Introducción

Utilizando los autómatas que estudiamos en el tema anterior, los autómatas finitos, vimos cómo se puede diseñar el analizador léxico de un compilador. Sin embargo, los autómatas finitos no se pueden utilizar para diseñar el analizador sintáctico del compilador. Para ello, nos hace falta utilizar autómatas más complejos, que son los llamados autómatas con pila, los cuales tienen asociada una pila que utilizan como elemento de memoria. Al igual que los autómatas finitos, los autómatas con pila tienen también asociada una cinta de entrada dividida en celdas, en las cuales se registra la palabra de entrada que suministremos al autómata.

Al final de la última clase, empezamos el estudio de los autómatas con pila. En la clase de hoy, estudiaremos con más profundidad estos autómatas. Empezamos recordando el concepto de autómata con pila.

Definición de un autómata con pila

Un autómata con pila es una estructura $M = (K, \Sigma, \Gamma, \Delta, q_0, F)$ donde:

(1) K es un conjunto finito y no vacío de estados.

(2) Σ es un alfabeto finito, el alfabeto de entrada.

(3) Γ es un alfabeto finito, el alfabeto de la pila.

(4) $q_0 \in K$ es el estado inicial,

(5) $F \subseteq K$ es el conjunto de estados aceptadores (o estados finales),

(6) Δ es un conjunto finito de elementos de la forma $((p, a, b), (q, x))$ donde $p, q \in K$, $a \in \Sigma \cup \{\lambda\}$, $b \in \Gamma \cup \{\lambda\}$ y $x \in \Gamma^*$.

Conceptos básicos

Si $M = (K, \Sigma, \Gamma, \Delta, q_0, F)$ es un autómata con pila, llamaremos transiciones a los elementos de Δ.

Al empezar el cómputo en un autómata con pila $M = (K, \Sigma, \Gamma, \Delta, q_0, F)$, estaremos en el estado inicial q_0, tendremos una palabra $x \in \Sigma^*$ en la cinta de entrada y tendremos la pila vacía.

Si en un paso de cómputo de M, queremos aplicar una transición $((p, a, b), (q, x))$ donde $a \in \Sigma$ y $b \in \Gamma$, el símbolo b tiene que estar situado en la cima de la pila. Al aplicar entonces la transición, sucede lo siguiente:

(a) se pasa del estado p al estado q,

(b) se lee el símbolo a de la cinta de entrada,

(c) se reemplaza en la pila el símbolo b por x.

Análogamente, si en un paso de cómputo de M, queremos aplicar una transición $((p, \lambda, b), (q, x))$ donde $b \in \Gamma$, el símbolo b tiene que estar situado en la cima de la pila. Al aplicar entonces la transición, sucede lo siguiente:

(a) se pasa del estado p al estado q,

(b) no se lee ningún símbolo en la cinta de entrada,

(c) se reemplaza en la pila el símbolo b por x.

De manera similar se procede con los demás tipos de transiciones.

Conceptos básicos

Si $M = (K, \Sigma, \Gamma, \Delta, q_0, F)$ es un autómata con pila, definimos una configuración de M como una palabra $\alpha p x \in \Gamma^* K \Sigma^*$.

Si en un paso de cómputo la configuración de M es $\alpha p x$, esto significa que el autómata M se encuentra en el estado p, x es la parte de la palabra de entrada que todavía no se ha leído y α es el contenido de la pila.

La noción de configuración nos permite entonces definir la noción de cómputo en un autómata con pila.

Cómputos en autómatas con pila

Supongamos que $M = (K, \Sigma, \Gamma, \Delta, q_0, F)$ es un autómata con pila y que $\alpha p x, \beta q y$ son configuraciones de M. Entonces:

(1) Decimos que $\alpha p x$ produce $\beta q y$ en un paso de cómputo, lo que representamos por $\alpha p x \vdash_M \beta q y$, si en M podemos pasar de $\alpha p x$ a $\beta q y$ aplicando una transición de Δ.

(2) Decimos que $\alpha p x$ produce $\beta q y$, lo que representamos por $\alpha p x \vdash_M^* \beta q y$, si en M podemos pasar de $\alpha p x$ a $\beta q y$ aplicando un número finito de de transiciones de Δ.

Lenguaje asociado a un autómata con pila

Supongamos que $M = (K, \Sigma, \Gamma, \Delta, q_0, F)$ es un autómata con pila.

(1) Una palabra $x \in \Sigma^*$ es reconocida (o aceptada) por M, si existe $q \in F$ tal que $\lambda q_0 x \vdash_M^* \lambda q \lambda$.

(2) Definimos el lenguaje reconocido (o aceptado) por M por

$$L(M) = \{x \in \Sigma^* : x \text{ es reconocida por } M\}.$$

Por tanto, para que una palabra sea reconocida por un autómata con pila M, tiene que haber un cómputo en el autómata M que lea toda la palabra de entrada de manera que al final del cómputo se llegue a un estado aceptador y la pila quede vacía.

Ejemplo

Consideremos el autómata con pila $M = (K, \Sigma, \Gamma, \Delta, q_0, F)$ donde $K = \{q_0, f\}$, $\Sigma = \{a, b, c\}$, $\Gamma = \{a, b\}$, $F = \{f\}$ y Δ consta de las siguientes transiciones:

1. $((q_0, a, \lambda), (q_0, a))$.
2. $((q_0, b, \lambda), (q_0, b))$.
3. $((q_0, c, \lambda), (f, \lambda))$.
4. $((f, a, a), (f, \lambda))$.
5. $((f, b, b), (f, \lambda))$.

Tenemos entonces el siguiente cómputo para la entrada $abbcbba$.

Ejemplo

estado	cinta	pila	transición
q_0	abbcbba	λ	–
q_0	bbcbba	a	1
q_0	bcbba	ba	2
q_0	cbba	bba	2
f	bba	bba	3
f	ba	ba	5
f	a	a	5
f	λ	λ	4

Ejemplo

Se observa que el autómata M, cuando está en el estado q_0, mete una a en la pila cada vez que lee una a en la cinta aplicando la transición 1, y mete una b en la pila cada vez que lee una b en la cinta aplicando la transición 2. Y así actúa el autómata hasta que entra una c en la cinta, en cuyo caso pasa al estado aceptador f aplicando la transición 3. Una vez que el autómata está en el estado f, cancela las a's que entren en la cinta con las a's que se encuentran en la pila aplicando la transición 4, y asimismo cancela las b's que entren en la cinta con las b's que se encuentran en la pila aplicando la transición 5. Entonces, para que al final del cómputo la pila quede vacía, ha de suceder que la parte de la entrada que aparezca después de la c en la entrada ha de coincidir con la inversa de la parte de la entrada que aparece antes de la c en la entrada. Por tanto,

$$L(M) = \{xcx^I : x \in \{a, b\}^*\}.$$

Autómatas con pila deterministas

Nuestro objetivo ahora es definir el concepto de autómata con pila determinista, que es el modelo de autómata con pila que se puede programar directamente. Previamente, necesitamos definir cuando dos transiciones son compatibles.
Sea $M = (K, \Sigma, \Gamma, \Delta, q_0, F)$ un autómata con pila.

(a) Dos transiciones $((p_1, a_1, b_1), (q_1, \alpha_1))$ y $((p_2, a_2, b_2), (q_2, \alpha_2))$ de M son compatibles, si se cumplen las tres siguientes condiciones:

(1) $p_1 = p_2$.

(2) $a_1 = a_2$ o $a_1 = \lambda$ o $a_2 = \lambda$.

(3) $b_1 = b_2$ o $b_1 = \lambda$ o $b_2 = \lambda$.

(b) M es determinista, si no tiene dos transiciones compatibles distintas.

Programación de autómatas con pila deterministas

Obsérvese que el que dos transiciones sean compatibles en un autómata con pila significa que las dos se pueden aplicar en un mismo paso de cómputo del autómata. Por tanto, en cualquier paso de cómputo de un autómata con pila determinista habrá a lo sumo una transición que se pueda aplicar. Esto hace que los autómatas con pila deterministas se puedan programar. Para ello, se procede de manera similar a como vimos con los autómatas deterministas finitos. Si programamos un autómata con pila determinista en Java, utilizaremos una variable tipo stack de Java para manejar la pila asociada al autómata y utilizaremos una variable booleana que tendrá el valor verdadero cuando el cómputo del autómata quede bloqueado (por no poder aplicar ninguna transición del autómata), en cuyo caso se saldrá del bucle "while" del programa para devolver el valor "false".

Ejemplo 1

Consideremos el autómata con pila M visto en el ejemplo anterior. Es decir, $M = (K, \Sigma, \Gamma, \delta, q_0, F)$ donde $K = \{q_0, f\}$, $\Sigma = \{a, b, c\}$, $\Gamma = \{a, b\}$, $F = \{f\}$ y Δ consta de las siguientes transiciones:

1. $((q_0, a, \lambda), (q_0, a))$.
2. $((q_0, b, \lambda), (q_0, b))$.
3. $((q_0, c, \lambda), (f, \lambda))$.
4. $((f, a, a), (f, \lambda))$.
5. $((f, b, b), (f, \lambda))$.

Se tiene entonces que M es determinista, porque no tiene dos transiciones compatibles distintas.

Ejemplo 2

Consideremos ahora el autómata con pila $M' = (K, \Sigma, \Gamma, \delta, q_0, F)$ donde $K = \{q_0, f\}$, $\Sigma = \{a, b\}$, $\Gamma = \{a, b\}$, $F = \{f\}$ y Δ consta de las siguientes transiciones:

1. $((q_0, a, \lambda), (q_0, a))$.
2. $((q_0, b, \lambda), (q_0, b))$.
3. $((q_0, \lambda, \lambda), (f, \lambda))$.
4. $((f, a, a), (f, \lambda))$.
5. $((f, b, b), (f, \lambda))$.

Tenemos entonces que M' no es determinista, ya que las transiciones 1 y 3 son compatibles, y asimismo las transiciones 2 y 3 son compatibles.

Los autómatas M y M' son muy similares. Su única diferencia está en la transición 3, que en M es $((q_0, c, \lambda), (f, \lambda))$, y en M' es $((q_0, \lambda, \lambda), (f, \lambda))$. El resto de las transiciones son las mismas en M y en M'.

Ejemplo 2

Por tanto,

$$L(M') = \{xx^I : x \in \{a, b\}^*\}.$$

El siguiente cómputo en M' reconoce la palabra $abbbba$.

Ejemplo 2

estado	cinta	pila	transición
q_0	abbbba	λ	–
q_0	bbbba	a	1
q_0	bbba	ba	2
q_0	bba	bba	2
f	bba	bba	3
f	ba	ba	5
f	a	a	5
f	λ	λ	4

Gramáticas incontextuales

Como ya hemos indicado anteriormente, los autómatas con pila se utilizan para poder diseñar el analizador sintáctico del compilador. Sin embargo, los autómatas con pila son difíciles de diseñar directamente. Para poder diseñarlos, se utilizan las gramáticas incontextuales, que son estructuras equivalentes a los autómatas con pila, pero mucho más fáciles de diseñar.

Como veremos, toda gramática incontextual tiene asociado un autómata con pila equivalente a la gramática. Entonces, una vez que tengamos construida la gramática incontextual adecuada para el lenguaje que estemos considerando, tomaremos el autómata con pila equivalente a la gramática, y a continuación procederemos a eliminar el indeterminismo de dicho autómata.

Definición de gramática incontextual

Una gramática incontextual es una estructura $G = (V, \Sigma, P, S)$ donde:

(1) V es un alfabeto, a cuyos símbolos se les llama variables.

(2) Σ es un alfabeto disjunto de V (es decir, $V \cap \Sigma = \emptyset$), a cuyos símbolos se les llama terminales.

(3) P es un subconjunto finito de $V \times (V \cup \Sigma)^*$, a cuyos elementos se les llama producciones (o reglas).

(4) $S \in V$ es la variable inicial.

Derivaciones en gramáticas incontextuales

Sea $G = (V, \Sigma, P, S)$ una gramática incontextual.

(1) Si $(A, x) \in P$, escribiremos $A \longrightarrow x$.

Una producción $A \longrightarrow x$ de G se aplica a una palabra $u \in (V \cup \Sigma)^*$ en la que aparece A. El efecto de aplicar la producción $A \longrightarrow x$ a u es sustituir en la palabra u una aparición de A por x. Por tanto, las producciones de una gramática se utilizan como reglas de reescritura.

(2) Para simplificar la notación, si $A \longrightarrow \alpha_1$, $\ldots$, $A \longrightarrow \alpha_n$ son reglas de una gramática incontextual con una misma parte izquierda, escribiremos $A \longrightarrow \alpha_1 | \ldots | \alpha_n$.

Derivaciones en gramáticas incontextuales

(3) Si $u, v \in (V \cup \Sigma)^*$, decimos que v se deriva de u en un paso, en símbolos $u \Rightarrow_G v$, si obtenemos v a partir de u aplicando una producción de P.

(4) Si $u, v \in (V \cup \Sigma)^*$, decimos que v se deriva de u, en símbolos $u \Rightarrow_G^* v$, si obtenemos v a partir de u aplicando un número finito de veces las producciones de P.

Lenguaje asociado a una gramática incontextual

Si $G = (V, \Sigma, P, S)$ es una gramática incontextual y $A \in V$, definimos $L(A) = \{x \in \Sigma^* : A \Rightarrow_G^* x\}$. Definimos entonces el lenguaje de G como

$$L(G) = L(S) = \{x \in \Sigma^* : S \Rightarrow_G^* x\}.$$

Decimos que un lenguaje L es incontextual, si existe una gramática incontextual G tal que $L(G) = L$.

Ejemplo 1

Consideremos la gramática incontextual $G = (V, \Sigma, P, S)$ donde:

(1) $V = \{S, X\}$,

(2) $\Sigma = \{0, 1, 2\}$ y

(3) P consta de las siguientes producciones:

1. $S \longrightarrow 0X$,
2. $S \longrightarrow 2$,
3. $X \longrightarrow 0S$,
4. $X \longrightarrow 1$.

Tenemos entonces las siguientes derivaciones en G:

Ejemplo 1

$$S \Rightarrow^2 2,$$

$$S \Rightarrow^1 0X \Rightarrow^4 01,$$

$$S \Rightarrow^1 0X \Rightarrow^3 00S \Rightarrow^2 002,$$

$$S \Rightarrow^1 0X \Rightarrow^3 00S \Rightarrow^1 000X \Rightarrow^4 0001,$$

$$S \Rightarrow^1 0X \Rightarrow^3 00S \Rightarrow^1 000X \Rightarrow^3 0000S \Rightarrow^2 00002,$$

$$S \Rightarrow^1 0X \Rightarrow^3 00S \Rightarrow^1 000X \Rightarrow^3 0000S \Rightarrow^1 00000X \Rightarrow^4 000001,$$

$$\vdots$$

$$\vdots$$

Por tanto, $L(G) = \{0^n 2 : n \text{ es par }\} \cup \{0^n 1 : n \text{ es impar }\}$.

Ejemplo 2

Consideremos la gramática incontextual G dada por las siguientes producciones:

1. $S \longrightarrow (S)$
2. $S \longrightarrow SS$
3. $S \longrightarrow \lambda$

Se tiene que $L(G)$ es el lenguaje de las palabras de paréntesis balanceados, es decir, el lenguaje de las palabras $x \in \{(,)\}^*$ tales que a todo paréntesis abierto en x le corresponde un paréntesis cerrado.

Por ejemplo, la palabra $x = (()())()$ es generada por la siguiente derivación:

$S \Rightarrow^2 SS \Rightarrow^1 S(S) \Rightarrow^3 S() \Rightarrow^1 (S)() \Rightarrow^2 (SS)() \Rightarrow^1 ((S)S)()$
$\Rightarrow^3 (()S)() \Rightarrow^1 (()(S))() \Rightarrow^3 (()())()$.

Gramáticas regulares

A continuación, estudiamos las llamadas gramáticas regulares, que tienen interés porque es sabido que mediante este tipo de gramáticas se pueden representar los tipos de datos de los lenguajes de programación.

Sea $G = (V, \Sigma, P, S)$ una gramática incontextual. Se dice que G es una gramática regular, si toda producción de P es de la forma $A \longrightarrow xB$ o $A \longrightarrow x$ donde $A, B \in V$ y $x \in \Sigma^*$.

Ejemplo

Consideremos la gramática regular G dada por las siguientes producciones:

1. $S \longrightarrow 1S$
2. $S \longrightarrow 0T$
3. $T \longrightarrow 0T$
4. $T \longrightarrow 1S$
5. $T \longrightarrow \lambda$

Tenemos entonces que si $S \Rightarrow^* xS$ donde $x \in \{0,1\}^*$, entonces la palabra x acaba en 1. Y si $S \Rightarrow^* xT$ donde $x \in \{0,1\}^*$, entonces la palabra x acaba en 0. Entonces, como T es la única variable que se anula (por la producción 5), deducimos que $L(G) = \{x \in \{0,1\}^* : x \text{ acaba en } 0\}$.