Teoría de la computación

Victor Melchor

Clase 16

Contenido de la clase

Capítulo 1

Autómatas Finitos y Gramáticas Regulares

En 1956 Noam Chomsky definió una jerarquía formada por 4 tipos de gramáticas en función de restricciones sobre las reglas.

Estas van de lo más general(tipo 0) a lo más específico(tipo 3).

La Jerarquía de Chomsky consta de cuatro niveles:

1. Gramáticas regulares (gramáticas de tipo 3)
2. Gramáticas sin restricciones (gramáticas de tipo 0)
3. Gramáticas sensibles al contexto (gramáticas de tipo 1)
4. Gramáticas libres del contexto (gramáticas de tipo 2)

1.1 Gramáticas regulares

Se les llama también lineales y es el grupo más restringido de gramáticas. Esta restricción consiste en que la parte derecha de la regla tendrá como máximo dos símbolos.

1.1.1 Clasificación

Hay dos tipos de gramáticas regulares:

1. **Gramáticas lineales por la derecha (GLD)** Sus reglas son de la forma:

$$\begin{aligned} A &::= a \\ A &::= aB \qquad \text{donde } A, B \in \Sigma_N,\ a \in \Sigma_T \\ A &::= \varepsilon \end{aligned}$$

2. **Gramáticas lineales por la izquierda (GLI)** Sus reglas son de la forma:

$$A ::= a$$
$$A ::= Ba$$
$$A ::= \varepsilon$$

donde $A, B \in \Sigma_N,\ a \in \Sigma_T$

Estos lenguajes son aquellos que pueden ser aceptados por un autómata finito. También esta familia de lenguajes pueden ser obtenidas por medio de expresiones regulares.

Teorema: La clase de los lenguajes generados por una gramática regular es precisamente la de los lenguajes regulares.

1.2 Procedimiento de Conversión de GR a AF

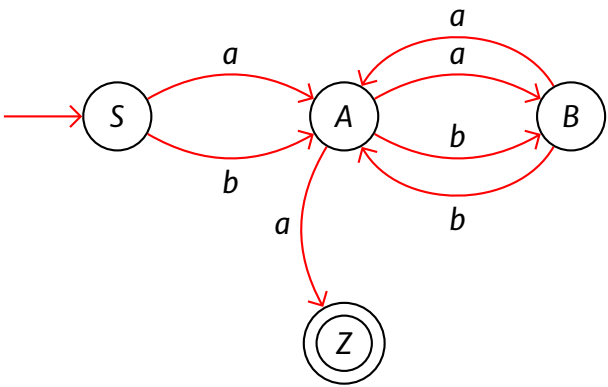
1. Para cada símbolo no terminal de la gramática asociar un estado en el autómata.
2. Para cada regla de la forma $A ::= bC$ en la gramática se origina una transición (A, b, C) en el autómata.
3. Para cada regla de la forma $A := b$ se tendrán transiciones (A, b, Z) donde Z será el único estado final del autómata.

Ejemplo: A partir de la gramática regular $G = (\Sigma_N, \Sigma_T, S, P)$, donde el conjunto P está dado por

$$P \begin{cases} \text{S::=aA} \\ \text{S::=bA} \\ \text{A::=aB} \\ \text{A::=bB} \\ \text{A::=a} \\ \text{B::=aA} \\ \text{B::=bA} \end{cases}$$

obtener un autómata finito

Solución

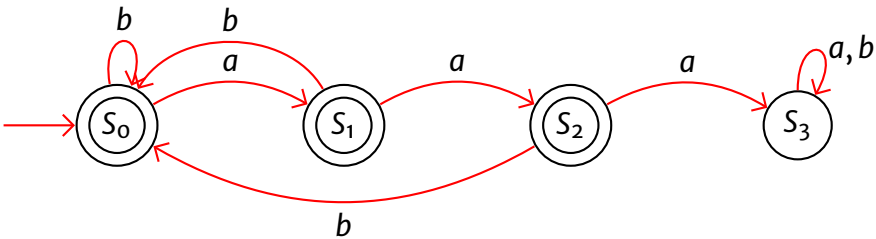


1.3 Procedimiento de Conversión de un AFD a GR

1. Para cada transición de la forma $((p, a), q)$ en el AFD habrá una regla en la gramática $X_p ::= aX_q$ donde X_i es la variable que corresponde al estado i del AFD

2. Por cada transición $((p, a), q)$ donde $q \in F$ incorporar a la gramática una regla adicional $X_p ::= a$ además de la regla $X_p = aX_q$

Ejemplo: A partir del AFD con diagrama de transición



obtener la gramática regular correspondiente

Solución

$$\Sigma_N = \{s_0, s_1, s_2, s_3\}$$
$$\Sigma_T = \{a, b\}$$
$$S = s_0$$

Las reglas para G son

$$
P\begin{cases}
S_0 ::= aS_1 \\
S_0 ::= bS_0 \\
S_1 ::= aS_2 \\
S_1 ::= bS_0 \\
S_2 ::= aS_3 \\
S_2 ::= bS_0 \\
S_3 ::= aS_3 \\
S_3 ::= bS_3 \\
S_0 ::= a \\
S_0 ::= b \\
S_1 ::= a \\
S_1 ::= b \\
S_2 ::= b
\end{cases}
$$

1.4 Conversión de AFD a GR

Dado un AFD $M = (S, I, \delta, s^*, F)$ se desea encontrar una GLD G tal que $L(M) = L(G)$

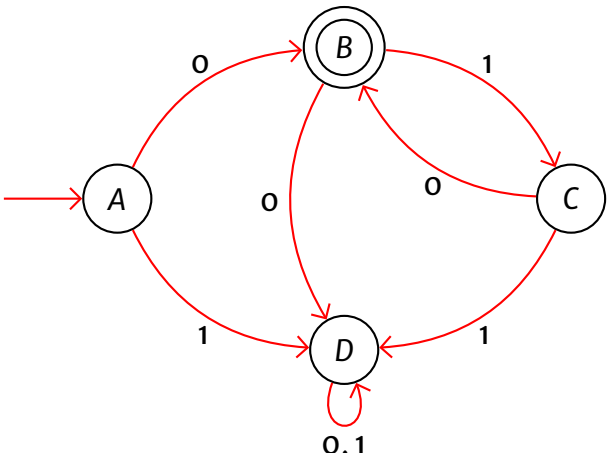
Si q no es un estado final, la gramática buscada es:

$$G = (\Sigma_N, \Sigma_T, q, P)$$

donde P consiste de producciones de la forma:

$$p ::= aq \qquad \text{si } \delta(p, a) = q$$
$$p ::= a \qquad \text{si } \delta(p, a) \in F$$

Ejemplo: Sea el AFD *M* dado por el diagrama de transición:



Obtener la gramática G tal que L(M)=L(G).

Solución

Las reglas para G son:

$$\mathcal{P}\begin{cases} \text{A::=0B|1D|0} \\ \text{B::=0D|1C} \\ \text{C::=0B|1D|0} \\ \text{D::=0D|1D} \end{cases}$$

debido a que D no deriva nada, lo omitimos. Si además, renombramos A por S obtenemos:

$$\mathcal{P}\begin{cases} \text{S::=0B|0} \\ \text{B::=1C} \\ \text{C::=0B|0} \end{cases}$$

Si q es final, es decir $\varepsilon \in L(M)$, la gramática G definida anteriormente generaría $L(M) - \{\varepsilon\}$

Podemos modificar G, agregándole un nuevo estado inicial S con producciones:

$$S ::= q|\varepsilon$$

La nueva gramática es derecha y genera $L(M)$.

Para producir una GLI para L,

1. Se construye un AF para L_{rev}
2. Se obtiene una gramática derecha G
3. Se invierten los lados derechos de las producciones obteniendo la gramática izquierda deseada.

1.5 Conversión de GR a AFND

Dada una GLD $G = (\Sigma_N, \Sigma_T, S, P)$, es posible encontrar un AFND-ε $M = (S^{nd}, I^{nd}, \delta^{nd}, s^*, F^{nd})$ tal que $L(M) = L(G)$

Sean:

1. $I^{nd} = \Sigma_T$
2. Los estados S^{nd} de M serán S además de los sufijos de partes derechas de las producciones.
3. $s^* = S$
4. Para definir la función de transición δ se tiene:

   Si $\alpha ::= \beta \in P$ entonces se define $\delta(\alpha, \varepsilon) = \beta$

   Si $a\alpha \in S^{nd},\ a \in I^{nd}$ se hace $\delta(a\delta, a) = \alpha$

5. Los estados finales F^{nd} de M serán los estados rotulados con ε

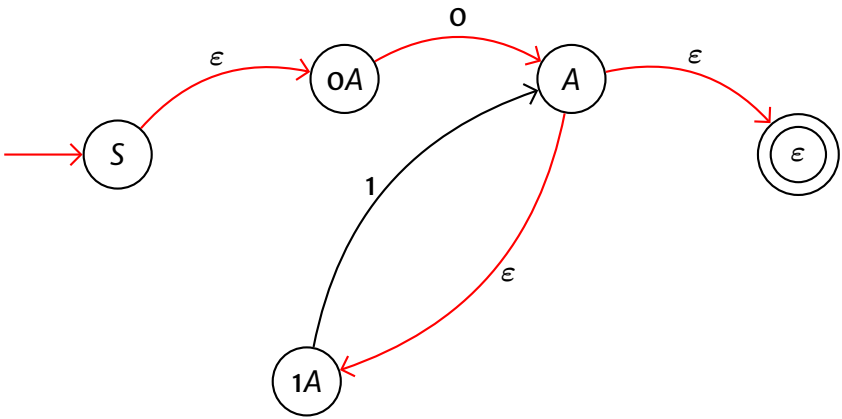
Ejemplo: Sea G la GLD definida por:

$$S ::= 0A$$
$$A ::= 1A|\varepsilon$$

Obtener el AF correspondiente

Solución

1. $I^{nd} = \Sigma_T = \{0, 1, \varepsilon\}$
2. $S^{nd} = \{S, 0A, A, 1\,A, \varepsilon\}$
3. $s^* = S$



4.

Por otro lado, si G es una GLI y el conjunto de sus producciones

es

$$P = \{A ::= \alpha\}$$

entonces

1. Se define una GLD G' cuyas producciones son:

$$P' = \{A ::= \alpha^{rev} / A ::= \alpha \in \mathcal{P}\}$$

2. Construir el AF para G'
3. Invertir el autómata

1.6 Gramáticas Sensibles al Contexto

Una gramática sensible al contexto, o gramática tipo 1 tiene producciones de la forma:

$$xAy \rightarrow xvy$$

donde $A \in \Sigma_N,\ v \in \Sigma^+, \quad x, y \in \Sigma^*$

Estas gramáticas tienen como restricción que la longitud de la parte derecha de las producciones es siempre mayor o igual que la longitud de la parte izquierda, es decir, no hay producciones compresoras.

Las lenguajes generados por éstas gramáticas se llaman lenguajes sensibles al contexto y su clase es $\mathcal{L}_1$

Ejemplo: Sea la gramática G

$$S ::= abc|aAbc$$
$$A ::= abC|aAbC$$
$$Cb ::= bC$$
$$Cc := cc$$

G es una gramática sensible al contexto

$$L(G) = \{a^n b^n c^n / n \geq 1\}$$

1.7 Gramáticas sin Restricciones

Llamadas también gramáticas recursivamente enumerables, o gramática tipo 0 (cero).

Sus predicciones son de la forma:

$$xAy ::= v$$

donde $A \in \Sigma_N, \quad x, y, v \in \Sigma^*$

Se demuestra que cualquier lenguaje de tipo 0 puede también ser generado por las gramáticas con estructura de frase.

Las producciones de las gramáticas con estructura de frase tienen la forma:

$$xAy ::= xvy$$

donde $A \in \Sigma_N, \quad x, y, v \in \Sigma^*$

Los lenguajes generados por éstas gramáticas se llaman Lenguajes con Estructura de Frase y se agrupan en la clase $\mathcal{L}_0$

Ejemplo: La gramática G:

$$aS \to bSb$$
$$aSb \to \varepsilon$$
$$SbS \to bcS$$

es una gramática sin restricciones.

1.8 Gramáticas Libres de Contexto

Se les llama también gramáticas de tipo 2, se caracterizan porque la parte izquierda de la producción está formada por un único símbolo no terminal. Siempre que se encuentra A se puede sustituir por su lado derecho

$$A ::= \nu \qquad \text{donde } A \in \Sigma_N;\ \nu \in \Sigma^*$$

éstas gramáticas son especialmente adecuadas para representar los aspectos sintácticos de cualquier Lenguaje de Programación.

Se observa que la definición incluye a la regla $S \to \varepsilon$.

Ejemplo: En cada inciso se presenta una gramática. Obtenga el lenguaje que genera dicha gramática:

1. Sea $\Sigma_T = \{a, b\}$ y sea $\mathcal{P}$ dado por

$$S ::= \varepsilon | aSb$$

Entonces $L(G) = \{a^n b^n / n \geq 0\}$

2. Sea $\Sigma_T = \{a, b\}$ y sea $\mathcal{P}$ dado por:

$$S ::= aSa|bSb|a|b|\varepsilon$$

Entonces $L(G) = \{w \in \Sigma_T^* / w = w^R\}$

3. Sea $\Sigma_T = \{a, b\}$ y sea $\mathcal{P}$ dado por:

$$S ::= aS|aB$$
$$B ::= bC$$
$$C ::= aC|a$$

Entonces $L(G) = \{a^n b a^m / n, m \geq 1\}$

4. Sea $\Sigma_T = \{(,)\}$ y sea $\mathcal{P}$ dado por:

$$B ::= \varepsilon | BB | (B)$$

Con esta gramática se obtiene las cadenas de paréntesis bien balanceados.

5. Sea $\Sigma_T = \{i, e\}$ y sea $\mathcal{P}$ dado por:

$$S ::= \varepsilon || SS | iS | iSeS$$

esta gramática expresa el tratamiento del `if` y `else` en un LP.

Se cumple

- Un `if` puede usarse sin ningún `else` correspondiente
- Un `if` puede ir balanceado con un `else`
- La concatenación de dos sucesiones `if else` válidas sigue siendo válida.

Sucesiones Válidas:

ieie, *iei*, *iie*, *iieie*, etc.

Sucesiones Incorrectas:

ei, *ieeii*