X mayor que 1, empezamos
Y ← 0		X es 1 ó 0
HALT		
[A]	V ← V - 1	
	IF W < V GOTO B	Si V mayor que 1, divido X entre V
	Y ← 1	Ningún V=X-1,...,2 divide a X
	HALT	
[B]	Z ← X%V	Divido X entre V, si el resto es 0, no es primo.
	IF Z != 0 GOTO A	Si el resto no es 0, sigo dividiendo por otros números.
	Y ← 0	
	HALT	

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

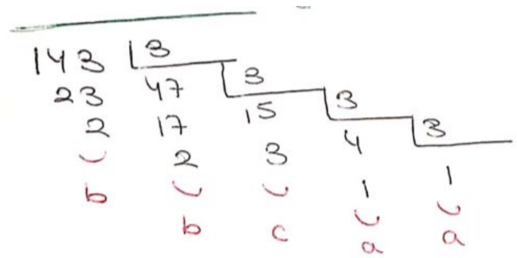
WUOLAH
Compra Coins y descarga sin publicidad.

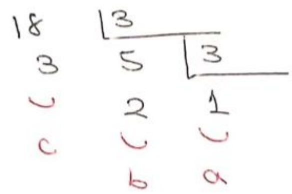
Estudiar sin publi es posible.
Compra Wuolah Coins y que nada te distraiga durante el estudio.

4. Programa con variables numéricas que calcule f(x) = y con y=Z(C(x)-), donde C y Z son las codificaciones sobre un alfabeto de n símbolos.
Supongo que con C(x)- se refiere a quitarle el último símbolo al número.

Si tomamos x=143, C(x) = aacbb. Podemos ver en el método de cálculo de C(x) que Z(C(x)-) = Z(aacb) = 47, que se obtiene dividiendo 143 entre 3 (número de símbolos del alfabeto) y tomando el cociente.

Si tomamos x = 18, como es múltiplo de 3, el resto es 0, luego hay que restarle 1 al cociente y poner de resto 3, luego ahora Z(C(x)-) = 5, no 6.





Así, vemos que si x%n ≠ 0, Z(C(x)-) = x/n. Si x%n = 0, Z(C(x)-) = x/n - 1. Donde n es el cardinal del alfabeto.

La demostración formal de esto es:
Dada una palabra $a_k \ldots a_1$, queremos comparar $Z(a_k \ldots a_1)$ con $Z(a_k \ldots a_2)$

Abc = 3*1 + 2*3 + 1*9
Ab = 2*1 + 1*3

$Z(a_k \ldots a_1) = \sum_{i=1}^{k} Z(a_i) n^{i-1}$ Es divisible por n si y sólo si $Z(a_1) = n$ (*)

$Z(a_k \ldots a_2) = \sum_{i=1}^{k-1} Z(a_{i+1}) n^{i-1} = n * \sum_{i=2}^{k} Z(a_i) n^{i-2} + Z(a_1)$

Por tanto $Z(a_k \ldots a_1) = n \cdot Z(a_k \ldots a_2) + Z(a_1)$. Entonces $Z(a_k \ldots a_2) = (Z(a_k \ldots a_1) - Z(a_1))/n$.
Para calcular el cociente de esa división, sólo tenemos que dividir x entre n y restar 1 en el caso de que $Z(a_1)=n$, es decir, si x%n=0. (*)

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

GET IT ON Google Play
Download on the App Store
WUOLAH

****

MACRO: X ← n
X ← 0
X ← X+1
 ... (n veces)
X ← X+1

MACRO división entera: Q, R ← X1/X2 (Supongo X2>0)

```
U ← X2
R ← X1
Q ← 0

IF R < X2 GOTO H            Si X1 < X2, el resto es X1

[A]   U ← U-1               Resto R - X2
      R ← R-1
      IF U != 0 GOTO A
      Q ← Q+1               Sumamos 1 al cociente
      IF R < X2 GOTO H      Si R es menor que X2, el resto es R
      U ← X2                Si R es mayor o igual que X2, vuelvo a restarle X2
      GOTO A

[H]   FIN
```

si x%n ≠ 0, Z(C(x)-) = x/n. Si x%n = 0, Z(C(x)-) = x/n - 1

SOLUCIÓN:

```
      U ← n
      Y, R ← X/U            Divido x entre n
      IF R ≠ 0 GOTO H       Si el resto no es 0, Z(C(x)-) es el cociente de la división
      Y ← Y - 1             Si el resto es cero, Z(C(x)-) = x/n - 1
[H]   HALT
```

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Compra Coins y descarga sin publicidad.

5. Programa con variables numéricas que calcule f(x) = y con y=Z(a_iC(x)), donde C y Z son las codificaciones sobre un alfabeto de n símbolos.

abc = c*1 + 2*3 + 1*9 = 18
babc = 3*1 + 2*3 + 1*9 + 2*27 = 72

Dada una palabra $a_k \ldots a_1$, queremos comparar $Z(a_k \ldots a_1)$ con $Z(a_i a_k \ldots a_1)$

$Z(a_k \ldots a_1) = \sum_{i=1}^{k} Z(a_i) n^{i-1}$

$Z(a_i a_k \ldots a_1) = Z(a_i)\cdot + \sum_{i=1}^{k} Z(a_i) n^{i-1}$

Por tanto $Z(a_i C(x)) = Z(a_i)\cdot n^{|C(x)|} + x$

MACRO: V ← Z(C(X)-), la función del ejercicio anterior

MACRO: X ← n
X ← 0
X ← X+1
... (n veces)
X ← X+1

SOLUCIÓN: ($Z(a_i)$ es una variable global)

Z ← X
U ← 1
M ← n

IF X ≠ 0 GOTO A — Si X no es cero, su cadena asociada tiene longitud positiva
GOTO H

[A] X ← Z(C(X)-) — U ← $n^{|C(X)|}$
U ← U·M
IF X ≠ 0 GOTO A

[H] Y ← $Z(a_i)$·U — Y← $Z(a_i)\cdot n^{|C(x)|} + x$ (Uso macros ejercicio 1)
Y ← Y+Z

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Compra Coins y descarga sin publicidad.