**Lenguajes independientes del contexto**

**GRAMÁTICAS REGULARES**

Las expresiones regulares y los autómatas finitos nos proporcionan dos medios para especificar o definir lenguajes. Las expresiones regulares nos proporcionan una plantilla o patrón para las cadenas del lenguaje. Todas las cadenas se corres- ponden con un patrón en particular y dichas cadenas serán las únicas que forma- rán dicho lenguaje. Igualmente, un autómata finito especifica un lenguaje como el conjunto de todas las cadenas que lo hacen pasar del estado inicial a uno de sus estados de aceptación. También se podría intexpretar un autómata como un generador de cadenas del lenguaje, según se plantea a continuación. Un símbolo se genera al recorrer el camino etiquetado con dicho símbolo y que parte del es- tado actual al siguiente. Se empieza con la cadena vacía y se obtiene una cadena del lenguaje cuando el recorrido llega a un estado de aceptación.

Por ejemplo, se considera el autómata finito dado por el diagrama de tran- sición de la Figura 3.1. Este autómata finito acepta el lenguaje regular a (a u bv) b. Imaginemos que se comienza en el estado inicial y se atraviesa el diagrama de alguna forma. Cuando un camino va de un estado a otro, la “salida" es el símbolo que etiqueta dicho camino. Por tanto se podría obtener la cadena de salida aa2b pasando por los estados c¡\ -

<72

*- 1/3 - (b3 - qj - <*

75

. Se ve fácilmente que las cadenas generadas de esta forma serán aceptadas por este autómata fini- to. Es más, cualquier cadena aceptada por este autómata puede ser generada por este método.

www.FreeLibros.com

106

Ü2u. Í 02 Í'Ü iÍÜÍQ / B2g4'Í82Q 3u.iÍB2Q

**an**

a

b Figura 3.1

Obsérvese que todas las cadenas del lenguaje precedente estarán formadas por una a seguida de alguna “parte final”. Si hacemos que E represente la parte final, lo dicho se puede representar simbólicamente mediante S —> aE. La flecha —> se puede interpretar como “puede ser” o “se compone de”. La parte final de una cadena estará formada por una de las dos listas de aes o bes. Por tanto, para indicar las múltiples posibilidades que hay para E podemos escribir E —»A y E —> B. Las dos listas de aes y bes se pueden expresar como A —» aA junto con A —>b para indicar que una cadena de aes va seguida de una b o como B ~^bB junto con B —>b, para indicar que una cadena de bes va seguida de otra b.

En resumen, tendremos las siguientes expresiones

*S —^ aE E -^ A E ^ B A -> b A —> aA B ^ b B -^bB*

Estas expresiones pueden ser consideradas como reglas de sustitución para la generación de cadenas. El símbolo que se encuentra a la izquierda de la flecha se puede sustituir por la cadena de la derecha.

Por ejemplo, podemos generar aab empezando por S, sustituyéndola por aE, sustituyendo la E por aA y finalmente sustituyendo la A por b. Tendremos una secuencia de cadenas comenzando por S y terminando con aab. En cada paso, las letras mayúsculas (S, E y A) representan la parte de la cadena final que\* toda- vía no se ha generado. Bajo estas circunstancias tiene sentido interpretar la fle- cha en las expresiones precedentes como “es sustituido por".

www.FreeLibros.com

B2g4'Í82Q 7g02r2g072gÜ2Q 02B 1ugÜ2<Üu Ó U N

Finalmente, introduciremos el símbolo | que será interpretado como “o”. Si se usa este símbolo, las dos reglas E —> A y £ —>B, se pueden unir en E —>A \B. La colección precedente de reglas para generar cadenas puede volverse a escri- bir como sigue:

1. S —» aE

*2. E -\* A \B*

*3. A —> ciA | B*

*4. B -$b B \b*

La cadena cr’b se puede generar a partir de 5 aplicando primero la regla 1 para obtener aE. Entonces, aplicando la reglí1 2 se obtiene aA y aplicando la re- gla 3 se obtiene aaA y aaaA\ finalmente, se puede aplicar la segunda parte de la regla 3 para obtener aaab. Podremos escribir una descripción del proceso de ge- neración como

*S => aE => aA => aaA => aaaA => aaab*

donde la doble flecha => se interpreta como “deriva”, “produce” o “genera”. Usaremos la notación E^>w para indicar que la cadena w se deriva a partir de 5 en 0 o más etapas.

Obsérvese que en este modelo hemos introducido una colección de nuevos símbolos para representar las porciones de cadena que no han sido generadas. Cuando las cadenas han sido completamente generadas, estarán formadas en su totalidad por símbolos del alfabeto £, pero antes de llegar a esto se obtendrán ca- denas formadas por símbolos del alfabeto y por nuevos símbolos. Los nuevos símbolos se llaman no terminales, para indicar que deben ser sustituidos por símbolos del alfabeto antes de que la cadena haya sido totalmente generada. Por otro lado, los símbolos del alfabeto £ se llaman terminales, para indicar que no es posible que sean sustituidos. Obsérvese, también, que el símbolo usado para representar a una cadena que no ha comenzado a generarse, debe ser necesaria- mente un no terminal. Finalmente, observemos que hemos generado las cadenas del lenguaje de izquierda a derecha —en las cadenas de las etapas intermedias por las que se pasa al aplicar las reglas, los no terminales deben aparecer sola- mente en el extremo derecho. Esto refleja la forma en la que un autómata finito reconocería una cadena del lenguaje.

Daremos la siguiente definición:

Definición 3.1.1. Una gramática regular G es una 4-tupla G = (£, N, S, P). donde £ es un alfabeto, N es una colección de símbolos no terminales, S es un no terminal llamado símbolo inicial, y P es una colección de reglas de sustitución, llamadas

www.FreeLibros.com

108

Ü2u. Í 02 Í'Ü iÍÜÍQ / B2g4'Í82Q 3u.iÍB2Q

producciones, y que son de la forma A —> w, donde A e N y w es una cadena so- bre l u N que satisface lo siguiente:

1. w contiene un no terminal como máximo.

2. Si w contiene un no terminal, entonces es el símbolo que está en el ex-

tremo derecho de w.

El lenguaje generado por la gramática regular G se denota por L (G).

Por ejemplo, considérese la gramática regular G = (Z, N, S, P), donde

*1 = {a, b} N = {S ,A } P : S ^ b A*

*A —^ aaA | b 18*

Obsérvese que L (G) contendrá todas las cadenas de la forma ba2nb y ba2". Es decir, L (G) = b (a2)\* (b u e). Se puede demostrar, por inducción sobre n, que todas las cadenas de la forma ba2"b o ba2" están en L (G) y, por inducción sobre la longitud de una derivación, se demuestra que L (G) está contenido en b {a2)\* (b u £). (La etaba base es para una derivación de longitud 2).

De la definición se deduce que el lado derecho de una producción es una cadena de I ’ ( N u £). Obsérvese que £ puede ser el lado derecho de una pro- ducción. En el ejemplo precedente, la producción A —> £ acaba con la generación de una cadena (al igual que la producción A -> b) ya que se “borra” el no ter- minal A.

Dado que las producciones emparejan no terminales de N con cadenas de I (/Vu £), conviene representarlas como pares ordenados de iVx X\* (N u £). Por tanto, el par (x, y) de N x I\* (N u £) representa a la producción x —> y. Las producciones de P del ejemplo anterior se podrían representar mediante

*P={(S,bA),(A,aaA),(A,b),(A,e)}*

Si se llega al acuerdo de escribir los no terminales con letras mayúsculas y los terminales con letras minúsculas y además se conviene que S se use como símbolo inicial, entonces una gramática regular puede ser completamente espe- cificada por medio de sus producciones. Por ejemplo, S —> a5| b especifica com- pletamente la gramática www.FreeLibros.com

regular que genera el lenguaje a b.

109

2HJ-Á,Á,V> 3J ¡O CJÁÁ,ÍQ ex0

3.1.1. Usar las reglas de la Figura 3.1 para derivar ab, ab3, aa b. ¿Es posible derivar

*ababl*

3.1.2. Supongamos que tenemos las reglas S —>aS\bT y 7" —> aa. Dar una derivación para abaa, aabaa y aaabaa. Probar como se deriva akba2 para k> \. ¿Es posible derivar las cadenas baa, b o a al

3.1.3. Obtener una gramática regular para los siguientes lenguajes:

*(a) a\*b u a*

*(b) a\*b\jb\*a*

*(c) (a\*bUb\*a)’*

3.1.4. La gramática regular dada por (.

*S —^ bA | flfíle A —»abaS B —>babS*

genera un lenguaje regular. Obtener una expresión regular para este lenguaje.

3.1.5. En nuestra definición de gramáticas regulares se dijo que si en el lado derecho de una producción hay un no terminal, éste debe estar situado en el extremo de- recho. Esto corresponde a la generación de cadenas de izquierda a derecha. Por esta razón, una gramática regular también puede llamarse gramática regular por la derecha. Una gramática regular por la izquierda es aquella cuyas cade- nas son generadas por la derecha, es decir, las producciones son pares de iVx(NuE) X\*.

(a) Obtener una gramática regular por la izquierda para el lenguaje

*{anbaa | n > 0).*

(b) Obtener las gramáticas regulares por la derecha y por la izquierda para

{w e {a,b, c}\* | iv termina en b y toda c va seguida por una a]

(c) Para toda gramática G = (X, N, S, P) que sea regular (por la izquierda o pol-

la derecha), se puede definir la inversa de G como G1 = (X, N, S. P'), donde

*P'={(A,x')\(A,x)e P]*

Por tanto, siA-^aB es una producción de G, entonces A —í> Ba es una pro- ducción de G1.

Supongamos que G es una gramática regular por la derecha.

www.FreeLibros.com

B2g4'Í82Q 7g02r2g072gÜ2Q 02B 1ugÜ2<Üu

110

Ü2u. Í 02 Í'Ü iÍÜÍQ / B2g4'Í82Q 3u.iÍB2Q

i. Probar que G1 es una gramática regular por la izquierda.

ii. Probar que w e L (G) si y sólo si w1 e L (G1) por inducción sobre el

número de producciones usadas para obtener w.

Se puede deducir de la parte (c) que la clase de los lenguajes genera- dos por gramáticas regulares por la izquierda es la misma que la clase de los lenguajes generados por gramáticas regulares por la derecha. Por eso habitualmente, el término gramática regular se aplica para referirse a cual- quier gramática ya sea regular por la izquierda o regular por la derecha.

ex LGb7 E5ubC G2LMdbG2C D d2ILMb62C G2LMdbG2C

Supongamos que L es un lenguaje regular. Se puede obtener una gramática regu- lar que genere L por medio de un AFD M = (Q, £, s, F, 8) para el cual L - L (M). Definimos G = (N, X, S, P) por

*N - Q I = Z S = s P = {(q, ap)\§ (q, a) p} u {{q, e)|<?e F]*

Es decir, q —> ap siempre que 8 (q, a ) - p y q —^>zúq es un estado de acep- tación del AFD.

Por ejemplo, el AFD dado en la Figura 3.2 acepta el lenguaje a\*b. La gra- mática regular correspondiente tiene las producciones

*q\ ~^aq\\bq2 q2 -> aq3 \ bq*

*3 1 8 qi aq?, | bq-i*

En esta gramática los q¡ son no terminales (una ruptura con la notación usual, la cual puede ser restablecida sólo con renombrar los q¡ y cj\ como símbo- lo inicial).

Obsérvese que w e L (M) para w = ai 02 ... o„ significa que

8 (s, c?i 02 ••• — p a a

a. b Q2

t)

a, b

Figura 3.2

www.FreeLibros.com

B2g4'Í82Q 7g02r2g072gÜ2Q 02B 1ugÜ2<Üu Ó Ó Ó

para algún p e F. Si escribimos q¡

+ 1

= 8 (q,, o¡) con q\ = s, entonces se obtiene

*8 (s, Oí*

02

*... 0„) = b(qi,Oi 02 ... o„)*

**= 8 (<?2> 02 ... On) = 8 (#3, 03 ... 0„)**

= 8 (<7», 0/j) = /? e F

Ahora, puesto que q¡+ i = 8 (<?,•, 0;), se obtiene que q¡ -»

0

/ g,+ i pertenece a G y, por tanto, (ya que = <yi)

5 = q\ => 0| q2 =>

0

l

02

*q*

3

**=> 0] 02 ... Onp => 01 02 ... On**

Así que w e L (M) implica que w es generada por G; es decir, ¿(M )cL (G ).

A la inversa, si w es generada por G, mediante la derivación siguiente

*q\ => 0i qi*

=> 0i 02 <73

=> 0 1 02 ... 0« p => 01 02 ... 0«

entonces en M tendremos

8(5, 01 02 ... 0„) = 8(qi,0i 02 ... o„)

. t óc

72

g i & & & ig,l = 8(q3, 03 ... On)

= 8 (<7h, 0») - p G F

\ (ya que 5 = qi). Asi, que w e L (G) implica que w e L (M), con lo que se tiene que L (G) c L (M). Entonces se deduce que L (G) = L (M).

www.FreeLibros.com

Ó Ó Ü2u. Í 02 Í'Ü iÍÜÍQ / B2g4'Í82Q 3u.iÍB2Q

También es posible partir de una gramática regular G y construir un AFN M de forma que L (G) = L (M). Sea G = (N, X, S, P) una gramática regular; se defi- ne M - (Q, X, .y, F, A) mediante

Q = N u {/}, donde/ es un símbolo nuevo

*s - S F = \ f}*

y A se construye como se indica a continuación, a partir de las producciones de P:

1. Si A —» ai ... anB es una producción de P con A y fí como no termina-

les, entonces se añadirán a Q los nuevos estados qi, <

72

, <?«-

1

y las transformaciones siguientes

**A (A, ai ... a„) = A(<?i,a2... a„) = ... =A(^,;\_i,a„) = B**

2. Si A —> ai ... c„ es una producción de P, entonces se añadirán a Q los

nuevos estados q\, <

72

*, qn-*

1

y a A, las transiciones siguientes

**A (A, a i ... a„) = A (¡yi, a2... a,;) — ... — A (q„ \_**

1

S

La construcción de A se puede concebir a partir de cómo estén etiquetadas las aristas del diagrama de transición correspondiente a M y entonces se añadi- rán los estados necesarios para cada uno de los símbolos de la cadena. Por tanto, si A —> ai ... a„ S, primero podríamos etiquetar las aristas entre A y B con ai ... G„ y después añadir n - 1 estados nuevos, en la arista resultante:

9 ------------> Q] • ------------> 02 O

9

O,,

**, a(í) —**

• \* ------------> • A B

Por ejemplo, la gramática regular

*S —> aB | bA | e A —> abaS B —> babS*

daría como resultado el AFN cuyo diagrama de'transición se muestra en la Fi- gura 3.3. »

Si G es una gramática regular y w e L (G) con vv = ai ... a,„ entonces para los no terminales Aj, A2, ..., A ,

1

, se tiene la derivación

www.FreeLibros.com

113

S =» ai A\ => ... => 01 ... Gn- l An- l => 01 ... 0,¡

y entonces en el AFN resultante de esta construcción se tendrá

*A (s,*

0 1

... G„) = A (Ai, 02 ••• 0n) = = A(A„\_ i, 0„) = /

Por lo tanto, we L( M) . A la inversa, si A (s, 0i ... 0„) =/, entonces 5 =>

0 1

... 0,„ con lo que w e L (G). Luego L (G) = L (M). Aunque hemos demostrado las técnicas de construcción más usuales, en re- alidad hemos demostrado mucho más:

Figura 3.3

Teorema 3.2,1. L es regular si y sólo si es generado por una gramática regular.

Por tanto, tenemos tres métodos generales de especificación de lenguajes: las expresiones regulares, los autómatas finitos y las gramáticas regulares.

2HJ-Á,Á,V> 3J ¡O CJÁÁ,ÍQ ex x

3.2.1. Construir una gramática regular para el lenguaje regular aceptado por el autó-

mata finito de la Figura 3.4.

3.2.2. Construir un AFN para la gramática regular

*S->aS\bB\b B —^ cC C —^ ÜS*

www.FreeLibros.com

B2g4'Í82Q 7g02r2g072gÜ2Q 02B 1ugÜ2<Üu

114

Ü2u. Í 02 Í'Ü iÍÜÍQ / B2g4'Í82Q 3u.iÍB2Q

3.2.3. Construir un autómata finito para la gramática regular

*S —> abA | B | baB \ e A->bS\b B —^ ciS*

3.2.4. Obtener una gramática regular para el lenguaje

L= jive {a, &¡\* | w no contiene la subcadena cía i

3.2.5. Obtener una gramática regular para L = {a"b" \ n > 0}.

3.2.6. Una producción regular por la izquierda es una producción de la forma A —> Bw, donde A y B son no terminales y w es una cadena de terminales. Una producción regular por la derecha es una producción de la forma A —> wB. Por tanto, las gramáticas regulares por la izquierda (véase Ejercicio 3.1.5) y las gra- máticas regulares por la derecha contienen solamente producciones regulares por la izquierda y producciones regulares por la derecha, respectivamente. Pro- bar que una gramática regular no puede contener ambos tipos de producciones.

LGb7 E5ubC 5Ij2í2Ij52IE2C j2d upIE2SEp

Recordaremos que en nuestra definición de gramáticas regulares se requiere que el lado derecho de todas las producciones contenga al menos un no terminal. Es más, cuando un no terminal está presente, debe aparecer al final de la cadena (final izquierdo o final derecho, dependiendo de sr es una gramática regular por la izquierda o por la derecha). Para expresar esto formalmente, se requiere que las producciones satisfagan P c W x I 4 (N u e) (o, en el caso de la regularidad por la izquierda, P c /V x (N u e) X\*). Este requerimiento restringe en gran me-

www.FreeLibros.com

115

dida la manera en la que se pueden formar las producciones y, en consecuencia, restringe las clases de lenguajes que se pueden especificar.

Supongamos que se permite que P

*Q*

*N*

*X*

*(N*

*KJ*

Z)\*, de forma que las pro- ducciones puedan tener cero, uno o más no terminales que aparezcan en cual- quier lugar del lado derecho de las mismas. Por ejemplo, la gramática dada por

*5 —> afi | M A H> a\aS\bAA B —> b\bS\aBB*

es una gramática de este tipo. Observe que, en definitiva, esta gramática no es una gramática regular. Por otro lado, todas las gramáticas regulares satisfacen este nuevo requerimiento en lo que respecta a la forma en la que se construyen las producciones y, por tanto, son gramáticas de este tipo. De esta forma, tendre- mos más de un tipo general de gramáticas.

*Definición 3.3.1. Una gramática independiente del contexto (GIC) es una 4-tupla*

*G ^ Í K X ^ P l*

donde N es una colección finita de no terminales, £ es un alfabeto (también co- nocido como conjunto de terminales), 5 es un no terminal determinado que se llama símbolo inicial y P c A 'x (N u X)\* es un conjunto de producciones.

*> El lenguaje generado por la GIC G se denota por L (G) y se llama lenguaje independiente del contexto (LIC).*

Por ejemplo, puesto que toda gramática regular es una GIC, se tiene que todo lenguaje regular es un LIC.jy

Al igual que una gramática regular, una GIC es una forma de probar cómo se generan cadenas en un lenguaje. Como con las gramáticas regulares, usare- mos la notación => para indicar el acto de generar como opuesto a — el cual es parte de una regla de producción. Cuando derivamos una cadena, los no termina- les representan la parte de la cadena que todavía no se ha generado. En el caso de las gramáticas regulares, la parte de la cadena no generada siempre aparece al final. En las GIC que no son gramáticas regulares, puede haber más de un trozo no generado y pueden aparecer en cualquier lugar de la cadena. Cuando la deri- vación se completa, todos los trozos no generados habrán sido sustituidos por cadenas (posiblemente vacías) de símbolos terminales. Consideremos la GIC dada por \*

www.FreeLibros.com

B2g4'Í82Q 7g02r2g072gÜ2Q 02B 1ugÜ2<Üu

Ó Ó L Ü2u. Í 02 Í'Ü iÍÜÍQ / B2g4'Í82Q 3u.iÍB2Q

La inducción sobre n prueba que esta gramática independiente del contex- to genera el lenguaje independiente del contexto {a"bn\n>0}. Por el Capítulo 2 sabemos que este lenguaje no es regular. Por tanto, hay lenguajes inde- pendientes del contexto que no son lenguajes regulares. Es decir, el conjunto de los lenguajes independientes del contexto contiene al conjunto de los lenguajes regulares.

Dedicaremos bastante tiempo al estudio de las gramáticas independientes del contexto y los lenguajes independientes del contexto. Sin embargo, antes de continuar, debemos mencionar otras formas de expresar las gramáticas regula- res. Al generalizar, las gramáticas independientes del contexto, debemos elimi- nar todas las restricciones con respecto al lado derecho de las producciones, per- mitiendo que el mismo pueda estar formado por cualquier cadena sobre Nu~L. Lo único que debemos tener en cuenta en la generalización es la parte izquierda de las reglas de producción. Una gramática de estructura de frase es aquella en la que los lados izquierdos de las reglas de producción pueden estar formados por cualquier cadena no vacía sobre iV u !, las cuales contienen algún no termi- nal. Por tanto, para una gramática de estructura de frase, la colección de reglas de producción P satisface

P c (A fu 1)\*N (N u 2)\* x(Wu I)\*

Las gramáticas de estructura de frase se conocen como de tipo 0 o gramáti- cas no restringidas.

El término independiente del contexto, cuando se aplica a gramáticas, su- giere que debería haber gramáticas que dependieran del contexto. Las gramáti- cas dependientes del contexto son gramáticas de estructura de frase, en las cua- les las producciones se restringen a a —> P, tal que I al < I f}|. Hay una forma normal para estas gramáticas, en la cual toda producción es de la forma «i A (X 2

—> ai (3 ai con (3 ^e. Tales producciones permiten que el no terminal A sea reemplazado por la cadena P, sólo cuando A aparezca en el “contexto” de a, y a 2.

Las gramáticas dependientes del contexto no pueden generar tantos lengua- jes como las gramáticas de estructura de frase, aunque permiten que las deriva- ciones se realicen de forma predecible. Sin embargo, obsérvese que, puesto que |S| = 1 y leí =

0

, es imposible derivar la cadena vacía en una gramática que sea verdaderamente dependiente del contexto. A menudo, los lenguajes de progra- mación se crean para ser dependientes del contexto con el fin de simplificar el proceso de la compilación.

www.FreeLibros.com

B2g4'Í82Q 7g02r2g072gÜ2Q 02B 1ugÜ2<Üu Ó Ó N

2HJ-Á,Á,V> 3J ¡O CJÁÁ,ÍQ exe

3.3.1. Dada la siguiente gramática independiente del contexto

*S ^ A A A -» AAA \a\bA\Ab*

(a) Obtener una derivación para la cadena b2aba2ba.

(b) Probar cómo puede obtenerse una derivación para

*bm' ab"'2 a... b'"2" ab'”2" \* para todo n >*

0

*y m¡, m2..... m2n*

+ 1

>

0

.

3.3.2. La gramática G independiente del contexto dada por

*S —>aSb|aSa|bSa\bSb 1*

6

no es una gramática regular, aunque L (G) es un lenguaje regular! Obtener una gramática regular G' tal que L (G') = L (G).

3.3.3. Obtener una gramática independiente del contexto para cada uno de los siguien-

tes lenguajes independientes de contexto:

*(a) {a"’b" | m > n}*

(b) {w c {a, b} \* | w tiene el doble de aes que de bes}

*(c) {ánbn\n <m < 2n}*

*(d) {a!"b"cpd q\m +n> p + q)*

Gypd2C j2 j2G5Nbu5 I p j2 bI d5C5C D b7y5L}2jbj

Cuando una cadena se deriva mediante una gramática independiente del contex- to, el símbolo inicial es sustituido por alguna cadena. Los no terminales de esta cadena son sustituidos uno tras otro por otra cadena, y así sucesivamente, hasta que se llega a una cadena formada sólo por símbolos terminales. No se puede re- alizar ninguna sustitución más, puesto que no hay no terminales que puedan ser sustituidos. A veces, es útil realizar un gráfico de la derivación, que indique de qué manera ha contribuido cada no terminal a formar la cadena final de símbo- los terminales. Tal gráfico tiene forma de árbol y se llama árbol de derivación (o árbol de análisis).

Un árbol de derivación para una derivación dada se construye creando un nodo raíz que se etiquetó con el símbolo inicial. El nodo raíz tiene unos nodos hijos para cada símbolo que aparezca en el lado derecho de la producción usada para reemplazar el símbolo inicial. Todo nodo etiquetado con un no terminal también tiene unos nodos hijos etiquetados con los símbolos del lado derecho de

www.FreeLibros.com

118

Ü2u. Í 02 Í'Ü iÍÜÍQ / B2g4'Í82Q 3u.iÍB2Q

la producción usada para sustituir ese no terminal. Los nodos que no tienen hijos deben ser etiquetados con símbolos terminales.

Consideremos la gramática independiente del contexto

*S->AB A a A \ a B-> bB\b*

La cadena aabbb puede ser derivada mediante

*S => AB => AbB=> AbbB => Abbb=> aAbbb => aabbb*

En la Figura 3.5 se presenta un árbol de derivación para esta derivación. Co- menzamos en la raíz S y generamos los hijos A y B. A y B son raíz del subárbol correspondiente a la parte de la cadena final que ellos generan. Obsérvese que todos los nodos hoja están etiquetados con símbolos terminales. Si se leen las hojas de izquierda a derecha, se obtiene la cadena aabbb.

b Figura 3.5

Finalmente, obsérvese que hay muchas derivaciones posibles para la cadena aabbb, las cuales también tienen el árbol de derivación anterior. Por ejemplo,

*S =$ AB => aAB=> aaB => aabB => aabbB => aabbb*

y

*S => AB => aAB=$ aAbB => a A bbB => aAbbb => aabbb*

Para esta cadena y esta gramática, todas las derivaciones de aabbb tienen el mismo árbol de derivación. Sin embargo, no tiene porque cumplirse siempre. Para verlo, considérese esta gramática

\*

*S->SbS\ScS\a*

Podemos derivar la cadena abaca de dos formas distintas como sigue:

www.FreeLibros.com

119 B2g4'Í82Q 7g02r2g072gÜ2Q 02B 1ugÜ2<Üu

*1. S=> SbS=> SbScS => SbSca => Sbaca => abaca*

*2. S => ScS => SbScS => abScS => abacS => abaca*

El árbol de derivación para la derivación 1 es

a I a (

mientras que el árbol para la derivación

2

es

a a

Obsérvese que los dos árboles son distintos, aunque las cadenas producidas son la misma. (La cadena derivada corresponde a los nodos hoja y se llama pro- ducto del árbol de derivación).

Una gramática se dice que es ambigua si hay dos o más árboles de deri- vación distintos para la misma cadena. Una gramática en la cual, para toda cade- na w, todas las derivaciones de w tienen el mismo árbol de derivación, es no am- bigua.

La ambigüedad puede ser un problema para ciertos lenguajes en los que su significado depende, en parte, de su estructura, como ocurre con los lenguajes naturales y los lenguajes de programación. Si la estructura de un lenguaje tiene más de una descomposición y si la construcción parcial determina su significa- do, entonces el significado es ambiguo. Consideremos la sentencia “Juan vio a un hombre con un telescopio”. El significado de esta sentenciares ambiguo debi- do a que “con un telescopio” puede describir al hombre que vio Juan o a la téc- nica que Juan empleó para ver al hombre. \*

Consideremos otro ejemplo de ambigüedad que obscurece el significado de las expresiones. Sea la siguiente gramática de asignaciones de expresiones:

www.FreeLibros.com

120

Ü2u. Í 02 Í'Ü iÍÜÍQ / B2g4'Í82Q 3u.iÍB2Q

*A ^ I ] = E*

*I a \_ \ b \ c*

*E - \* E ± E \E \* E \W \_ \ I*

Los símbolos terminales han sido subrayados. La cadena a := b + c \* a es una cadena de este lenguaje de sentencias de asignación. Hay dos árboles de derivación distintos para ella (véase Figura 3.6)

*A A*

*b I I I I a*

c a b e

Figura 3.6

Si pretendemos determinar cómo se calcula el valor de la derecha del opera- dor de asignación (el símbolo :=), se obtienen dos resultados posibles, b + (c \* a) o (b + c) \* a. En general, estos resultados no son iguales.

En algunos casos, si la gramática es ambigua, se puede encontrar otra gra- mática que produzca el mismo lenguaje pero que no sea ambigua. Por ejemplo, la gramática

*S ^ A \ B A —^ a B —^ ci*

es ambigua porque tiene dos árboles de derivación para la cadena a. Una gramá- tica equivalente que no es ambigua es

*S —^ ci*

Si todas las gramáticas independientes del contexto para un lenguaje son ambiguas, se dice que el lenguaje es un lenguaje independiente del contexto in- herentemente ambiguo. El lenguaje '

*L - {alb->ck | i - j o j = k}*

www.FreeLibros.com

121

es inherentemente ambiguo. Intuitivamente, una gramática para L debe tener una clase de árbol de derivación para generar las cadenas para las cuales i= j y otro para las cadenas en las cuales j = k. Si una cadena tiene i = j - k, tendrá dos deri- vaciones.

Vimos anteriormente que una\* cadena dada no puede tener más de una deri- vación igual en una gramática independiente del contexto no ambigua. Las deri- vaciones distintas corresponden a la elección de distintos no terminales a expan- dir. Por convención, dos formas de generar una cadena tienen una única salida. En una derivación por la izquierda el no terminal que se expande es, siempre, el del extremo izquierdo. Por tanto para la gramática

*S —> SbS\ScS\ a*

una derivación por la izquierda de abaca será

*S => ScS=> SbScS => abScS => abacS => abaca*

Una derivación por la derecha es aquella en la cual el no terminal que se expande es el del extremo derecho. Por tanto

*S => ScS=> Sea => SbScá => Sbaca => abaca*

es una derivación por la derecha de la cadena dada. Obsérvese que las dos deri- vaciones tienen el mismo árbol de derivación:

*a a*

Este árbol de derivación también es compartido por otras derivaciones. En esta gramática

*S => SbS => abS => abScS => abacS => abaca ■\**

es una derivación por la izquierda distinta de la precedente. La presencia de dos derivaciones por la izquierda distintas se corresponde con la existencia de dos árboles de derivación distintos. Por tanto, una gramática ambigua se caracteriza por tener dos (o más) derivaciones por la izquierda para la misma cadena.

www.FreeLibros.com

B2g4'Í82Q 7g02r2g072gÜ2Q 02B 1ugÜ2<Üu

123

Nos gustaría establecer las restricciones necesarias para que las produccio- nes se formen de manera que el árbol de derivación resultante no sea necesaria- mente complejo o inútilmente sencillo. A la vez, no se pretende constreñir la formación de producciones hasta el punto de que no se pueda generar ningún lenguaje independiente del contexto a partir de los conjuntos de producciones que cumplan las restricciones. Pretendemos encontrar un modelo formal están- dar (o una forma normal) para las producciones.

Como primer paso en el desarrollo del modelo, necesitamos limpiar las gra- máticas para eliminar las producciones y símbolos inútiles. Consideremos la gramática independiente del contexto del ejemplo siguiente:

**Ejemplo 3.5.1**

*S^A a \B \D B->b A —» aA\bA\B C —> a bel*

Obsérvese que C nunca formará parte de una derivación que parta del sím- bolo inicial, es decir, no existe S => aC(3 para toda cadena a y p de (WuZ)\*. Por consiguiente, el símbolo C y la producción

*C ¡U abd*

son inútiles en el sentido de que nunca podrán contribuir a la generación de una cadena de L (G). El símbolo D se obtiene a partir de S pero nunca deriva una ca- dena de símbolos terminales y, por tanto, nunca formará parte de una derivación de una cadena de terminales. Por tanto, D también es inútil, aunque por distinta razón. Por otro lado, el símbolo B es, en cierto modo, redundante, ya que deriva únicamente un símbolo terminal; por tanto, las derivaciones S => B => b y A B ==> b podrían ser reducidas de forma que S => b y A => b, y eliminándose B. Finalmente, obsérvese que si se elimina la producción C -» abd, el símbolo terminal d no puede aparecer en ninguna cadena de terminales generada me- diante la gramática resultante.

Por tanto, hemos identificado muchos aspectos en los que las gramáticas in- dependientes del contexto pueden ser depuradas. Cualquiera de e,sos problemas pueden ser eliminados sin afectar a la capacidad de generación de la gramática. Primero se eliminarán los no terminales que no deriven cadenas de terminales, tales como D en el Ejemplo 3.5.1.

Sea G = (N, Z, S, P) una gramática independiente del contexto. Transforma- remos G en G' = (N \ Z, S, P') de forma que L (G) = L (G') y, para todo A e N',

www.FreeLibros.com

B2g4'Í82Q 7g02r2g072gÜ2Q 02B 1ugÜ2<Üu

124

Ü2u. Í 02 Í'Ü iÍÜÍQ / B2g4'Í82Q 3u.iÍB2Q

se obtenga que A ==> w para algún w e 2T. Para realizarlo, construiremos iterati- vamente el nuevo conjunto de no terminales N ' y el nuevo conjunto de produc- ciones P' como sigue:

**Algoritmo 3.5.1.**

1. Inicializar N ' con todos los no terminales A para los que A —> w, es una

producción de G, con w e X\*.

2. Inicializar P' con todas las producciones A —> vv para las cuales A e N '

*y w e 1 \*.*

3. Repetir

Añadir a N ' todos los no terminales A para los cuales A —» vv, para algún vv e (/V' u £)\* que sea una producción de P y añadirla a P'.

hasta que no se puedan añadir más no terminales a N'.

Obsérvese que el bucle del paso 3 termina, ya que N y P son finitos. Esen- cialmente, lo que estamos haciendo es recorrer hacia arriba todos los posibles ár- boles de análisis a partir de las cadenas de terminales, anotando los no termina- les (y las producciones) que se encuentren. Todo no terminal (y producción) que no aparezca en N', no contribuirá a formar una subcadena de cualquier cadena “final’" de terminales que sea generada por la gramática. Por tanto, su elimina- ción no altera el lenguaje generado.

Por ejemplo, en la gramática del Ejemplo 3.5.1, esperábamos eliminar el no terminal D. Después de aplicar el Algoritmo 3.5.1 a esta gramática, obtendremos la siguiente:

*iS —^ Aci ¡ B A ^ a A \b A \B B —> b C —> abd*

Las gramáticas independientes del contexto se han definido de forma que se permite que existan producciones que tengan e en el lado derecho. El Algorit- mo 3.5.1 trata e como una cadena de un terminal. Por tanto, al transformar la

• gramática

... •. .. ■• ■ • •

*f . ' S —^ ÜA I £ \* A -> a A | bB | £ B -^b B*

www.FreeLibros.com

125

se obtendrá la gramática

*S —> ciA | £ A -> a A 1*

8

Las producciones de la forma A —> e se llaman producciones e. A veces son necesarias, pero otras no son ni necesarias ni deseables. Podremos eliminarlas cuando no sean necesarias.

A menudo, tras la transformación de una gramática, nos quedan produccio- nes como C —> abd que no se usan. La razón por la cual dicha producción se en- cuentra en la gramática transformada es que, de la aplicación de dicha produc- ción, se obtiene una cadena de terminales. Sin embargo, su presencia no es de- seable ya que nunca se podrá derivar una cadena a partir del símbolo S que contenga el no terminal C. Es más, puesto que el símbolo terminal d sólo apare- ce en una cadena que se obtenga a partir de dicha producción, puede ser elimina- do del alfabeto sin que el lenguaje generado sea alterado. El siguiente algoritmo elimina aquellos terminales y no terminales que no aparezcan en las cadenas que se deriven a partir de S. La gramática transformada resultante garantiza que un símbolo X será un terminal o no terminal de dicha gramática si y sólo si S => C/.X P para algunas cadenas a y (3 sobre (N u Z)\*.

Sea G = (N, Z, S, P) una gramática independiente del contexto. Transforma- remos G en la gramática G' = (N ', Z', S, P') de forma que L (G) = L (G') y para todo X e iV 'n l', se tenga que S => a X P para las cadenas a y p de (N' n Z')'. Para realizarlo, se construirán iterativamente los conjuntos de terminales, no ter- minales y producciones de la manera siguiente:

**Algoritmo 3.5.2.**

1. Inicializar N' de forma que contenga el símbolo inicial S, e inicializar

P' y l 'a 0.

2. Repetir

Para A e N', si A —> w es una producción de P, entonces:

1. Introducir A —» w en P'. 2. Para todo no terminal B de w, introducir B en N'. 3. Para todo no terminal

0

de w, introducir a en Z'.

hasta que no se puedan añadir nuevas producciones.

Obsérvese que, puesto que P, N y Z son finitos, el bucle de la etapa 2 siem- pre termina. El algoritmo ha sido diseñado para tener en cuenta todos los termi- nales y no terminales que sean accesibles desde S. Si un terminal o un no termi- nal no puede ser conseguido a partir de S, nunca será incluido en N ' o Z'. Si un

www.FreeLibros.com

d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

126

E2pG b j{ bME 7bEbC D d2ILMb62C 9pG7bd2C

no terminal no es accesible, entonces todas las producciones que lo tengan en su lado izquierdo serán también excluidas.

Consideremos la gramática del Ejemplo 3.5.1 que fue transformada median- te el Algoritmo 3.5.1 en

*S -\*A a\B B -\* b A ^ a A \b A \B C —» abd*

La gramática obtenida al aplicar el Algoritmo 3.5.2 es

*S-> Aa\B ' A —> a A | bA | B*

*B -> b*

Obsérvese que la producción C —> abd ha sido eliminada, junto con el no terminal C y el terminal d.

Hay que tener en cuenta que es importante el orden en el que los dos algo- ritmos precedentes son aplicados a una gramática. Consideremos la gramática

*S —> AB | a F*

—^ £7

Si aplicamos el Algoritmo 3.5.1 antes que el Algoritmo 3.5.2 obtendremos un resultado distinto al que obtendríamos si aplicamos primero el Algoritmo 3.5.2 y después el Algoritmo 3.5.1.

*S —^ AB j ci*

Al\*. 3.5.1 Al\*. 3.5.2 A —^ a

*S AB | (2 | Al\*. 3.5.2. 5 —> AB | Cl Al\*. 3.5.1 S —) Cl A —>a j A —> a j A ^ a*

Ahora dirigiremos nuestra atención a las producciones s. Dichas prodicio- nes son de la forma A —» £. Indudablemente, si £ e L (G), no podremos eliminar tales producciones para que e pueda ser generado por la gramática. De esto se deduce que, si £ no está en L (G), todas las producciones £ pueden ser elimi- nadas.

Se dice que un no terminal A es anulable si A =^> £. Para la eliminación de las producciones £, es crucial identificar los no terminales anulables. El siguien-

www.FreeLibros.com

127

te algoritmo identifica el conjunto 5\£, de todos los no terminales anulables en una gramática independiente del contexto G = (N, Z, S, P).

**Algoritmo 3.5.3.**

1. Inicializar con todos los no terminales A para los cuales existe una

producción e, A —» e.

2. Repetir:

Si B —> w para algún w e (JV uI)"y todos los símbolos de w están en íAÍ, añadir B a

hasta que no se añadan más no terminales a .

Por ahora, sólo nos ocuparemos de las gramáticas independientes del con- texto G = (N, Z, S, P), para las que L (G) no contiene a e. Una vez que han sido identificados los no terminales anulables, se modifican las reglas de producción con el fin de poder eliminar las producciones e. Esto se realiza sustituyendo pro- ducciones de la forma B ^>X\X

2

... X„ por las producciones que se formen al eliminar los subconjuntos de X¡ que son anulables. Se debe tener cuidado en no incluir B -» e, incluso si todos los X¡ son anulables.

Se crea el nuevo conjunto de producciones P' como sigue:

Si B —> X\ X2 ■■■ X,¡ es una producción de P, entonces en P' introducire- mos todas las producciones de la forma B -» Y\ Y2 ... Y„, donde las Y¡ satisfagan:

Y¡ = X¡ si X, no es anulable.

*Y¡ = X¡ o*

8

si X¡ es anulable.

Y¡ no es e para todo i (es decir, no se introduce en P' ninguna producción de la forma B —»e)

Es importante señalar que a partir de una producción B —> X\ X2 ... X„ de P, se pueden conseguir nuevas producciones en P'. Por ejemplo, si B —»X¡ X2 y tanto X\ como Xj son anulables, se podrían obtener las producciones

*B ^> X x\X2\XxX2*

**\***

Consideremos la gramática G:

*\* S —> a A A —> a A I e*

www.FreeLibros.com

d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

128

E2pG b j2 bME 7bEbC D d2ILMb62C 9pG7bd2C

Obsérvese que A es el único no terminal anulable (y que e £ L (G)). Si con- sideramos la producción S —> aA, tendremos que X\ = a y X2 =A. Por tanto, aña- diremos a la nueva colección las producciones S —>a\aA. La gramática que re- sulta, tras considerar todas las producciones originales, será

*5 —»a A | a A aA | a*

Obsérvese que se ha eliminado la producción e, A —> £. En una gramática independiente del contexto, si L (G) contiene e, se pueden eliminar todas las producciones £ de G menos una. Primero, se eliminan todas las producciones £ de G. Esto transformará la gramática G en G' para la cual L (G') = L (G) - {£}. Después se añade la producción S —> £, la cual restituirá £ al lenguaje generado.

Las producciones de la forma A —» B, donde A y B son no terminales, se lla- man producciones unitarias o no generativas. La presencia de producciones uni- tarias no indica, necesariamente, que un símbolo es inútil. Sin embargo, las pro- ducciones unitarias hacen que la gramática independiente del contexto sea inne- cesariamente compleja.

Por ejemplo, la aplicación de una producción de la forma A —>B simple- mente renombra un no terminal y añade un paso más a la derivación. Cualquier cadena que sea derivable a partir de B también lo será a partir de A. Por tanto, se puede eliminar ese paso extra saltando por encima de B. Por ejemplo, si las pro- ducciones de A y B son

A - ^ B B —^ wj ¡ C

donde C e N y w\ e (/V u L )\ podemos eliminar la producción A —> B e incluir la producción A —»vvi | C. Obsérvese que, al eliminar la producción unitaria A —> B, se introduce la producción unitaria A —> C. Se podría repetir este proce- so hasta que no existiera ninguna producción unitaria en la gramática, pero se podría realizar otro planteamiento con el fin de eliminar la circularidad de este proceso.

Obsérvese que, en el proceso precedente, la producción unitaria A —» C se obtiene como resultado de las producciones A —»B y B -» C de la gramática ori- ginal. Si conocemos todos los no terminales X tales que A =?> X, solamente me- diante producciones unitarias, se podría entonces evitar introducirlas repetida- mente e ir eliminando dichas producciones unitarias una a una. Para ver cómo se puede realizar este proceso, supongamos que tenemos las producciones

www.FreeLibros.com

129

*A B B —> C \ w\ C D D —> W '2*

Entonces se tiene que A => B => C => D. Obsérvese que las producciones A —> vv 1

1W 2

permiten que de A se deriven las mismas cadenas que se derivaban con las cinco producciones originales. Las nuevas producciones se obtienen a partir de las producciones no unitarias del conjunto de producciones original, creando una producción A —» y para toda producción no unitaria X —» y, donde X e {B,C,D}. A continuación presentaremos esta técnica de una forma más precisa.

Primero, para A e N se define

Unitario (A) = {B e /V| A =^> B usando solamente producciones unitarias}

(Obsérvese que A e Unitario (A) puesto que A =^> A mediante 0 produccio- nes). Sea G = (N, E, 5, P) una gramática independiente del contexto que no ten- ga producciones e. Construiremos una gramática independiente del contexto equivalente G' = (N, Z, S, P') en la que P' no contenga producciones unitarias, como se describe a continuación:

1. Inicializar P' de forma que contenga todos los elementos de P.

2. Para cada A e N, obtener el conjunto Unitario (A).

3. Para cada A para el cual Unitario (A) ^ {A}

Para cada B e Unitario (A)

Para cada producción no unitaria B —> w de P

añadir A - ) w a P /.

4. Eliminar todas las producciones unitarias que haya en P'.

Por ejemplo, en la gramática

*S —» A | Aa A -+ B B C\b \* C —^ D | cib*

*D ^ b*

tenemos

www.FreeLibros.com

d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

130

E2pG b j2 bME 7bEbC D d2ILMb62C 9pG7bd2C

Unitario (S)= {S,A, B, C, D} Unitario (A) = {A, B, C, D} Unitario (B) = {B, C, D} Unitario (C) = {C, D] Unitario (£>) = {D}

El algoritmo introduce primero las producciones

*S —> b|ab A -^b\ab B —> ab\b C -\*b\ab*

y entonces se eliminan las producciones S —>A, A — B —>C y C -» D. La gramática resultante es

*5 —> b\ab\Aa A b\ab B —> ab\b C —> b | ab D -\* b*

Esta gramática puede simplificarse más, por medio de las otras técnicas. En lo visto anteriormente, hemos realizado la eliminación de producciones en gramáticas independientes del contexto de una forma bastante incómoda. Como última etapa en la simplificación de gramáticas independientes del con- texto, presentaremos un modelo o forma normal para las producciones. Se dice que una gramática independiente del contexto está en forma normal de Chomsky si no contiene producciones e y si todas las producciones son de la forma A —» a, para a e Z, o de la forma A —» BC, donde B y C son no terminales. Es decir, en la forma normal de Choms2y el lado derecho de cada producción contiene un único símbolo terminal o una par de no terminales. Obsérvese que, para una gra- mática en forma normal de Choms2y, el árbol de derivación para cualquier deri- vación está bastante bien construido ya que, excepto en las hojas, ¡el árbol es bi- nario!

Si G es una gramática independiente del contexto y

8

(G), G puede ser transformada en una gramática en forma normal de Choms2y. Para ello, primero se eliminan todas las producciones

8

, los símbolos inútiles y las producciones unitarias de G. Obsérvese que, una vez que se ha realizado lo anterior, si A —> w es una producción de G, se puede asegurar que | w\ > 1. Es más, si | vr| - ], en- tonces w es un símbolo terminal de Z, puesto que no hay producciones unitarias.

www.FreeLibros.com

131

Por otro lado, si |h>| > 1, entonces w puede contener tanto terminales como no terminales. Ahora transformaremos G convirtiendo tales w en cadenas que con- tengan sólo no terminales.

Supongamos que tenemos una producción de la forma A —> w, donde w = X\ X2 ... Xn. Si X¡ es un símbolo terminal, llamado a, sustituiremos X¡ por un nuevo no terminal C

0

y añadiremos la producción C„ —» a. Una vez que se aplica a G está conversión, todas las producciones son de la forma A —> w, don- de vv es un símbolo terminal o una cadena formada sólo por no terminales.

La última etapa para la transformación de G en forma normal de Choms2y, consiste en eliminar las cadenas con más de dos no terminales que se encuentren en el lado derecho de una producción. Para ello, si A —> B\ B2 ... Bn es una pro- ducción con n >

2

, la reemplazaremos por n -

1

producciones

*A -> B\ D\ D\ —> B2 D2*

*D„~ 2 —> Bn*

- 1

*Bn*

En ellas, los D¡ serán nuevos no terminales. En la gramática transformada resultante, el lado derecho de cada producción está compuesto por un único ter- minal o por una cadena de dos no terminales. Por lo tanto, todo lenguaje inde- pendiente del contexto que no contenga £ puede ser generado mediante una gra- mática independiente del contexto en forma normal de Choms2y.

Por ejemplo, consideremos la GIC

*S->bA\ aB A —^ bAA j ciS \ a B^>aBB\bS\b*

Obsérvese que esta gramática no contiene producciones £, producciones unitarias ni símbolos inútiles. Después de la primera transformación, la gramáti- ca se convierte en

*S -> C bA\CaB A -» ChAA\CaS\a B -> CaBB\ChS\b Ca —^ o, Cb —> b*

www.FreeLibros.com

d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

132

E2pG b j2 bME 7bEbC D d2ILMb62C 9pG7bd2C

En esta versión, el lado derecho de todas las producciones está formado por un único símbolo terminal o por una cadena de dos o más no terminales. Des- pués de la última conversión, la forma normal de Choms2y de la gramática será

*S-> ChA\CaA A —> Cfr D l | Ca 5 1 Cl D\-\*AA*

*=*

—>

*Ce, q?S | C¡) S | b qf £ ==*

Si L es un lenguaje independiente del contexto que contiene

6

, se puede ob- tener un gramática independiente del conter.to en forma normal de Choms2y para L - {£} y después añadir a la misma, la producción S —> e. La gramática re- sultante estará en forma normal de Choms2y exceptuando la producción e.

2HJ-Á,Á,V> 3J ¡O CJÁÁ,ÍQ exn

3.5.1. Aplicar el Algoritmo 3.5.1 a las siguientes gramáticas:

*(a) S aAb\cEB\CE*

*A —> clBE\eeC B ^ ff \D C —) gFB | ae D -^h*

(b) S -> aB

*/I ->bcCCC\dA B —» e C —>fA D —» Dgh*

3.5.2. Aplicar el Algoritmo 3.5.1 a la siguiente gramática:

*S —^ ci | o A | B j C A ■—> ciB! £ ». B —> Aci C —» bCD D -» ccc*

www.FreeLibros.com

133 d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

3.5.3. Aplicar el Algoritmo 3.5.2 a la siguiente gramática independiente del contexto:

*5 —»aAb A —» ccC B —> dd\D C —^ cíe O —»/*

3.5.4. Aplicar el Algoritmo 3.5.2 a la siguiente gramática independiente del contexto:

*S a\aA\B A aB\e B —»Aa D —» ¿Wd*

3.5.5. Eliminar los símbolos inútiles de la siguiente gramática por medio de los Algo-

ritmos 3.5.1 y 3.5.2:

*S->A\AA\AAA A —> A5a| ACa | a B ABa\Ab\z C —» Cab | CC £) —» C£>| C<¿| CEa E->b*

3.5.6. Obtener la colección de no terminales anulables que pertenecen a la siguiente

gramática:

*S -» aA | bA | a A -» aA | bAb 1*

8

3.5.7. Obtener, para la siguiente gramática, el número de no terminales anulables:

S —> ABaC A AB B —> b ¡ £\* C-H>Z?|e

www.FreeLibros.com

*n ^ d*

134

E2pG b j2 bME 7bEbC D d2ILMb62C 9pG7bd2C

3.5.8. Eliminar las producciones E de la gramática:

*S -» aA [ bA | a A -» aA | bAb I £*

3.5.9. Eliminar de la siguiente gramática las producciones £:

*S —>AB A aA I abB \ aCa B^>bA\BB\z C —> £ D-^dB\BCB\*

3.5.10. Eliminar de la siguiente gramática las producciones e:

*S —^ Ü | ciA | B A —> ÜB I £ B —^ Aci*

3.5.11. Eliminar las producciones £ de la gramática siguiente:

S -> C /i —^ AZ? fí -» ¿le C-> Z)le D->d

3.5.12. Simplificar la siguiente gramática tanto como sea posible

*S —> atí\aaB A -» £ B —> bA B —> e*

3.5.13. El lenguaje asociado con la siguiente gramática independiente del contexto con-

tiene e. Eliminar las producciones e excepto S £.

S —^ AB \ aB I e A —> BBB | aB | a I e B —^ a | úA t £

% 3.5.14. Realizar una algoritmo para construir Unitario (A) siendo A un no terminal de

una GIC.

www.FreeLibros.com

135

3.5.15. Eliminar todas las producciones unitarias de la siguiente gramática inde-

pendiente del contexto:

*5 CBa\D /I bbC B Se |ddd C eA \f\C D E\SABC E —»gh*

3.5.16. Eliminar todas las producciones unitarias de la siguiente gramática inde-

pendiente del contexto:

*S —> Aa|Ba | B A ->Afile B-\*aA\BB\z*

[Obsérvese que e e L (G)].

3.5.17. Convertir las siguientes gramáticas a forma normal de Choms2y:

*(a) S->AB\CA*

*A -> a B —> BC\AB C —^ ciB | b*

*(b) S->aAb\cHB\CH*

*A —» dBH\eeC B -\*ff\D C —» gFB | ah D —> i E —\*jF F —> dcGGG \ cF G ^ k F H —> Hlm*

3.5.18. Probar que, al realizar la conversión a forma normal de Choms2y, se puede ele- var al cuadrado el número de las producciones de una gramática independiente del contexto.

www.FreeLibros.com

d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

136

E2pG b j2 bME 7bEbC D d2ILMb62C 9pG7bd2C

íGpí52jbj2C j2 dpC d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

Las gramáticas en forma normal de Choms2y nos permiten obtener la relación que existe entre la longitud de una cadena y el número de pasos de su deriva- ción. Si se deriva £, se obtiene a partir de una única producción S —> £. Se puede probar, mediante inducción, que si se puede derivar w y | w\ >

0

, entonces la de- rivación tiene exactamente 2\w\ etapas. Por etapa, entendemos una sustitución. Escribir el símbolo inicial es una etapa, así como lo es sustituir a él o a cualquier otro terminal por el lado derecho de una producción.

Supongamos que G es una gramática independiente del contexto en forma normal de Choms2y y consideremos el árbol de derivación para una cadena cualquiera de L (G). Si un nodo tiene dos hijos, entonces los nodos hijos serán etiquetados con un no terminal y podremos tener al menos dos hijos más del mismo. (Esto es debido a que G está en forma normal de Choms2y). Es decir, el nodo raíz puede tener dos hijos, cada uno de sus hijos puede tener dos hijos y así sucesivamente. En cada nivel se pueden duplicar el número de nodos con res- pecto al anterior. Por tanto, en el nivel k podremos tener 2k nodos. (El nivel 0 es el nivel del nodo raíz).

Por otro lado, si un nodo tiene un único hijo, entonces dicho nodo hijo será etiquetado mediante un terminal, debido nuevamente, a que la gramática está en forma normal de Choms2y. Por tanto, todo símbolo terminal que etiquete a un nodo hoja corresponde a dos nodos, la hoja y su padre.

Supongamos que el camino más largo desde la raíz a las hojas consta de m + 2 nodos (y m +

1

arcos). Entonces, el árbol de derivación tiene m +

2

niveles (es decir, los niveles

0

,

1

,..., m +

1

) y los nodos del último nivel (las hojas) son hijos únicos de sus nodos padre. En el nivel m hay, como máximo, 2'" nodos pa- dre. Por tanto, si el camino más largo de un árbol de derivación consta de m + 2 nodos, entonces 2"' es la longitud máxima de la cadena derivada. Otra forma de decirlo es que, si la cadena derivada tiene longitud mayor que

2

'", entonces el ca- mino más largo debe contener más de m +

2

nodos.

Ahora, cada nodo del árbol de derivación de una cadena se corresponde con la aplicación de una producción. Por lo tanto, la relación que existe entre el nú- mero de producciones que comprende cada etapa y la longitud de la cadena de terminales resultante sugiere que una gramática independiente del contexto en forma normal de Choms2y, genera un promedio de un terminal por cada dos producciones.

El conocimiento de la relación existente entre la longitud de la cadena y su derivación en una gramática independiente del contexto, nos permite demostrar el siguiente lema de bombeo para los lenguajes independientes del contexto.

www.FreeLibros.com

137

Lema 3.6.1. Sea L un lenguaje independiente del contexto que no contiene e. Entonces existe un entero k para el cual, si z e L y | z | >k, entonces z se puede volver a escribir como z = uvwxy con las propiedades siguientes:

1

*. | vwx | <k.*

2. Al menos o v o x no es

8

.

3. uv'wx'y e L para todo i > 0.

Demostración. Supongamos que G = {N,Y,,S,P) es una gramática independiente del contexto en forma normal de Choms2y con L = L(G). Sea n el número de no terminales de N y sea k = 2n. Supongamos que z e L con | z | > k.

Vamos a considerar el camino más largo del árbol de derivación correspon- diente a z■ Puesto que |z| > 2”, este camino debe contener más de n + 2 nodos. De los n + 2 últimos nodos del camino, el último corresponderá a un terminal. Por tanto n + 1 de dichos nodos están etiquetados con no terminales. Puesto que sólo hay n no terminales en N, alguno se repetirá. Supongamos que A está repeti- do en el camino. Entonces se tiene que

*S => uvAxy => uvwxy- z*

para algunas subcadenas u, v ,iv ,x e y. Puesto que sólo hemos considerado los n + 2 últimos nodos de este camino, el camino de A a vwx puede tener como má- ximo n + 2 nodos. Por tanto

| vwx | < 2"

según las observaciones previas a este teorema.

Además, puesto que A ==> vAx, debemos obtener que A => V'AJÍ para algún i > 0. Por tanto, S => uv 'wx'y para todo i > 0.

Finalmente, obsérvese que, puesto que A =>v/bc y puesto que G está en for- ma normal de Choms2y, se debe tener que

A =>

6

*C=> vAx=$ vwx*

para algunos no terminales B y C. Entonces B =4 v y C=^> Ax o si no B vA y C => x. Ya que v y x son cadenas de terminales y no hay producciones

8

, en el primer caso se tiene que | v| >

1

con lo que v ^

8

y, en el segundo c§so, |x| >

1 con lo que x ^

8

. En cualquier caso al menos uno de ellos no es

8

. □

Al igual que el lema de bombeo para lenguajes regulares, el lema de bom- beo para lenguajes independientes del contexto nos proporciona la posibilidad de probar si ciertos lenguajes no son independientes del contexto. Para ello, usa-

www.FreeLibros.com

d2ILMb62Cx 5Ij2í2Ij52IE2C j2d upIE2SEp

138

E2pG b j2 bME 7bEbC D d2ILMb62C 9pG7bd2C

remos esencialmente el mismo planteamiento que en el caso de los lenguajes re- gulares.

Por ejemplo, el lenguaje L= {albJ\j = i2} no es independiente del contexto. Supongamos que L es independiente del contexto. Probaremos que es imposible. Si L es independiente del contexto, entonces se puede aplicar el lema de bombeo y por tanto habrá un k que satisfaga las condiciones del Lema 3.6.1. Conside- remos C = akbk . Desde luego, C e L y TCT > k. Por tanto, C se puede descompo- ner en

*z = uvwxy*

y se puede asegurar que uv'wx'y e L para todo /> 0 , tal que | vx | > 1 y | vwx | < k. Obsérvese que, si v = arbs para algún ry s, entonces V = (arbs)‘ y, por tanto, uvlw¿y tiene bes antes de aes con lo que no puede pertenecer a L. De for- ma similar, si x = arbs, tampoco pueden ser bombeados. Por lo que debemos ob- tener que

*v - a r y x ~ a s o*

*v = br y x - b 5 o también*

*v = a' y x - b s*

para algún valor de r y s. En el primer caso se tiene

*uv2wx2y = ak + r+sbk~*

En el segundo caso tenemos

*u v 2wx2y = akbk~+ r+s*

En ambos casos, cuando al menos uno de los dos r o s son mayores o igua- les a

1

(como se requiere en el lema de bombeo), la cadena resultante no puede pertenecer a L. En el tercer caso, se obtendrá

*uviwxiy = cik+{i- x)rbkl + {i- {)s*

el cual no pertenece a L para toda i a excepción de un número finito. Por tanto, L no puede ser independiente del contexto puesto que para él no se cumple el lema de bombeo.

El lema de bombeo tiene otros usos además de demostrar que un lenguaje no es independiente del contexto. Primero vamos a considerar el siguiente pro- blema: para una GIC G arbitraria, ¿L (G) es finito?

www.FreeLibros.com

139

Supongamos que G = (N, X, S, P) es una GIC arbitraria que está en forma normal de Choms2y. Se supone que N tiene n elementos. Entonces, de la demos- tración del Lema 3.6.1, se sabe que, si una cadena de L (G) tiene una longitud mayor que 2" ~ ', entonces L (G) es infinito. A la inversa, puesto que la colección de terminales de G es finita, si L (G) es infinita, entonces alguna cadena de L (G) contiene más de 2

n ~ 1

símbolos. Supongamos ahora que z e L(G) es la cadena más corta de L (G) con la propiedad de que | z | > 2" ~ 1. Pediremos que

2

”-

1

< \ z | <

2

" - ‘ +

2

"

Para verlo, supongamos que |z| > 2"-

1

+ 2". Entonces, de la demostración del Lema 3.6.1, se deduce que

2

= uvwxy y uwye L (G), para las subcadenas u, v, w, x, e y apropiadas. Entonces, puesto que | uwx\ < 2", se obtiene que | uwy\ > TCT -2 '1>2"-1.

Por otro lado, | uwy\ < \ uvwxy \ = |z |. Pero se había supuesto que z era la cadena más corta de longitud mayor que

2

'' ~

1

y, por tanto, se llega a una contra- dicción. Por consiguiente,

2« -i < j-j <2»-> +2'¡

En consecuencia, L (G) es infinito si y sólo si existe un z e L (G) para el cual

*2"~' < |s| <2"~ ] +2"*

Hay un número finito de cadenas sobre y, si es necesario, podremos comprobar si pertenecen a L (G). De aquí que tengamos un algoritmo para com- probar si un lenguaje independiente del contexto es finito o no.

Obsérvese que, en la eliminación de estados inútiles, proporcionamos de forma inadvertida un algoritmo que responde a otras cuestiones sobre un lengua- je especificado por una gramática independiente del contexto. L (G) es no vacío si y sólo si su símbolo inicial genera una cadena de terminales. Tenemos un al- goritmo para eliminar los no terminales que no generan cadenas de terminales. Por tanto, L (G) - 0 si el símbolo inicial no pertenece a la colección de no termi- nales de la gramática transformada.

Otra cuestión a tratar sobre los lenguajes independientes del contexto, hace referencia a sus miembros: Dado un lenguaje independiente del contexto L sobre el alfabeto Xy una cadena w e X\*, ¿vv e L o no?

Lema 3.6.2. Sea G - (N. X, 5, P) una gramática independiente del contexto que no tie- ne producciones £ y que está en forma normal dé Choms2y. Sea x una cadena de X\*. Se puede determinar, para cada A e N y para cada subcadena w de x, si A => vv.

www.FreeLibros.com

d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

140

E2pG b j2 bME 7bEbC D d2ILMb62C 9pG7bd2C

Demostración. Sea n= \ x | . Puesto que hay muchas subcadenas de x, las nombraremos mediante su posición inicial y su longitud. Sea w¡j la subcadena que comienza en la posición i y tiene una longitud j. Probaremos que el lema se cumple para todo w¡j. Lo realizaremos por inducción sobre la longitud de la subcadena, es decir, sobre j.

Supongamos que / = 1. Entonces | w¡j\ = 1 y, por tanto, wy es un símbolo terminal. Como la gramática está en forma normal de Choms2y, para algún no terminal A se tiene que A w¡j si y sólo si existe una producción A —> w¡j en P. Es posible determinarlo, ya que P es finito.

Ahora supongamos que j > 1 y que la afirmación se cumple para toda subca- dena de longitud menor que j. Obsérvese que A w¡j si y sólo si A —> BC para algún par B y C de no terminales para los cuales B => w¡k y C=> w¡ + k,j-k para algún k entre 1 y j - que j y, por la hipótesis 1. Entonces de inducción, tanto w¡k es como posible w¡ + k determinar \* j- k tienen si longitud B ==> £

menor w¡k y si C =5> Wj + kj-k- Además podemos determinar si A =4 w¡j para cada i entre 1 y n y cada j entre

1

*y n - i +*

1

. □

En la demostración anterior se observa que si j = n entonces se puede deter- minar si

*S ==> w\j — w\n — x*

Es decir, se puede determinar si x

6

L (G) para cada x e I “. Se cumple que .v s L (G) si y sólo si S => x y, por el Lema 3.6.2, sabemos que es posible determinar si S => .v. Naturalmente, no es lo mismo saber que algo es posible, que hacerlo. Se conocen varios algoritmos para determinar si x e L (G). Vamos a presentar uno llamado algoritmo de CYK y que se debe a Coc2e, Younger y Kasami. El algoritmo de CYK simplemente construye con- juntos N¡j de no terminales que generan las subcadenas w¡j de x. Una vez hecho, si S e N\„, entoncesx e L (G) (donde |jc[ = n). Obsérvese que se elimina mucho trabajo por el hecho de que no pueden existir subcadenas de longitud mayor que n - i +

1

que empiecen en la posición i. El algoritmo de CYK, se enuncia como sigue:

1

. Para cada i =

1

, 2,..., n, sea

*Mi = {A \A —> w¡\} »*

♦ Es decir, N¡\ es el conjunto de todos los no terminales que producen el /-ésimo símbolo de x

www.FreeLibros.com

141

2. Para j = 2, 3,..., n, hacer lo siguiente:

Para i = 1, 2 , n - j + 1, hacer lo siguiente:

a. Inicializar

***N¡j***

d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

= 0.

b. Para k= 1, 2, 1, añadir a N¡j todos los no terminales A para

*los cuales A —» BC, con B e Nik y C e N¡+k,j-k-*

3. Si S e N\n, entoncesx e L (G).

Obsérvese en que el algoritmo de CYK requiere que se tenga una gramática independiente del contexto en forma normal de Choms2y. El algoritmo para la construcción de los conjuntos de no terminales

***N¡j***

sigue la idea de la demostra- ción del Lema 3.6.2. En la etapa 2b, que es en la que se construyen los N¡¡, se emparejan los no terminales de N¡k con los de N¡+k,j-k, y se trata de encontrar- los en el lado derecho de las producciones. Cuando se obtienen tales lados dere- chos, se añade a N¡j el no terminal que está en el lado izquierdo de la producción dada.

Ejemplo 3.6.1

Consideremos la gramática independiente del contexto

*A —^ 3 A j ci B CC| b C ^A B \a*

Para la cadena x= bbab, se tiene la siguiente tabla donde cada casilla rep- resenta al conjunto N¡j:

*j~*

1

7 = 3 /= 4

*b j=*

1

7 =

2 B

0

*A s ,c*

*b j -2 B A, S S, C*

*a*

7

*= 3 A, C s ,c •V*

*B*

Ya que S está en Ni4, x se genera a partir del símbolo inicial 5. Por tanto, x está en el lenguaje generado por esta gramática.

www.FreeLibros.com

-

142

E2pG b j2 bME 7bEbC D d2ILMb62C 9pG7bd2C

Supongamos que Gy = (Ni, Xj, Si, P\) y G2 = (iV

2

, P2) son dos gra- máticas independientes del contexto para las cuales N\ y N2 son disjuntos. Defi- niremos la gramática independiente del contexto G = (N, X, S, P), donde

*N = N I U N2 U {S} Z = Z i u l*

2 /> = />, u/>

2

u{(S,S ,), (S, S2) }

siendo S un nuevo símbolo. Es decir, P contiene las producciones de P\ y P2, además de dos nuevas producciones S —> Si | S2. Se ve fácilmente que L (G) '= L (Gi) u L (G2). Primero, si w e L (Gi), entonces se tiene que Si =4 w, con lo que S => Si => w y, por tanto, se tiene que w e L (G). De forma similar, si w g L (G2), entonces se tiene que w e L(G) y, por tanto, L (Gi) u L (G2) c L (G). Por otro lado, si w e L (G), entonces S => w. Pero S —> S

1

1S

2

son las úni- cas producciones que tienen en su lado izquierdo el símbolo S. Entonces o bien

S => Si =$ vv

o

*S =$ S*

2

==> w

Ahora bien, como Ni n N 2 = 0, sólo se pueden usar producciones de P\ en Si => w, con lo que si S => Si w, se deduce que w e L(G\). De un modo similar, si S => S

2

w, se obtiene que w e L (G2). Por tanto, L (G) c L (Gi) u L (G

2

). Con lo que hemos demostrado el siguiente teorema:

Teorema 3.6.3. Si L\ y L2 son lenguajes independientes del contexto, entonces L\ u

¿2 es un lenguaje independiente del contexto.

Es decir, el conjunto de los lenguajes independientes del contexto es cerra- do con respecto a la unión. Este conjunto también es cerrado con respecto a la cerradura de estrella.

Teorema 3.6.4. Si L es un lenguaje independiente del contexto, entonces Ü también

lo es.

Demostración. Sea L - L (G) para G = (N, X, S, P). Constituiremos una gramática inde- pendiente del contexto G' = (N\ I, S', P') que genere L\*, de forma que N' - N u ¡S', T), donde S' y T son nuevos símbolos, y añadiremos las produc- ciones S' —» S T I e y T —> S T I e. Obsérvese que si w¡, W2, ..., wn e L (G), se tiene

www.FreeLibros.com

,

1 2

, S

2

143

S'=\*ST=\*SST

=4S...ST

(

H

términos)

*=> S ... SE*

==> Wl ( f . . . ü X

Por tanto, L \*cL (G'). □

Teorema 3.6.5. La concatenación de lenguajes independientes del contexto es inde-

pendiente del contexto.

Demostración. Véase el Ejercicio 3.6.5. □

En los Teoremas 3.6.3, 3.6.4 y 3.6.5 hemos enunciado tres propiedades de cierre de los lenguajes independientes del contexto. Desgraciadamente, los len- guajes independientes del contexto no son cerrados con respecto a la intersec- ción. El lenguaje

L - {í/7?'c'|í >0}

no es independiente del contexto, como se demostró en la aplicación del lema de bombeo (Lema 3.6.1). Obsérvese que L\ = {a‘b^c^\i,j>0} es generado por la gramática independiente del contexto

A -» a AI £ C —^ bCc I £

*y el lenguaje Lo = {a‘b'cJ \ i, j >*

0

} es generado por

*S ^ A C A —> aAb\e C —^ cCl £*

y, por tanto, son lenguajes independientes del contexto. Pero se tiene que L - L \ n L

2

, y con lo que, la intersección de lenguajes independientes del con- texto no es necesariamente un lenguaje independiente del contexto.”

Además, los lenguajes independientes del contexto no son cerrados con res- pecto a la complementación. Si denotamos Z\* - L¡ mediante L. entonces

L\C\L2- (¿i u Li)

www.FreeLibros.com

d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

144

E2pG b j2 bME 7bEbC D d2ILMb62C 9pG7bd2C

Ya que la unión de lenguajes independientes del contexto produce un len- guaje independiente del contexto, obsérvese que, si al complementar lenguajes independientes del contexto siempre se obtuvieran lenguajes independientes del contexto, la intersección anterior debería haber sido independiente del contexto.

2HJ-Á,Á,V> 3J ¡O CJÁÁ,ÍQ exa

3.6.1. Probar que cada uno de los siguientes lenguajes no son independientes del con-

texto.

*(a) {a‘b'c'\ i > 1}*

*r9u {a‘b'cj \j>i}*

*(c) {a‘b^ck\ i<j<k]*

(d) {a'| / es primo}

(e) {w e [a, ¿>, c}\*| w tiene el mismo número de aes que de bes y ces} (f)

*{a"b"c'"\n < m < 2n)*

(g) {ww| w e {a, b}\*}.

3.6.2. Determinar si el lenguaje generado por

S —> ciciA ¡ B B a A \ b A —> ríS 16 1 £ es finito o infinito.

3.6.3. Determinar si bba, bab y babba están en L (G) correspondiente a la gramática G

del Ejemplo 3.6.1, usando el algoritmo de CYK.

3.6.4. Sea G = (N, X, S, P) una G1C. Al igual que en la demostración del Teorema

3.6.4, construir G’ - (Ar. X. S, P'). Demostrar que L (G’) c V\

3.6.5. Demostrar el Teorema 3.6.5.

bME 7bEb j2 í5db

Hemos visto que las gramáticas independientes del contexto amplían la capaci- dad para especificar lenguajes al incluir algunos lenguajes que no son reconoci- dos por un autómata finito. En esta sección consideraremos un autómata que será capaz de reconocer todo lenguaje independiente del contexto.

Intuitivamente, el problema que se plantea con los autómatas finitos es que sólo tienen capacidad para una “memoria” finita. Lenguajes independientes del contexto tan sencillos como {anb"\n > 0} necesitan guardar gran cantidad de in-

www.FreeLibros.com

145

formación; se debe verificar no solamente que todas las aes preceden a las bes, sino que además se tienen que contar las aes. Puesto que el número de aes es arbi- trario, necesitamos establecer una limitación sobre el número de aes que se pueden contar.

El lenguaje independiente del contexto {wcw'^we {a,b,c} } necesita algo mas que una capacidad sin límites para contar los símbolos. Además se deben guar- dar los símbolos de la cadena w para compararlos con los símbolos de la cadena w'.

En esta sección definiremos un autómata que cuenta con un mecanismo que permite almacenamiento ilimitado y opera como una pila. Este autómata se lla- ma autómata de pTícL)En la siguiente sección, se estudiará la conexión que existe entre los autómatas ée pila y los lenguajes independientes del contexto.

<Qin autómata de pila se comporta de forma similar a como lo hacen los autó- matas finitós/del Capítulo 2. En todo momento se encuentran en un estado y el cambio de estado depende del estado actual y de una información adicional. En el caso de los autómatas finitos del Capítulo 2, la “información adicional” con- siste en el símbolo de entrada actual. En el caso de los autómatas de pila, se in- cluyen el símbolo de entrada actual y el símbolo que está en ese momento en la cima de la pila. Además de cambiar de estado, el autómata de pila cambia, tam- bién, la cima de la pila.

Definición 3.7.1. Un autómata de pila no determinista (ADPND) es una 7-tupla,

M - (£>, Z, T, A, s, F, z) donde

Q es un conjunto finito de estados

*Z es un alfabeto de entrada*

r es un alfabeto llamado alfabeto de la pila

*A es una regla de transición*

*se. Q es el estado inicial o de partida*

*z e r es el símbolo inicial de la pila*

*F c Q es el conjunto de estados finales o de aceptación.*

De nuestra descripción anterior se deduce que una regla de transición, para obtener el siguiente estado y la acción a realizar sobre la pila, debe conocer el estado actual, el símbolo de entrada actual y la cima actual de la pila. Por consi- guiente, A se define por medio de ternas de la forma (q, a, y), donde q es un es- tado de Q, a es un símbolo de E u {£} y y e T. El resultado de aplicar A a dicha terna es una colección de pares (p, w), donde "p e Q es el estado siguiente y >v e T \* es la cadena que se introducirá (o apilará) en la pila en lugar del símbolo Y que estaba antes allí. Por tanto, se puede definir la relación A como

www.FreeLibros.com

d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

146

E2pG b j2 bME 7bEbC D d2ILMb62C 9pG7bd2C

A C

2 X

(E U {£})

X

T

X <2 X

r\*

Ya que estamos definiendo un autómata de pila no determinista, cabe espe- rar que A no tiene por qué ser una función. Por tanto, si aplicamos A a la terna (< 7

i, a, b) se obtiene el conjunto {(<

72

*, cd), {qi, dcé), (*

93

, efe)}, del cual se elige de forma no determinista uno de los pares, de manera que el autómata de pila cam- bia para reflejar dicha elección. Obsérvese que, puesto que los pares resultantes pertenecen a g x T yya que £ e P , se puede tener que de A (q, a, b) se obtenga (p, £). Esto indica que el estado siguiente es p y que el símbolo b se elimina (o se desapila) de la cima de la pila.

Al definir A c Q x (E u {£}) x T x Q x T\*, se fuerza a que A deba conside- rar un símbolo de la pila en todos sus movimientos. Es decir, no es posible nin- gún movimiento si la pila está vacía. Por esta razón, se supone que inicialmente la pila contiene algún símbolo z, que es el símbolo inicial de la pila. Por otro lado, un movimiento tal como

*A (q, £, a) = {(pf aa)}*

indica que el ADPND puede cambiar a un estado p y apilar una a en la pila sin consumir ningún símbolo de la entrada. Finalmente, obsérvese que si A (q,

0

, y) = 0, no es posible ningún movimiento desde el estado q con el símbo- lo de entrada

<3

y con el símbolo de la pila y. En este caso, el ADPND parará su ejecución.

Consideremos el autómata de pila no determinista definido por

*á—T■—S■ 23, <?4} I ={a,b} r= {A ,B }*

*Z = A z á—■0*

*s = q\*

y A dado por la siguiente tabla:

*A {a, A) ib, A) (£ ,A) (a,B) 0b,B) (£, B)*

*—T {(q2, BA), (q4,A)} <?2*

{(?4, £)}

\* i

f e BB)} {(<?3,£)} \*

<73 {(<74, ¿)} {(<73, £)}

www.FreeLibros.com

147

Ya que A depende del estado actual, el símbolo de entrada actual y el sím- bolo actual de la cima de la pila, la tabla tiene filas que corresponden a los es- tados y columnas que corresponden a los pares de símbolos de entrada y de la pila.

Obsérvese que no hay transiciones para todas las temas posibles de estado, símbolo de entrada y símbolo de la pila. Por tanto, si el ADPND pasa a un esta- do para el cual no se especifica un estado siguiente y una acción de la pila para los símbolos actuales de la pila y la entrada, el ADPND no puede volver a reali- zar ningún movimiento. En otras palabras, su actividad termina. En particular, cuando el autómata está en el estado <

74

, que es el estado de aceptación, no hay ninguna transición sea cual sea el símbolo de la cima y de la entrada.

Las acciones que realiza un ADPND son sencillas. La transición A (<

72

, a, B) — {(qi, BB)} apila B sobre la pila y no cambia el estado. La transi- ción A {q2, b, B) = {(<

73

, s)} desapila la B y cambia de estado. La transición A [qi, a, A) = {{qi, BA), (<

74

, £)} indica que se realizará la elección de una acción de forma no determinista. Como, para esta transición, el ADPND empieza en el estado q\ con A en la cima de la pila, esta transición indica que o bien se apila B y se pasa al estado

<72

o si no se desapila A y se pasa al estado de aceptación <

74

. Si se realiza esto último, el ADPND parará.

“Por otro lado, si el ADPND se mueve al estado c/2, entonces obsérvese que cada vez que a aparece en la entrada se apila una B en la pila. Es más, puesto que el ADPND permanece en el estado qi hasta que se encuentra la primera b y entonces se mueve al estado <

73

, ninguna b puede preceder a una a. Finalmente, en el estado

¿73

sólo se consideran las bes y. cuando se encuentra cualquier b. se desapila B de la pila. (Sólo pueden desapilarse las Bes que fueron apiladas, debi- do a encontrarse una a en la entrada).

El lector podría verificar que las únicas cadenas que pertenecen al lenguaje {anbn | n >

0

} u {«}, son las únicas cadenas de entrada que, una vez que han sido consumidas, causan que el ADPND termine en el estado final q

4

. El autómata de pila no determinista del ejemplo anterior acepta el lenguaje {a"bn|n >

0

} u {< 7

} puesto que las cadenas que una vez consumidas causan que el ADPND se mueva de su configuración inicial a su estado final son exacta- mente las cadenas que pertenecen al lenguaje anterior. Procederemos a formali- zar qué significa que un ADPND acepte un lenguaje pero primero introducire- mos una notación usual.

Durante el procesamiento de una cadena, se pueden describir las sucesivas configuraciones por las que pasa el ADPND en función de su estado actual, de los contenidos de la pila y de la entrada no leída. La terna (q, w, u), donde q es el estado actual, w es ia cadena de entrada restanta y u el contenido de la pila (con el símbolo de la cima en el extremo de la izquierda), se llama descripción ins- tantánea (DI) del autómata. Describe la configuración del autómata de pila en

www.FreeLibros.com

d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

148

E2pG b j2 bME 7bEbC D d2ILMb62C 9pG7bd2C

un instante en particular. Indicaremos un movimiento de una configuración a otra situando el símbolo i- entre dos descripciones instantáneas

(Y/i, aw, bx) t- ( 7

/

2

*, w, yx)*

representa un movimiento que resulta de (<

72

, y) e A (q\, a, b). Podemos denotar los movimientos con un número arbitrario de pasos por medio de t- y í (donde \* indica “cero o más pasos” y + indica “uno o más pasos”).

Ahora definiremos formalmente lo que significa que un autómata de pila no determinista acepte un lenguaje.

Definición 3.7.2. Sea M = (Q, Z, T, s, z, F, A) un autómata de pila no determinista. El

lenguaje aceptado por M se denota por L (M) y es el conjuñ'o

L (M) = {w € E\* I (.y, w, z) t- (p, e, u) para p e F y « e F )

Obsérvese que la aceptación requiere que M se mueva a un estado final cuando la cadena w se agote. M puede terminar o no con la pila vacía. (Sin em- bargo, obsérvese que cuando la pila se vacía el ADPND debe parar, ya que todas las transiciones requieren un símbolo de la pila).

Ejemplo 3.7.1

Supongamos que queremos construir un autómata de pila no determinista que acepte el lenguaje

L ~ {w e {«, ¿>}'| vv contiene la misma cantidad de oes que de bes}

Debemos contar el número de ocurrencias de aes y bes. Esto puede reali- zarse introduciendo símbolos en la pila cuando se lee algún carácter de entrada y extrayéndolo cuando se lee otro. Sea M el ADPND dado por

2

= 2/!-<?

2

}

2

*= [a,b] r = {A, B, Z} s = q\ z = Z F — {<? 2*

}

y la regla de transición A dada por la lista

www.FreeLibros.com

149

*A(qu zrZ) = {(q2,Z)} A (qu a,Z)={(quAZ)} i A (qu b,Z) = {(qi,BZ)};. A (qu a, A) = {{q\,AA)} A (q\, b, A) = {q\, e)}*

• A (?i, a, B) ={(?,,£)} ■ A (qu b,B)={(qh BB)}

Para procesar la cadena abba, M realiza los siguientes movimientos:

*(qi, abba, Z) l- (q\t bba, AZ) s*

*\-(qi,ba,Z) h- (q\, a, BZ) h (qi, e, Z) I- (q2, £, Z)*

y el ADPND para en el estado de aceptación q%. Luego el abba está en L (M).

Ejemplo 3.7.2

Podemos hacer uso del hecho de que los símbolos pueden ser recuperados de la pila en orden inverso a como fueron introducidos. Consideremos el len- guaje L = {wcw11 w e {a, b]\*}. Un ADPND que acepta L debería introducir los caracteres de entrada en la pila hasta que se encuentra una c y, entonces, compa- ra los caracteres de la entrada con la cima de la pila, desapilando la pila si con- cuerdan. Un ADPND M para el mismo puede ser dado mediante

*Q-{c¡i, q*

2

, <?3} £={«,£>} T ={a, b,z} s = q\ z = símbolo inicial de la pila F={<?3}

y A viene dado por la lista siguiente:

www.FreeLibros.com

d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

150

E2pG b j2 bME 7bEbC D d2ILMb62C 9pG7bd2C

*A{qu a, z)={(q\,az)} A (< 7*

*A(qu c, z)= {(q2, z)} A(qu c, a)={(q2, fl)} A(qu c,b)= {(q2, b)} A (q2, a, a)= \ {q2, e)} A{q2, b,b)={{q2, e)} A (q2, £.,Z)= {(qi, z)}*

Cada autómata finito no determinista se puede interpretar como un ADPND que nunca opera sobre la pila. Por ejemplo, consideremos el AFN cuyo diagrama de transición es

*P2*

p

Sea el ADPND M dado por

y A dado por

*i, a, á) = {{qu aá)} A(q¡,a, b) = {(qh ab)} A(qi,b, z)= {(qu bz)} A (q\,b, a)= {(qu ba)} A (qu b,b)= {(qu bb)}*

< 2

*= {qi, qz)*

T = {z},el símbolo inicial de la pila s = q i

A («. z)

q\ {(<?2,¿)}

qi U<72,z)}

Está claro que M acepta el mismo lenguaje que el autómata finito.

2HJ-Á,Á,V> 3J ¡O CJÁÁ,ÍQ exs

3.7.1. Obtener un ADPND que acepte {a"b" \ n > 0}.

3.7.2. Describir el proceso realizado por el ADPND M de] Ejemplo 3.7.1 sobre las ca-

denas abaababb y abaa. ¿Son aceptadas por M?

3.7.3. Describir el proceso que realiza el ADPND M del Ejemplo 3.7.2 sobre Jas cade-

*nas c, abcba, abcab y babbcbbab.*

3.7.4. El lenguaje L = {ww1 | >v e {a, /;}+} es similar al lenguaje del Ejemplo 3.7.2, ex- cepto porque no hay ningún carácter que marque el límite entre w y w Por tan- to, para que un autómata de pila no determinista acepte L tendrá que existir una elección no determinista en un punto medio que pasa de apilar caracteres en la

www.FreeLibros.com

151

pila a compararlos y desapilarlos. Usar el ADPND del Ejemplo 3.7.2 como pun- to de partida y construir un ADPND que acepte L añadiendo o eliminando las transiciones apropiadas.

3.7.5. Obtener los ADPND para los siguientes lenguajes:

*(a) [anb2n\n>*

0

*} (b) [anbmcn + m\n=0 y m >0}*

*(c) {anb"'\n <m<3n}*

(d) {we {a,b}”\w contiene una a más que bes]

*(e) {a"bm\n=0 y m\*n) (f) {w[C\V2*

1

*W|, W2 s {a,b}\* y w¡ ^ wj}*

3.7.6. ¿Qué lenguaje será aceptado por el M dado a continuación

Q= {quq2, qi) Z = {a,b} F = E u {z}, donde z es el símbolo inicial de la pila s-q \ z á—l0

y A viene dado por la tabla

*A (a. z) (a. b) ib, a) (b. b)*

<7i

{(</;. «.). ((]?.£)}

<72

*m , o í*

{(<72./;)}

*{(cn,b))*

¿Cuál será el lenguaje aceptado si se cambia F por F = [q¡, (¡2, <

73

}?

3.7.7. Obtener un ADPND que acepte ciaba. La presencia de un dispositivo de me- moria en un autómata puede permitir que se economicen estados. Un autómata finito no determinista para el lenguaje precedente, necesariamente tiene al me- nos cuatro estados y, por tanto, un ADPND para este lenguaje que ignore su pila deberá tener, al menos, cuatro estados. Obtener un ADPND para adba que sólo tenga dos estados.

bME 7bEbC j2 í5db D d2ILMb62C 5Ij2í2Ij52IE2C o j2d upIE2SEp

El principal resultado de esta sección es que los autómatas de pila no determinis- tas aceptan los lenguajes independientes del contexto. Esto requiere que, para cada lenguaje independiente del contexto, debamos poder encontrar un autómata

www.FreeLibros.com

d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

152

E2pG b j2 bME 7bEbC D d2ILMb62C 9pG7bd2C

de pila no determinista que lo acepte y a la inversa, para cada lenguaje aceptado por un autómata de pila no determinista, obtener una gramática independiente del contexto. Primero demostraremos que cualquier lenguaje independiente del contexto es aceptado por un autómata de pila no determinista presentando un método de construcción para dicho ADPND.

Sea G = (N, X, S, P) una gramática independiente del contexto. Desearíamos construir un autómata de pila no determinista que acepte L (G). El ADPND que construiremos debería poder verificar que todas las cadenas de L (G) son deriva- bles. La idea principal es construir un ADPND que pueda realizar una deriva- ción por la izquierda para cualquier cadena del lenguaje. Representaremos la de- rivación guardando en la pila los no terminales del extremo derecho de la deri- vación, mientras que la parte izquierda (compuesta totalmente por terminales) es idéntica que la cadena de entrada que se ha leído.

Empezaremos por introducir en la pila el símbolo inicial (de G). En cada una de las etapas posteriores realizaremos una de estas acciones:

1. Si el símbolo que está en la cima de la pila es un no terminal A, lo susti-

tuiremos por el lado derecho de la producción para A, A —» vv, o

2. Si el símbolo de la cima de la pila es un terminal y se corresponde con

el siguiente símbolo de la entrada, los desapilaremos de la pila.

Estas acciones imitan una derivación por la izquierda para la cadena de en- trada. Si se agota la cadena de entrada y en la cima de la pila se encuentra el símbolo inicial de la pila, aceptaremos la cadena puesto que es posible la deriva- ción de la misma. Obsérvese que el no determinismo se presenta en la etapa 1 cuando se elige el lado derecho de alguna producción para que sea introducido en la pila.

Para definir dicho autómata de pila no determinista M, sean

Q={q\,q7,q2) F = A í u Z u { z ¡ , donde z es un símbolo inicial de la pila

(y es distinto de todos los símbolos de N u

2

*) F={qi)*

*s = q\*

y la regla de transición A está compuesta por cuatro tipos de transiciones:

♦ 1. A (qi, e, z) = {(<?

2

, Sz)}, la cual se corresponde con la introducción del símbolo inicial en la pila.

www.FreeLibros.com

153

2. A (<

72

, e, A) = {(<

72

, w)\A -» w es «na producción de P} para cada no terminal A de N.

*3. A (q2y a, a) = {(<*

72

, £)} para cada símbolo terminal a de X.

4. A (<

72

, £, z) = {(

9 3

, z)}.

Ejemplo 3.8.1

Sea G una gramática independiente del contexto cuyas producciones son

**S —» aSa|bSb|c.**

Obsérvese que ¿ (G) = {wcvv^l w G {a, b}\*}. El autómata de pila no deter- minista correspondiente tiene la siguiente regla de transición

**A (qu e, z) = {(<72, Sz)} A (<72, e, S) = {(<72, aSa), {q2, bSb), (<72, o)} A {q% a,a) = A (<72, b, b) ~ A (^2, c, c) = {(<72, e)}**

**A (<72, e, z) = {(<73, e)}**

**La cadena abcba es aceptada mediante la siguiente secuencia:**

*(<71, abcba, z) H (<72, abcba, Sz)*

**l- (<72, abcba, aSaz) h (c/2, beba, Saz) se extrae un símbolo de la pila i- (<72, beba, bSbaz) H (í/2, cba, Sbaz) otra extracción l- (<72, cba, cbaz) l- (qi, ba, baz) otra extracción l- (<72, «, az) otra extracción l- (¿72, £, z) otra extracción H (<73, £, z) se acepta**

Supongamos que se ha construido un ADPND a partir de una gramática in- dependiente del contexto, de la forma vista anteriormente. Obsérvese que si se tiene que

- (<

72

, x, Aa) t- (<

72

, \*, (Ja) ♦ entonces, en la gramática es posible que A => (3. Por tanto, si w = a\a

*2*

... a,, es aceptada por el ADPND, debemos tener que

www.FreeLibros.com

d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

154

E2pG b j2 bME 7bEbC D d2ILMb62C 9pG7bd2C

(42, )S$fjjj )■p vC) H (<72, ai«2 ■ • •

«íPiz)

I- Z—f■ <32 ... «n, Piz)

*h (<72, {np■ )pCR*

i~ (<72, £, z) H (93, £, z)

y por tanto se obtienen las derivaciones siguientes

n

cíiPi => dia P = > ...= > )T$

2

*... )X N ü*

Por consiguiente, si w es aceptada por un ADPND, entonces w se deriva de la gramática.

A la inversa, supongamos que tenemos una gramática en forma normal de Choms2y. Si v =>

«102

... )X■ entonces tendremos una derivación de )T$f ... )X■ por la izquierda y de la forma

*v =>/\iOCi => )TZ}T => ¡STFfZ}f + )T2*

20\*2

*%m%f ) T*

Por tanto, en un ADPND derivado de esta gramática, se puede tener la se- cuencia

(<72, )T)2 ... )p■ vCR h (<72, «i«2 . •. )p■ ¡ST }TCR

*1- Z¡af■ {mf ... )p■ a iCR*

h (<72, e, CR

1- (<73, e, CR

Es decir, el ADPND acepta la cadena vv - )T)C ... )pj Por tanto, se deduce el siguiente teorema:

Teorema 3.8.1. Si L es un lenguaje independiente del contexto, entonces existirá un

ADPND para el cual L - L (M).

El Teorema 3.8.1 representa la mitad de nuestro objetivo. Nuestra tarea pen- diente es demostrar su inverso, es decir, que cualquier lenguaje aceptado por un ADPND es independiente del contexto. Para generalizar, necesitaremos tomar un ADPND arbitrario y probar cómo se obtendrá una gramática independiente del contexto que genere el lenguaje aceptado por él. Con el objeto de simplificar el problema, trataremos con uno equivalente, desde el principio, en el cual el ADPND se comporte correctamente.

www.FreeLibros.com

155

Sea M un ADPND arbitrario. El lenguaje aceptado por la pila vacía de M se define como

*oZsR &üTZ—T■ü■CRkZD■?x■xR0*

En este caso, q\ es el estado inicial de M y p es cualquier estado de Q. N (M) es, sencillamente, el conjunto de todas las cadenas que conducen a M des- de el estado inicial, a cualquier estado en el cual la pila esté vacía. Puesto que to- das las transiciones se definen en función de un símbolo de la pila, tal configura- ción es una configuración de parada de M.

Obsérvese que para un ADPND, no se cumple necesariamente que N (M) = L (M). Por ejemplo sea M, cuyas transiciones son

*A(q\,a, z)={(q\,az)} FZ—T■\■CR áZ—S■xR0 A(q\,a,a)= {(<*

73

*, a)}*

*donde Q = {qy, qz, q^}, F= {<*

73

}, z es el símbolo inicial de la pila y q\ es el esta- do inicial. Obsérvese que N (M) = {b} mientras que L (M) - {a2}.

Sea M una ADPND. Podemos transformar M en un ADPND M' tal que L (M) = N (M'). Resultará que M' tendrá un único estado final.

Primero observemos que si M nunca desapila z de la pila, se realiza por me- dio de una transición de la forma (q, e) e A (q, t, z) para algunos estados q y q y algún x e F u {e}. Puede ser que q' e F o no. Se sustituyen todas las transi- ciones donde q no esté en F por nuevas transiciones ZDT■ z) para un nuevo estado p\. Se sustituyen todas las transiciones en las que q esté en F por ZDf■ z) para un nuevo estado p2 y se incluye pz en F. Después de realizar esto, N (M) = 0 pero L (M) no ha variado. Entonces se añaden estados y transiciones, de forma que, una vez que M entre en un estado final, vacíe su pila. Para todo q e F, se añaden las transiciones

A (q, £, y) = {(p

3

, y)}, para todo y e T A (p

3

. £.Y)=Í (P

3

. £)}, para todo y e V - {z} A(p

3

*, £, z) = {(P4> £)}*

Después de hecho esto, se obtiene que F= {^

4

}. El ADPND íesultante, M'. acepta cadenas en un único estado final con la pila vacía. Por tanto, L (M') = N (MO = L (M).

Además se requiere que todas las transiciones sean de la forma

*y A(q,a,A) = {c\,c2,...,cn}*

www.FreeLibros.com

d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

156

E2pG b j2 bME 7bEbC D d2ILMb62C 9pG7bd2C

donde cada c¡ = (p, e) o c¡ = (p, BC). Es decir, cada transición incrementa o de- crementa el contenido de la pila en un único símbolo.

Para construir un gramática independiente del contexto para un ADPND, in- vertiremos el proceso de la última sección para que una derivación simule los movimientos que un ADPND realiza para aceptar una cadena. Para ello, en cada etapa, la parte correspondiente a la cadena parcialmente generada formada por no terminales, reflejará el contenido de la pila, mientras que el prefijo que for- man los terminales es la parte de la cadena que ya ha sido procesada. Una confi- guración de un ADPND contiene un estado además del contenido de la pila. Ya que la derivación que simulará el movimiento de un ADPND depende del estado actual, queremos que nuestra gramática independiente del contexto lleve tam- bién el rastro de los estados por los que se pasa. Una solución de este problema nos lleva a usar no terminales de la forma [qAp], donde interpretaremos [qAp] => w como la acción del ADPND correspondiente, que saca A de la pila y se mueve del estado q al estado p mientras consume la cadena de entrada w.

Ya que se requieren ciertos criterios para formar las transiciones, se sim- plifica el proceso de enuriteración de las producciones correspondientes. Si (.q¡, e) e A (q¡, a, A), la producción correspondiente es [¿

7

,- Aqj) -> a, ya que el ADPND pasa del estado q¡ al estado q¡ y desapila A de la pila sobre el símbolo de entrada a.

Por otro lado, si (qj, BC) e A (q¡, a, A), obsérvese que la entrada a produce que A sea eliminado de la pila, pero entonces B y C serían rechazados. Ya que el ADPND acepta cuando está la pila vacía, B y C deben ser eliminados necesaria- mente. Su eliminación se obtiene por medio de los cambios de estado apropia- dos. Por tanto, si (qj, BC) e A (q¡,a, A), incluiremos todas las producciones de la forma [q¡Aqm] —> a [qj Bqn] [q„ Cqm\. donde q„ y qm pueden ser cualquiera de los estados de Q.

Finalmente, tendremos como símbolo inicial [szqj], donde s es el estado ini- cial, z es el símbolo inicial de la pila y q¡ es el (único) estado de aceptación del ADPND.

Ejemplo 3.8.2

Consideremos el ADPND cuyas transiciones son las siguientes

1

*. FZ—S■)■CR áZ—QsRR*

**4.**

*A (q2, \■FR {(<?2> £)} 2. A(q¡,a,A)= {(q,,AA)} 5. A (q2, e, A) = {(qi, £)1 3. A(qu b,A) = {(<?*

2

,£)}

6

. A (q2, £. z) = {(#¡? £)}

donde z es el símbolo inicial de la pila, F= {<

73

} y q\ es el estado inicial. Obsér- vese que este ADPND sólo pasa al estado de aceptación cuando la pila está va-

www.FreeLibros.com

157

cía. Además, todas las transiciones son de la forma requerida. En una derivación de la gramática asociada, tendremos como símbolo inicial [q\zq-i] y las transi- ciones 3,4, 5 y

6

se traducen fácilmente como las siguientes producciones:

*[q\Aq2] -> b lqiAq2) ->ble*

*[qizq-i] -» e*

Las transiciones 1 y 2 generan las siguientes producciones:

*De J: [q\zq<\-> a[q\Aq\] [q\Z.q\}\a [q{Aq2] [q2zq\\\a[q\Aqi\ [qizqú [q\zq2] -> a [q\Aq\] [q\zq2]\a[q\Aq2] \q2zq2)\a[qxAq3\ [q^zq2] [q\zq\*\ -> a [q\Aq\] [q\zqü |\*. [q\ Aqi\ [qizqú I a [q\Aq{] [qjzxfr]*

*De 2: [q\Aq\] -> a [q\Aqx] [q\Aq(\\a [q\Aq2] [q2Aqx\\ci [q\Aqj\ [qyAqx\ [q\Aq2] —» a [q\Aq\] [q\Aq2]\a [q\Aq2] [q2Aq2]\a [q\Aq3\ [,qyAq2] [q\Aqú -» a [q\Aq\] [q\Aqi\\a [q\Aq2] [q2Aq^}\a [q\Aqü [q^Aqú*

La cadena aabb es aceptada por este autómata con lá siguiente secuencia de configuraciones:

*(q\. aabb, z) l- (qi, abb, Az)*

1 - (<

72

*, e. z) h (q,, bb,AAz) l- (q2, b, Az)*

1 - (#

3

, e, e)

La derivación correspondiente en la gramática precedente es

*[q\zqi] => a [q\Aq2] [q2zqi\*

*=» aa[q\Aq2\ [q2Aq2] [q2zqi\ => aab [qiAqúiqizqz] => aabb [q2zqi\ => aabb*

(Obsérvese que el “primer q” en cada cadena perteneciente a la derivación, corresponde al estado actual en la secuencia del ADPND. Además, las secuen- cias de símbolos “centrales” se corresponde con el contenido déla pila cuando es aceptada aabb).

A menudo, esta construcción crea gramáticas complicadas. Por ejemplo, en el caso precedente, los símbolos no terminales [qizq\\, [q^zq

1

], [qizqi], [qiAq\\, [qyAqú, [q^Ac/i] y \q3zq3}. nunca aparecen en el lado izquierdo de una produc-

www.FreeLibros.com

d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

158

E2pG5b j2 bME 7bEbC D d2ILMb62C 9pG7bd2C

ción. Por tanto, dichos símbolos nunca podrán aparecer en una derivación de una cadena de terminales. A pesar de haber introducido una complejidad no desea- ble, esta técnica de construcción se puede usar para cualquier ADPND cuyas re- glas de transición satisfacen las condiciones. Se puede aplicar las técnicas de simplificación vistas en la Sección 3.5 a la gramática resultante, con el fin de eli- minar las partes inútiles. Obsérvese que, de la forma de construir una gramática a partir de un ADPND que satisfaga las condiciones dadas, se obtiene que

*(q¡, uv,Ax) h- (cjj, v, x)*

por medio de las operaciones que realiza un autómata de pila, entonces en la gra- mática resultante se tendrá

*[q¡ Acjj | u*

Es decir, si el ADPND elimina A de la pila cuando se lee la cadena u y pasa del estado q¡ al estado qj, entonces el no terminal [q¡ Aq¡ ] puede derivar u.

A la inversa, obsérvese que, si en alguna derivación se tiene

*f q¡ Aqk ] a [q¡ Bqr] [qr Cqk ]*

entonces A puede ser sustituido por BC en la pila cuando se lea a y se pasará del estado q¡ al estado qj.

Por otro lado, si [q¡ Aq¡ ] => a es una derivación con un único paso, entonces A (q¡, a, A) contiene (q¡, e), con lo que A podrá ser desapilado al consumirse la cadena de entrada a. Por tanto, si [q¡ Aq¡ ] => u. la secuencia de cadenas de termi- nales y no terminales que forman la derivación se corresponde con la secuencia de movimientos que realiza un ADPND para el cual (q¡, uv, Ax) h (c¡j, v, x). Por tanto, cualquier cadena de terminales generada por la gramática es aceptada por el ADPND, y viceversa. Enunciaremos el siguiente teorema:

Teorema 3.8.2. Si L es aceptado por un ADPND. entonces L es un lenguaje inde-

pendiente del contexto.

Los autómatas de pila no deterministas nos proporcionan otra alternativa para caracterizar los lenguajes independientes del contexto. Es tan fácil probar que un lenguaje es un lenguaje independiente del contexto, demostrando que es aceptado por un ADPND, como probar que es independiente del context® si es generado por una gramática independiente del contexto. A veces incluso es más fácil describir un ADPND que una gramática. Además, el modelo del ADPND

www.FreeLibros.com

159

nos facilita la tarea de probar ciertas propiedades de los lenguajes inde- pendientes del contexto.

Teorema 3.8.3. Si L\ es un lenguaje independiente del contexto y L-i es un lenguaje re-

gular, entonces L\ n L

2

es un lenguaje independiente del contexto.

Demostración. Sea M\ = (£>i, Zi, T, si, z, F\, Ai) un ADPND que acepta el lenguaje

L\. Sea M

2

*= (Q2, Z*

2

*, S2, F2,*

8

) un autómata finito determinista que acepta el len- guaje regular L2. Construiremos un ADPND que acepte L\ n L2 mediante la combinación de M\ y M2. Este ADPND simulará paralelamente las acciones de Mi y M2 y aceptará una cadena cuando y sólo cuando M\ y M

2

acepten una. De- finiremos el ADPND M = (Q, Z, T, s, z, F, A) como sigue:

*= DT x f*

S = (S\,S2) F = F\ x F2 \_\_. Z = Z i u Z

2

y la regla de transición A se define de forma que se tiene que

*((<?£, Pl), x) G A {{q¡, q}), a, b)*

si y sólo si

8

*(pj, a) = p¡ y (q¡c ,x )e Ai (q¡, a, b). Es decir,*

*A ((q¡, p j), a,b)= { ((<*

7

\*, p¡), x) I (qk, x) e Ai , «, ¿) y

8

*{pj, a) = {p¡ }}*

Obsérvese que los estados de M se etiquetan mediante pares que representan los estados en las que estarán M\ y Mi, respectivamente, tras procesar la entrada. Una cadena se acepta si provoca que M termine en un estado (q,p), donde q g F\ y p g Fi. Es decir, se acepta si M\ acepta la cadena acabando en el estado q y M2 acepta la cadena acabando en el estado p. La inducción sobre el número de pasos de una computación prueba que (

5

, w, z) f- ((q, p), £, u) con c/g F¡ y p g F2, si y sólo si (si, w, z) h (?, e, u) y

8

(.V 2

, w) =p. □

En la Sección 3.7, descubrimos que la intersección de lenguajes inde- pendientes del contexto no tenía que ser necesariamente un lenguaje inde- pendiente del contexto. Es decir, la familia de los lenguajes independientes del contexto no es cerrada bajo la intersección. El Teorema 3.8.3 prueba que esta fa- milia es cerrada bajo la intersección regular.

Por ejemplo, el lenguaje L = {a"b’l\n> 0 y n ¿ 100} es un lenguaje inde- pendiente del contexto. Aunque dicha propiedad puede ser probada mediante la construcción de una gramática independiente del contexto que lo genere o de un

www.FreeLibros.com

d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

160

®J V -ÜO 3J O •®ÍX O®O > ) ¡J Q Y •O HJ > UV -X O ¡J >

ADPND que lo acepte, es más elegante demostrarlo aplicando el teorema. Obsérvese que L\ = {a'’b"\n > 0} es un lenguaje independiente del contexto. Además, sabemos que L2 = a b \* y L

3

= {a100/?100} son regulares. Por tanto, L

*4*

= Lj - L-i también es regular, ya que los lenguajes regulares son cerrados con respecto a la diferencia de lenguajes. Entonces tendremos que L = L\ n L\ y, por tanto, según el teorema, L es independiente del contexto.

El Teorema 3.8.3 también se puede usar para demostrar que algunos len- guajes no son independientes del contexto. Por ejemplo, sabemos que {anb"cn | n > 0} no es independiente del contexto. Sea

*L - {w e {a, b, c}\*\w contiene el mismo número de aes, bes y ces)*

Entonces L no es independiente del contexto, porque si lo fuera, entonces

{anbncn | n > 0} = L n {a Ve\*) también lo sería.

2HJ-Á,Á,V> 3J ¡O CJÁÁ,ÍQ ext

3.8.1. Obtener una secuencia de descripciones instantáneas para el ADPND del Ejem- plo 3.8.1, que acepte la cadena babebab. ¿En qué punto fallaría el intento de aceptar la cadena abcabl

3.8.2. Obtener un ADPND que acepte el lenguaje generado por la siguiente gramática

independiente del contexto:

*S —) ciAA A —> bS\aS\a*

3.8.3. Obtener un ADPND que acepte el lenguaje generado por la gramática inde-

pendiente del contexto

*S —> aSb | aSbb I e*

3.8.4. Construir una ADPND que acepte el lenguaje generado por la gramática inde-

pendiente del contexto

*S —-> aABB | aAA A —^ aBB1 £ B bBB | A*

3.8.5. Demostrar cómo se convierte un ADPND arbitrario en uno para el cual todas

las transiciones son de la forma

A (q, a, A) = {ci, C 2

, ...»c„}, donde cada c¡ = (p, e) o c¡ = (p, BC).

www.FreeLibros.com

161

3.8.6. Obtener una derivación de a3b3 mediante la gramática del Ejemplo 3.8.2.

3.8.7. Aplicar la construcción precedente al ADPND dado por M = (Q, L, F, s, z, F, A),

donde Q = {<

71

, q2, qi, q4}, F={q3), s = q[, 1 = {a,b}, T = {A,B,z} y A como se muestra a continuación

*A{qua,z)={(q\,Az)} A (qiy b,A) = {(q2, £)} A (<?4, e, z) = {(< 7*

i, Az)} A (<

72

*, e, z) = {(q-i, e)} A(qi,a,A)= {(q4, e)}*

3.8.8. Obtener una gramática independiente del contexto que genere el mismo lengua- je que el ADPND M = (<2, E, T, s, z, F, A), donde Q= {q\,q2}, 2= {a, b], T = {a,z}, s = q\, F= {q2} y A dado mediante

*A(qu a, z)= {(tfi.az)} A(qu b, a)= {(q¡,aa)} A (<*

71

*, a, d)={(q*

2

,e)}

3.8.9. Obtener una grámatica independiente del contexto para el lenguaje aceptado por el ADPND M = (Q, E, T, s, z, F, A), donde Q = {q¡, q2, 93}, Z={a,b}, T = {A, z},

*b*

es el símbolo inicial de la pila, F = {<

73

}, y A dado por

*A (q¡, a, z) = {{qy,Az)} A(qi,a,A)= {(<?], A)} A b, A) = {(<*

72

» e)} A(q2, e,z)= {(43, e)}

Obsérvese que este ADPND no satisface las dos condiciones requeridas para la construcción de una GIC.

*3.8.10. Sea M = {Q, I, T, s, z, F, A), donde Q= [q\, q2, q^q\*}, E= {a, b}, T = {X, z},*

s = q\,z es el símbolo inicial de la pila, F = 0 (es decir, este ADPND sólo acepta cadenas cuando la pila se vacía), y A dado por

A (¿

7

*i, b, z) ={(qi,Xz)} A(qi,C,z) = :((<?i,e)'} A (qu b,X) = {(<*

71

*, XX)} A (q2, b, X) = {(q2, e)J A (< 7*

*i, a,X)= {(q2,X)} A(q2,a,z) = {(?i,z)}*

Después de realizar los cambios necesarios para que este ADPND satisfaga las condiciones requeridas para construir una gramática independiente del contexto, obtener una gramática independiente del contexto para el lenguaje aceptado por este autómata.

www.FreeLibros.com

d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

162

E2pG b j2 bME 7bEbC D d2ILMb62C 9pG7bd2C

3.8.11. Obtener una gramática independiente del contexto que genere el lenguaje acep-

tado por el ADPND

M=(Q, £, T, s, z, F, A) donde Q={quqi) Z={a,b} . . T = {A, z}, y z es el símbolo inicial de la pila s = q i F = \ qi} '

y A viene dado por

*A (<?i, a, z) = {{q\, Az)} A (<*

71

, b,A) = {(^i, AA)) A {q\, a, A) = {(<72, £))

3.8.12. Demostrar que

*{anbn | n >*

0

y n no es múltiplo de 5}

es un lenguaje independiente del contexto.

*3.8.13. Sea L = [a‘b^ck\i j o j^ k)*

(a) Probar que L es. un lenguaje independiente del contexto.

(b) Probar que si £ es cualquier alfabeto para el cual L c £\*, entonces £\* - L

a no \*T b \*

*e es ) .*

independiente del contexto. (Indicación: Considérese el lenguaje

3.8.14. Considérese el lenguaje formado por todas las cadenas sobre {a, b} que contie- nen el mismo número de aes que de bes, pero que no contienen la subeadena aab. ¿Es un lenguaje independiente del contexto?

3.8.15. Probar que el conjunto de los lenguajes independientes del contexto no es cerra- do con respecto a la complementación. Es decir, en general no es cierto que L y L\* - L sean lenguajes independientes del contexto.

exá 9pG7b IpG7bd j2 LG25ybu4

La forma normal de Choms2y es una de las formas normales que se usan para las gramáticas independientes del contexto. Otra forma normal que tiene una gran importancia teórica y práctica es la forma normal de Greibach. En la forma norma de Greibach, se restringe la posición en la que pueden aparecer los termi- nales y los no terminales. Empezaremos presentando"dos resultados usuales.

Teorema 3.9.1. Si A —» a B j es una producción de una gramática independiente del

contexto y si B -» Pi | P

2

I ••• ! son todas las producciones que tienen a B en

www.FreeLibros.com

163

su lado izquierdo, entonces la producción A —> aBy se puede reemplazar por A -» «Piyl ap

2

y | ... I a(3my sin que varíe el lenguaje generado por la gramática.

Demostración. Sea G la gramática original y sea G' la gramática que resulta de la transformación. Tenemos que probar que L (G) = L (G'). Primero demostrare- mos que L (G') c L (G). Supongamos que w e L (G'). Si la derivación de w no contiene ninguna de las producciones A —»a(3,y, entonces w e L(G) y queda probado. Por otro lado, si la derivación de w usa una producción A —> afi, y, en- tonces en G se puede usar A => aBy => cxp,- y para derivar w. Por tanto, vv e L (G). Entonces se tiene L (G') c L (G).

Para demostrar que L (G) c L (G'), consideremos que w e L (G). Si A —> aBy no se usa en la derivación de w, entonces, puesto que todas las otras producciones de G están en G', we. L (G') y quedará probado. Si n -? aBy se usa en la derivación de w, entonces en algún momento, B debe ser reemplazado por uno de los P¡. Esto se puede hacer de forma inmediata unificando ambos pa- sos en uno A => a(3, y. Pór tanto, w es derivable en G', con lo qué w e L (G') y se obtiene que L (G) c L (G')• □

Por ejemplo, consideremos la gramática independiente del contexto cuyas producciones son

*5 —» a | aa§L¡ Bab | abBc*

*4 P> —> haabS \ bba*

Si realizamos la sustitución presentada en el Teorema 3.9.1, obtendremos la gramática

*S —> a | acü¿acdiS \ aabba \ baabSab | bbaab*

*I abbacfbS c | abbbac 0 B —> baabñ\ bba*

Obsérvese que las producciones de B, B —> baabS| bba no desaparecen, pero no podrán formar parte de ninguna derivación. Sin embargo, con el fin de sim- plificar la gramática resultante, una de las técnicas de simplificación vistas en la Sección 3.5 podría eliminar dichas producciones inútiles.

Una producción de la forma A -» oo4, donde a e (N u Z)!í, se conoce como recursiva por la derecha. De forma semejante, una producción recursiva por la izquierda será de la forma A —> Aa. Las producciones recursivas por la derecha ^producen árboles que se expanden por la derecha, mientras que los árboles co- rrespondientes a las producciones recursivas por\*la izquierda se expanden por la izquierda. En muchas aplicaciones correspondientes a las gramáticas, no es de- seable que exista recursividad por la izquierda. El siguiente teorema proporciona

www.FreeLibros.com

d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

164

E2pG b j2 bME 7bEbC D d2ILMb62C 9pG7bd2C

una forma de eliminar la recursividad por la izquierda en las gramáticas inde- pendientes del contexto. Teorema P

3.9.2. Sea G una gramática independiente del contexto y A un no terminal de G. Si A -» A a

1

1 Aa.21... | Aa„ son todas las producciones para A, que son recursi- vas por la izquierda, y si A —> (3i | P

2

I •• ■ I Pm son las restantes producciones para A, entonces se puede construir una gramática equivalente introduciendo un nue- vo no terminal Z y reemplazando todas las producciones precedentes por

A - » pi | P

2

1... | p», | p,Z| p

2

Z |... | P„,Z Z -» ai | a

2

¡... | a„ | aiZ| a

2

Z |... | a„Z

Demostración. Obsérvese que en ambos casos las cadenas denvables de A mediante el

uso de una o más producciones, forman un lenguaje regulas\*,-

{pi,p

2

, {ai,a

2

,...,a„}\*. □

Considérese la gramática <r- '\*• independiente .

del contexto dada mediante

5 —> 5a|S¿>|c/4 A —^ Aa | a I £

Obsérvese que hay producciones recursivas por la izquierda con S y A en su lado izquierdo. Aplicando el teorema al no terminal S e introduciendo un no ter- minal nuevo Zi, se obtiene la siguiente gramática transformada que es inde- pendiente del contexto

*5 —> cA | cAZ\ Z\ —> a | b\aZ\ \ bZ\ ^ A a | ci IE*

Entonces, aplicando el teorema al no terminal A e introduciendo un no ter- minal nuevo Z

2

, se obtienen las producciones

6

*—^ cA j cAZ\ Z\ —> a | b | aZ\ | ¿>Zi*

A —> ci | C1Z21 £ | Z

2 Z

2

*-» a | C1Z2*

*%*

Obsérvese que al eliminar las producciones recursivas por la izquierda se introducen nuevos terminales y producciones recursivas por la derecha.

www.FreeLibros.com

165

Definición 3.9.3. Una gramática independiente del contexto está en forma normal de Greibach (FNG) si todas las producciones son de la forma A —> aa, donde a es un símbolo terminal y a e ( l u N)\*.

Obsérvese que esta formal normal requiere que toda producción tenga un símbolo del alfabeto como primer símbolo del lado derecho de las producciones. Por tanto una gramática en FNG no puede tener producciones recursivas por la izquierda. Es más, puesto que cada producción requiere que haya al menos un símbolo del alfabeto, una gramática independiente del contexto en FNG sólo puede generar lenguajes no vacíos que no contengan e.

Es posible construir una gramática independiente del contexto en forma normal de Greibach,para cualquier gramática independiente del contexto que no contenga e. El algoritmo para realizar esto consiste en diferentes etapas.

Supongamos que L es un lenguaje independiente del contexto no vacío que no contiene e. Primero, sea G = (£, N, S, P) una gramática independiente del contexto en forma normal de Choms2y que genera L. Supongamos que N = {A\, A2, ..., <4,ih donde Ai =S. Entonces modificamos las producciones de forma que si Ar —> A^a es una producción, entonces r<s. Supongamos que he- mos modificado las producciones de forma que para 1 < i < k, si A, —> Aj a, en- tonces i < j. Demostraremos como modificar las producciones para Ak+\.

*Si Ak*

+ 1

—> Aj a es una producción con k + 1 >j, se genera un nuevo conjun- to de producciones para reemplazar las Ay que aparecen en el lado derecho de las producciones, por el lado derecho de todas las producciones de la forma Aj —> (3, como se vio en el Teorema 3.9.1. Cuando se realicen dichas sustituciones, se ob- tendrán producciones de la forma Ak

+ 1

—>Ar a. Puede ocurrir que k + \ <r o que r < k + 1. Si k + 1 < r, se tendrá una producción de la forma deseada. Si r< k + 1, deberemos repetir el proceso. Puesto que solamente hay k índices me- nores que k +

1

, después de repetir el proceso k -

1

veces, se habrán eliminado todas las producciones de la forma Ak

+ 1

—> Ar a para las cuales r<k + 1. Por tanto, las producciones que quedarán, serán todas de la forma Ak

+ 1

—> Ar a, con k + 1 < r. Obsérvese que las producciones para las cuales k + i - r son produc- ciones recursivas por la izquierda que pueden ser eliminadas mediante el Teore- ma 3.9.2 introduciendo un no terminal nuevo Z\* + j.

Repetiremos el proceso para cada uno de los no terminales originales desde Ai hasta A„. Por tanto sólo se tendrán producciones de las tres formas siguientes:

1

*. Ak-> Aj a, con k <j*

2

. Ak -» aa, para a e £

3. Zk -» a, para a e (N u {Z¡, Z2, ..., Z„}>\*

www.FreeLibros.com

d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

166

E2pG5b j2 bME 7bEbC D d2ILMb62C 9pG7bd2C

Obsérvese que, puesto que An es el no terminal con mayor índice, todas las producciones para An deben ser del tipo 2. Es decir, en el extremo izquierdo del lado derecho de una producción para A„, debe haber un símbolo terminal. Tam- bién, toda producción para A„\_ i debe tener en el extremo izquierdo del lado de- recho, un terminal o An. Si tiene A„, puede ser reemplazado por el lado derecho de una de sus producciones como se hace en el Teorema 3.9.1. Dichas produc- ciones comenzarán con un terminal. Luego procederemos a transformar las pro- ducciones correspondientes a An- 2, A

„ \_ 3

y así sucesivamente, hasta que el lado derecho de todas las producciones correspondientes a los no terminales origina- les comiencen con un terminal.

Finalmente, consideraremos las producciones para Z\, Z% ..., Z„. Puesto que inicialmente teníamos una gramática independiente del contexto en forma nor- mal de Choms2y y sólo hemos aplicado los Teoremas 3.9.1 y 3.9.2, ninguna de las producciones Z, —> a tendrá otro Z¡ en el extremo izquierdo de su lado dere- cho. Por tanto, todas la producciones correspondientes a los Z/ tendrán termina- les o A¡ al principio de su lado derecho. Para los que tengan al principio del lado derecho alguna A¿, aplicaremos una vez más, el Teorema 3.9.1 y todas las pro- ducciones resultantes estarán en la forma deseada.

La transformación vista nos da pie para enunciar el siguiente teorema:

Teorema 3.9.4. Todo lenguaje independiente del contexto no vacío, que no contiene la palabra vacía e, se puede generar mediante una gramática independiente del con- texto G = (¿V, £, S, P) en la cual todas las producciones son de la forma A —> ciw para A e N , a e ' L y w e N \* .

Consideremos la gramática independiente del contexto en forma normal de Choms2y (cuyos no terminales han sido etiquetados convenientemente)

*Ai —> A2 A21 ci A2 —> A*

1

*A2*

1

*b*

Obsérvese que las producciones A

1

—> A

2

A

2

I a ya se encuentran en la forma necesaria para realizar la primera etapa. (Las producciones deben satisfacer A¡ -» Aj a para i < j sólo cuando hay un no terminal en el lado derecho. Por tanto Ai —> a es aceptable). Consideraremos las producciones correspondientes a A2. La producción A2 —> b se acepta, pero A2 —> A1A2 no. Debemos substituir A

1

, ob- teniendo las producciones A2 —> A2A2A2 \ C1A2. Eliminamos la recursividad por la izquierda y obtenemos el conjunto de producciones\*

%

Aj —> A

2

A

2

IÍÍ A

2

*—> a M | aAiZ\ b \ bZ*

www.FreeLibros.com

167

Finalmente, substituimos Ai de forma apropiada para que el lado derecho de todas las producciones comience con un terminal. Esto produce

*Ai - » aA2A2\aA2ZA2\bA2\bZA2\a Af —> aA 2\aA 2Z\ b\ bZ*

*Z —> aA2A2\aA2ZA2\bA2\bZA2\aA2A2Z*

*\aA fZA fZ \ bA fZ \ b Z A fZ*

Obsérvese que todas las producciones de la forma dada en el Teorema 3.9.4 son necesariamente de la forma apropiada para la FNG. Por tanto, se obtiene el siguiente corolario:

Corolario 3.9.5. Todo lenguaje L independiente del contexto y no vacío, que r.o con- tenga e, puede ser generado mediante una gramática independiente del contexto en forma normal de Greibach.

Una definición alternativa para la FNG requiere que todas las producciones sean de la forma A —> aw para A e N, a e E y w e N\*. Las dos definiciones de FNG son equivalentes (Ejercicio 3.9.5).

2HJ-Á,Á,V> 3J ¡O CJÁÁ,ÍQ exá

3.9.1. Eliminar la recursividad por la izquierda de

*S —> Sa | aAc | c -4 —>Ab\ba*

3.9.2. Eliminar la recursividad por la izquierda de S —> aSb | e.

3.9.3. En las observaciones anteriores al Teorema 3.9.3, la afirmación “ninguna de las producciones Z¡ —> a, tienen otro Zj en el extremo izquierdo del lado derecho de la producción”, depende de que el lado derecho de todas las producciones correspondientes a A, empiece con un terminal o con Aj A¡, para algún j y algún k. Probarlo por inducción sobre el número de aplicaciones del Teorema 3.9.1 y 3.9.2.

3.9.4. Pasar a forma normal de Greibach

(a) 5 —» aSb | ab

(b) S —> AA | a

*A->SS\b*

*(c) S —> Sa | Sb | cA*

A —) A a | o I £

www.FreeLibros.com

d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

168

E2pG5b j2 bME 7bEbC D d2ILMb62C 9pG7bd2C

3.9.5. Probar que la siguiente definición de forma normal es equivalente a la Defini- ción 3.9.3: Una gramática independiente del contexto G = (N, £. S, P) está en forma normal de Greibach si toda producción de P es de la forma /i —> aw para A e N, a e Zy we N\*.

íGpyd27bC

3.1. Consideremos el lenguaje L = {a'¿'c'|¿>l} del Ejercicio 3.6.1 (a). Este lenguaje no es un lenguaje independiente del contexto puesto que, para un k dado, pode- mos elegir z = akbkck y demostrar que ninguna elección posible de las subcade- nas u, v, w, x e y satisface el lema de bombeo (3.6.1). Este lenguaje presenta uno de los mayores inconvenientes que tiene el lema de bombeo: no hay forma de decir qué parte de la cadena puede considerarse como la parte uwx. Esta in- certidumbre nos lleva a considerar una variedad de subcasos que pueden resul- tar tediosos, a pesar de usar una técnica efectiva. En este primer ejercicio trata- remos de presentar una versión más fuerte del lema de bombeo que nos permita controlar el enfoque que se dé a las partes de la cadena que son bombeadas.

Sea G = (N, L, 5, P) una gramática independiente del contexto en forma normal de Choms2y con ¡ /Vj = k, y sea n = 2k + 1. Supongamos que z £ ¿ (G) con |z¡ > n, y que “marcamos” al menos n posiciones de z (es decir, queremos distinguir las partes marcadas de las que no lo están). Queremos encontrar un camino en el árbol de derivación para w, análogo al visto en el lema de bombeo. Construiremos el camino partiendo de la raíz y descendiendo por la izquierda o por la derecha. Descenderemos por el subárbol izquierdo o por el subárbol dere- cho para determinar que subárbol genera el mayor número de posiciones de z que están marcadas. El camino resultante se extiende desde la raíz hasta una de las posiciones marcadas.

Obsérvese que cualquiera de los nodos del árbol de análisis (y que están presentes en el camino que hemos encontrado), tienen cero, uno o dos hijos, y cada uno de los hijos es, a su vez, raíz de un subárbol que puede generar o no posiciones marcadas. Si un nodo del árbol tiene dos hijos que son raíz, de subár- boles que generan posiciones marcadas, Se llamará punto de rama.

1. Probar que, para cada punto de rama presente en el camino que hemos construido, tenemos al menos la mitad de descendientes marcados que para el punto de rama anterior.

2. Probar que el camino construido contiene al menos k + 1 puntos de rama.

El resultado visto es un resultado más débil que el que se debe a W. Og- den. Nos referiremos a él como “lema de Qgden”.

Lema de Ogden. Sea L un lenguaje independiente del contexto. Entonces existe una constante n para la cual, si z e L y tenemos al menos ;? posicio- nes de z marcadas, podemos escribir que z = uvwxy, de forma que

www.FreeLibros.com

169

1

. vx tiene al menos una posición marcada.

2

. vwx tiene como máximo n posiciones marcadas.

3. Para todo i > 0, uv ‘wx'y e L.

3. Demostrar el lema de Ogden. Indicación: Por el Ejercicio 2, hay al menos k+ 1 puntos de rama en un camino elegido convenientemente. Apliqúese la demostración del lema de bombeo para el último de los k +

1

.

2. En los problemas siguientes se puede usar el lema de Ogden.

1. Aplicar el lema de Ogden para demostrar que el lenguaje

(aW 'l i >

1

}

no es independiente del contexto.

2. Usar el lema de Ogden para demostrar que

*{a‘bjckdl\i = 0 o j = k = l}*

no es un lenguaje independiente del contexto.

3. Usar el lema de Ogden para demostrar que

*{a!b]ck\ i < j < k]*

no es independiente del contexto.

4. ¿Es un lenguaje independiente del contexto el lenguaje

*{a'bjck\i&j, j \* k e i ^ k}l*

¿Por qué o por qué no? ¿Qué ocurre con

*{a‘bjck\i\* j y j \* k) l*

3. Sea L = {ww\w e [a, ¿>}\*}. En el Ejercicio 3.6.1 (g) se propuso que se demos- trará que L no era independiente del contexto. Demostrar que L, el complemen- to de L en {a, b}\*, es independiente del contexto. Indicación: Obsérvese que cualquier cadena de longitud impar sobre {a, b}\*, automáticamente pertenece a L. Por otro lado, cualquier cadena de longitud par de L se puede escribir como ii\X\V\UiX2V7, donde xi y x% están en {a,b},x \\*X2y |mi| = |

«21

y 2il = IV

21

- Considérese la unión de dos lenguajes independientes del contexto, uno de los cuales contenga sólo las cadenas de longitud impar sobre {a, b}’ y el otro con- tenga las cadenas de la forma U\X\V\U2X2Vz.

www.FreeLibros.com

d2ILMb62C 5Ij2í2Ij52IE2C j2d upIE2SEp

170

E2pG5b j2 bME 7bEbC D d2ILMb62C 9pG7bd2C

3.4. Un tipo de autómata similar al ADPND, es- el autómata de pila con dos pilas y es una 7-tupIa M = (Q, Z, T, A, s, F, z). En éste, Q, Z, T, s y F son los mismos que en los ADPND, pero la relación de transición A es de la forma

A c Q x f Z u {e})x rxrx<

2

x r \* x r \*

Obsérvese que, cuando describimos una transición de A, se tiene en cuenta un símbolo de cada una de las pilas. Por ejemplo, una transición de la forma A (q, a, Ti, x2) = {(p, a, e)} cambiaría del estado q al estado p, sustituiría el sím- bolo Ti de la primera pila por el símbolo (o cadena) a, y desapilaría el símbolo %2 de la segunda pila.

1. Probar que cualquier lenguaje independiente del contexto es aceptado por

un autómata de pila con dos pilas. 2. Probar que el lenguaje L= {a'b'c1 | i > 0} es aceptado por un ADP con dos

pilas.

3. Describir una técnica por medio de la cual el lenguaje L sea reconocido por

un ADP con dos pilas

*L= {a'b‘c'd'\i>0}*

3.5. Se dice que el deterninismo está presente en un autómata cuando no se puede elegir cómo se comportará en cada estado. Una condición que debe cumplirse para que un autómata de pila sea determinista es que no puede haber más de un elemento en los conjuntos A (q, o, x). Sin embargo, esto no es suficiente para garantizar el determinismo. Considérese el ADP que contiene las transiciones

*A (q, a,A)={ (p, w)} A (q, e,A)={(p\w')}*

Obsérvese que este ADP tiene que elegir cómo comportarse una vez que se encuentra en el estado q. Por tanto, incluso si todos los conjuntos A (q¡, o, t) contienen un único elemento, puede seguir habiendo no determinismo. Defini- mos un autómata de pila determinista (ADPD) como un ADP en el cual no hay ninguna configuración para la cual el ADP tenga que elegir entre más de un mo- vimiento. En otras palabras, un ADP es determinista si (a) cada A (q, a, A) tiene un elemento como máximo, y (b) si A (q, a, A) s\* 0, entonces A (q, e, A) = 0. Un lenguaje independiente del contexto es un lenguaje independiente del contexto determinista (LICD) si es aceptado por un ADPD.

1. Probar que todo lenguaje regular es un LICD.

2. Probar que no todo LICD es regular.

3. Obviamente, cualquier LICD es un LIC ya que todo ADPD es un ADPND. Usar esto para demostrar que no todo LIC es un LICD. Indicacióh: En el Problema 3.3, se obtuvo un lenguaje independiente del contexto cuyo com- . plemento no era independiente del contexto.

www.FreeLibros.com