Autómatas de pila

Horst H. von Brand
vonbrand@inf.utfsm.cl

Departamento de Informática
Universidad Técnica Federico Santa María

Contenido

Justificación

Vimos que los autómatas finitos están limitados por tener memoria finita (sus estados). Si queremos autómatas con capacidades mayores, debemos adicionarles memoria de alguna forma. Experimentalmente resulta que el paso siguiente correcto es adosar una pila (un *stack*). Al resultado se le llama *autómata de pila* o *de stack*, en inglés se le llama *Pushdown Automaton*, por lo de «empujar hacia abajo la pila» (si el tope queda a la misma altura) al agregar un elemento. Se abrevia PDA.

Definición de PDA

Definición

Un *autómata de pila* $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ consta de:

- Q: Conjunto finito de *estados*
- Σ: *Alfabeto de entrada*
- Γ: *Alfabeto de pila*
- δ: *Función de transición*, $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to 2^{Q \times \Gamma^*}$
  (acá los subconjuntos de $Q \times \Gamma^*$ son finitos)
- q_0: *Estado inicial*, $q_0 \in Q$
- Z_0: *Pila inicial*, $Z_0 \in \Gamma$
- F: Conjunto de *estados finales*, $F \subseteq Q$

Idea informal

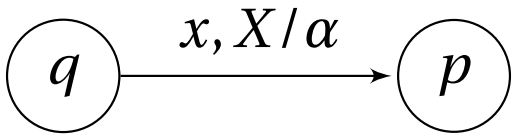
La idea del autómata $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ es que en una movida $(p, \alpha) \in \delta(q, x, A)$ (acá $p, q \in Q$, $A \in \Gamma$, $x \in \Sigma \cup \{\varepsilon\}$ y $\alpha \in \Gamma^*$) significa que si M está en el estado q, lee x de la entrada y tiene A en el tope de su pila, puede ir al estado p y reemplazar el tope de la pila por α (reemplazar A por ε corresponde a borrarlo, reemplazar por A deja la pila tal cual, reemplazar por $A\gamma$ agrega γ, entre otras posibilidades). El PDA inicia su procesamiento en el estado q_0 con únicamente Z_0 en la pila. Si hay alguna secuencia de movidas que llevan a M a consumir la palabra y terminar en un estado final, acepta.

Idea informal

Vemos que nuestro modelo es no-determinista de base (para una combinación dada de estado, tope de la pila y entrada puede tener múltiples movidas disponibles).

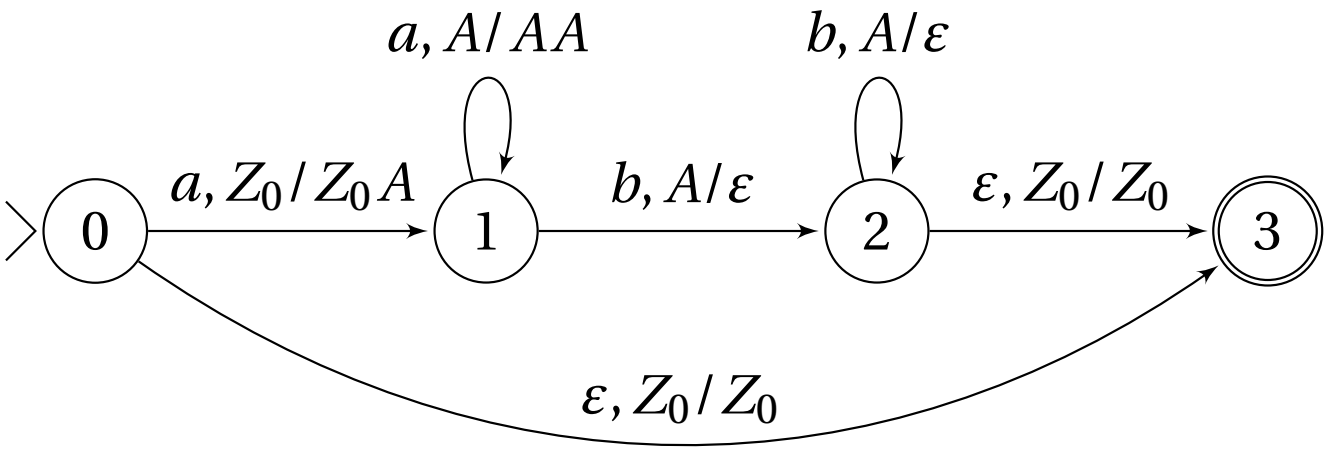
Notación gráfica

Reusamos la misma notación gráfica que adoptamos para NFAs. Solo que debemos agregar información adicional a las transiciones, el tope de la pila X y su reemplazo α:



Generalmente asumiremos Z_0 implícitamente como pila inicial.

Un PDA que acepta $\{a^n b^n : n \geq 0\}$



El PDA descrito tiene:

$$Q = \{0,1,2,3\} \qquad F = \{3\}$$
$$\Sigma = \{a,b\} \qquad \Gamma = \{Z_0, A\}$$

Por convención, la pila inicial es Z_0.

Un PDA que acepta $\{a^n b^n : n \geq 0\}$

Modus operandi

Vemos que hay dos alternativas de transición desde el estado inicial 0: la superior es leer a con Z_0 en la pila y reemplazar el tope de la pila por $Z_0 A$ (agrega A) yendo al estado 1; la inferior es leer ε (nada) con Z_0 en la pila, reemplazar el tope de la pila por Z_0 (queda igual) pasando al estado final 3.
En el estado 1, por cada a leída de la entrada agregamos A a la pila y volvemos a 1. Si leemos b, eliminamos una A de la pila y pasamos a 2.
En el estado 2, eliminamos una A de la pila por cada b leída y volvemos a 2. Si hemos consumido todas las A (destapamos Z_0 que sigue al fondo de la pila), podemos no leer nada de la entrada, dejar la pila tal cual e ir al estado final 3.

Un PDA que acepta $\{a^n b^n : n \geq 0\}$

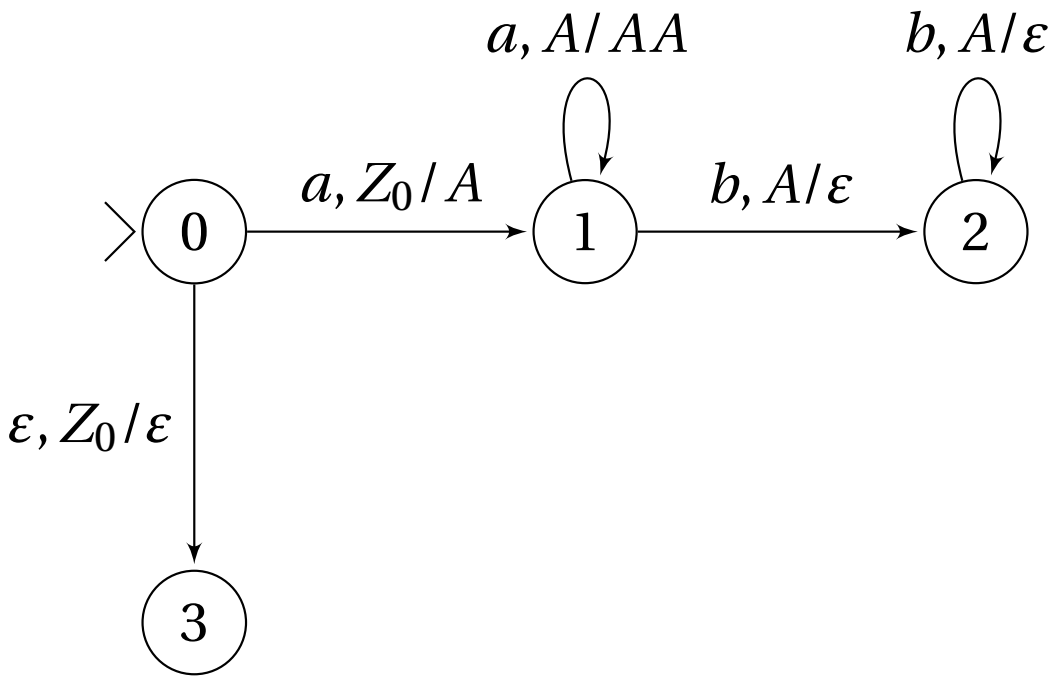
Modus operandi

Vemos que puede aceptar ε tomando el camino inferior.
En el camino superior por cada a leída agrega una A a la pila, luego por cada b leída elimina una A de la pila. Solo si el número de a y b coincide estaremos al final de la palabra con Z_0 en el tope de la pila, pudiendo transitar al estado final para aceptar.

Otra forma de aceptar $\{a^n b^n : n \geq 0\}$

Lo anterior se modela según un NFA con memoria, aceptamos si al final estamos en un estado final. También resulta natural aceptar si al final terminamos con la pila vacía (haciendo caso omiso del estado en que termina el autómata). Esto es muy natural en este ejemplo: acumulamos A al leer a, descontamos A al leer b; si los números coinciden, la cuenta final es 0 (pila vacía).

Otra forma de aceptar $\{a^n b^n : n \geq 0\}$

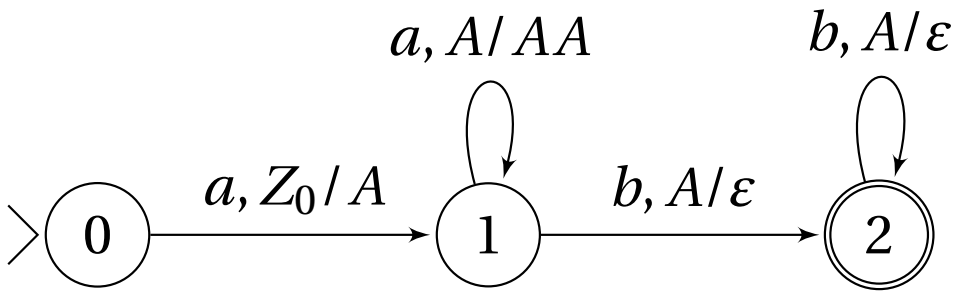


Otra forma de aceptar $\{a^n b^n : n \geq 0\}$

Modus operandi

En el estado 0, tenemos la opción de vaciar la pila e ir al estado 3 sin leer nada. Si no queda nada por leer, aceptamos. O sea, acepta ε.
La otra opción desde el estado 0 es reemplazar Z_0 por A leyendo a.
En el estado 1 acumulamos A igual que antes, vamos a 2 al leer b, eliminando una A.
En el estado 2 descontamos una A por cada b leída.
Solo si el número de a y de b coincide queda vacía la pila al final de la palabra.

Acepta lenguajes distintos



Para este autómata es $\mathscr{L}(M) = \{a^m b^n : m \geq n \geq 1\}$, mientras $\mathscr{N}(M) = \{a^n b^n : n \geq 1\}$.

Descripción instantánea

Al igual que lo hicimos para NFAs, la idea es contar con una descripción compacta del estado del PDA para describir su cómputo. Debemos agregar el contenido de la pila.

Definición

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ un PDA, donde sin pérdida de generalidad suponemos $Q \cap \Sigma = \varnothing$. Una *descripción instantánea de* M (ID, por el inglés *instantaneous description*) es $(\alpha q \beta, \gamma)$, con $\alpha, \beta \in \Sigma^*$, $q \in Q$ y $\gamma \in \Gamma^*$.

La idea es que M está en el estado q, leyó α y le queda por leer β, actualmente su pila contiene γ.

Relación de transición

Definición

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ un PDA. La *relación de transición de* M entre descripciones instantáneas de M, anotada $\vdash_M$, se define por:

$$(\alpha q a \beta, \gamma A) \vdash_M (\alpha a p \beta, \gamma \sigma) \qquad (p, \sigma) \in \delta(q, a, A)$$

$$(\alpha q \beta, \gamma A) \vdash_M (\alpha p \beta, \gamma \sigma) \qquad (p, \sigma) \in \delta(q, \varepsilon, A)$$

Lenguaje aceptado

Definición

Sea $M = (Q, \Sigma, \delta, q_0, F)$ un PDA, $\vdash_M$ su relación de transición. El *lenguaje aceptado por M (por estado final)* es:

$$\mathscr{L}(M) = \{\alpha \in \Sigma^* : (q_0\alpha, Z_0) \vdash_M^* (\alpha q_f, \gamma) \wedge q_f \in F\}$$

El *lenguaje aceptado por M por pila vacía* es:

$$\mathscr{N}(M) = \{\alpha \in \Sigma^* : (q_0\alpha, Z_0) \vdash_M^* (\alpha q, \varepsilon)\}$$

Lenguaje aceptado

Para aceptar por pila vacía el conjunto de estados finales no importa. Por convención, al diseñar un PDA que acepta por pila vacía se da $\varnothing$ como conjunto de estados finales.

En general, $\mathscr{L}(M) \neq \mathscr{N}(M)$.

Relación entre los conjuntos de lenguajes

Esto inmediatamente plantea la pregunta de la relación entre los conjuntos de lenguajes así definidos.

Dado un PDA M mostraremos cómo construir un PDA que acepta el lenguaje $\mathscr{L}(M)$ (aceptado por M por estado final) por pila vacía y cómo construir un PDA que acepta el lenguaje $\mathscr{N}(M)$ (aceptado por M por pila vacía) por estado final.

Podemos elegir cómo aceptar al diseñar un PDA, según lo que resulte más cómodo.

Estado final a pila vacía

La idea básica es añadir al PDA que acepta por estado final movidas que lo hagan vaciar la pila si está en un estado final. Esto presenta un problema: si el PDA termina de procesar la palabra con la pila vacía, pero *no* termina en un estado final, aceptaría erróneamente. Para evitar lo anterior, agregamos un nuevo símbolo inicial a la pila debajo de la pila inicial de M, luego simulamos el funcionamiento de M. Movidas adicionales del nuevo PDA, disponibles para un nuevo estado alcanzable solo desde estados finales, podrán remover el nuevo símbolo.

Estado final a pila vacía

Teorema

Sea M un PDA. Podemos construir un PDA M' tal que $\mathscr{N}(M') = \mathscr{L}(M)$.

Estado final a pila vacía

Demostración

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$. Sea $Q' = Q \cup \{q'_0, q'_f\}$, donde q'_0 y q'_f son nuevos estados; $\Gamma' = \Gamma \cup \{Z'_0\}$, con Z'_0 un nuevo símbolo. Pondremos Z'_0 bajo la pila de M, simulamos M, y luego limpieza del resultado.

Estado final a pila vacía

Demostración

Definimos la función de transición δ' agregando las siguientes a δ:

$\delta'(q_0', \varepsilon, Z_0') = \{(q_0, Z_0' Z_0)\}$ $\quad$ Z_0' bajo de la pila de M, va a q_0

Para todo estado $q_f \in F$ y todo $X \in \Gamma \cup \{Z_0'\}$ agregamos:

$\delta'(q_f, \varepsilon, X) = \{(q_f', \varepsilon)\}$ $\quad$ q_f' se encarga de vaciar la pila

Para todo $X \in \Gamma \cup \{Z_0'\}$ agregamos:

$\delta'(q_f', \varepsilon, X) = \{(q_f', \varepsilon)\}$ $\quad$ Vaciar la pila

El PDA $M' = (Q', \Sigma, \Gamma', \delta', q_0', Z_0', \varnothing)$ hace lo prometido. □

Pila vacía a estado final

La idea básica es añadir al PDA que acepta por pila vacía movidas que le permitan ir a un estado final cuando su pila está vacía. Pero una vez vacía la pila el PDA se detiene, no puede seguir moviéndose (no hay símbolo del tope de la pila para una transición). Para manejar lo anterior, nuevamente agregamos un nuevo símbolo inicial a la pila debajo de la pila inicial de M, luego simulamos el funcionamiento de M. Solo movidas adicionales del nuevo PDA podrán remover el nuevo símbolo. Se ejecutan movidas que vacían la pila leyendo ε en un nuevo estado, al que se entra solo desde un estado final de M leyendo ε. Si tales movidas ocurren al final de la entrada, acepta por pila vacía.

Pila vacía a estado final

Teorema

Sea M un PDA. Podemos construir un PDA M' tal que $\mathscr{L}(M') = \mathscr{N}(M)$.

Pila vacía a estado final

Demostración

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$. Sea $Q' = Q \cup \{q'_0, q'_f\}$, donde q'_0 y q'_f son nuevos estados; $\Gamma' = \Gamma \cup \{Z'_0\}$, con Z'_0 un nuevo símbolo.
Definimos la función de transición δ' agregando las siguientes a δ:

$$\delta'(q'_0, \varepsilon, Z'_0) = \{(q_0, Z'_0 Z_0)\} \qquad Z'_0 \text{ bajo de la pila de } M, \text{ va a } q_0$$

Para todo estado $q \in Q$ agregamos:

$$\delta'(q, \varepsilon, Z'_0) = \{(q'_f, \varepsilon)\} \qquad \text{Va a } q'_f \text{ si la pila de } M \text{ está vacía}$$

El PDA $M' = (Q', \Sigma, \Gamma', \delta', q'_0, Z'_0, \{q'_f\})$ hace lo prometido. □

PDA determinista

Un PDA será determinista si en cada situación tiene a lo más una única movida posible. Esto requiere que $|\delta(q, x, X)| \leq 1$ siempre. Pero no es suficiente: el PDA puede aún elegir entre leer ε o $a \in \Sigma$. Hay que eliminar esta posibilidad.

PDA determinista

Definición

El PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ se dice *determinista* (abreviado DPDA, por *Deterministic PDA*) si cumple:

$|\delta(q, x, X)| \leq 1$ A lo más una opción

Para todo q, X tal que $\delta(q, \varepsilon, X) \neq \varnothing$ y todo $a \in \Sigma$:

$\delta(q, a, X) = \varnothing$ Solo una de movida con ε o un símbolo

Lenguaje de PDA determinista

Lamentablemente:

Teorema

Hay lenguajes aceptados por PDAs que no son aceptados por ningún DPDA.

Lenguaje de PDA determinista

Demostración

La siguiente demostración es un tanto informal. Puede completarse a una demostración rigurosa, pero se complica muy substancialmente sin aportar a entender el fenómeno.
La idea es diseñar un lenguaje que fuerce a un PDA que lo acepta a hacer uso del no-determinismo, obligando a «adivinar».
Consideremos el lenguaje:

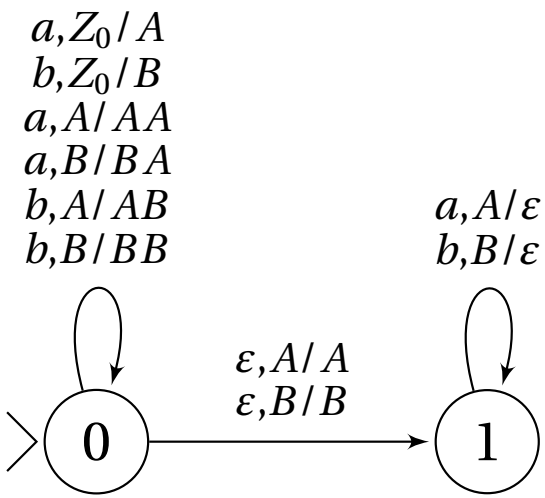
$$L = \{\omega\omega^R : \omega \in \{a, b\}^+\}$$

Son *palíndromos* (de largo par), palabras que se leen igual de atrás adelante.

Lenguaje de PDA determinista

Demostración

El PDA ilustrado reconoce L por pila vacía:



Lenguaje de PDA determinista

Demostración

En el estado 0 lee la primera mitad de la palabra. Por cada a pone una A en la pila, por cada b una B.
Solo $\delta(0, a, Z_0)$ y $\delta(0, b, Z_0)$ permiten eliminar Z_0 de la pila, lee al menos un símbolo.
Al ir de 0 a 1 está «adivinando» que está al medio de la palabra.
En el estado 1 verifica que la segunda mitad es el reverso de la primera.

Lenguaje de PDA determinista

Demostración

Si el DPDA M reconoce L por pila vacía, debe terminar con la pila vacía al leer aa. Pero entonces se detiene al leer los primeros dos símbolos de $aaaa$, y no acepta.

La demostración formal parte de que si L es aceptado por un DPDA M, tiene la propiedad de prefijo: si $\alpha \in L$, ningún prefijo propio de α pertenece a L.

Interés por DPDA

Un autómata determinista es interesante, es mucho más fácil de simular mediante un programa.
En aplicaciones como lenguajes de programación, autómatas deterministas significan que el lector humano puede procesar la palabra sin ambigüedades, hay menos alternativas válidas.

Resumen

- Definimos formalmente nuestro modelo de autómata. El modelo es no-determinista de base.
- Hay dos definiciones naturales del lenguaje aceptado por el PDA M: por estado final ($\mathscr{L}(M)$) y por pila vacía ($\mathscr{N}(M)$). Generalmente $\mathscr{L}(M) \neq \mathscr{N}(M)$.
- Definimos PDAs deterministas (DPDAs). Resulta que PDAs deterministas reconocen solo un subconjunto de los lenguajes reconocidos por PDAs.
- Vimos que los lenguajes aceptados por estado final por algún PDA son exactamente los lenguajes aceptados por algún PDA por pila vacía.