Autómatas de Pila
(*pushdown automata*)

Alan Reyes-Figueroa
Teoría de la Computación **(Aula 17) 09.octubre.2023**

Definición

Movimientos de un PDA

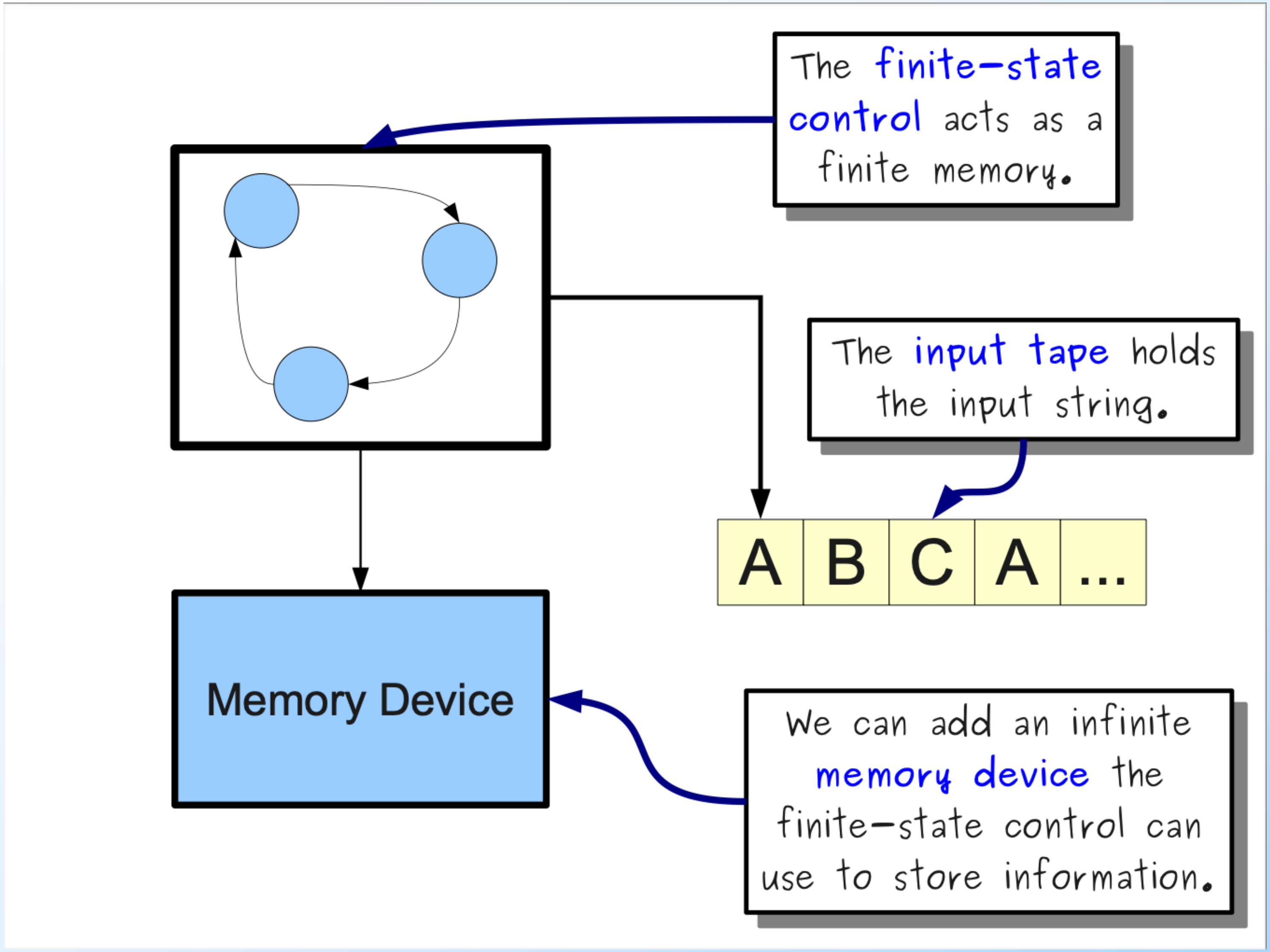
Lenguaje de un PDA

PDAs Determinísticos

Gramáticas Libres

- Hemos visto que los lenguajes libres de context (CFL) son aquellos generados por gramáticas CFG.
- ¿Existe alguna forma de construer autómatas que reconozcan un lenguaje libre de contexto?
- **Respuesta:** Sí, pero debemo añadir un dispositivo de memoria (ilimitada).

Gramáticas Libres



Añadiendo Memoria

- Podemos "aumentar" un automata al añadir un dispositivo de memoria:
  - almacenar información extra
  - realizar las transiciones (estados + memoria)
  - ejecutar comandos a memoria

<u>Stack-based Memory</u>:

- Sólo el top del stack es visible (en todo momento).
- Símbolos nuevos pueden agregarse al stack ("push"), reemplazando el símbolo en el top del stack.
- El símbolo en el top del stack puede eliminarse ("pop"), revelando el símbolo debajo de él.

Autómatas de Pila

- Un autómata de pila (*pushdown automata* o PDA) es el equivalente a una gramática libre de contexto CFG.
- Sólo los autómatas de pila no deterministas definen todos los lenguajes libres del contexto.
- La versión determinística modela *parsers*.
  - La mayoría de lenguajes de programación son definidos por un PDA determinista.

Intuición: Autómatas

- Un PDA se puede pensar como un ϵ-NFA con la propiedad adicional de que puede manipular una pila (*stack*).
- Los movimientos de un PDA se determinan por:
  1. El estado actual (de su "NFA"),
  2. El símbolo de entrada actual (ó ϵ), y
  3. El símbolo actual en el top de la pila.

Intuición: Autómatas

- Siendo no deterministas, los autómatas de pila pueden "elegir" la siguiente movida.
- En cada elección, el PDA puede:
  1. Cambiar de estado, y, además
  2. Reemplazar el símbolo top de la pila por una secuencia de cero o más símbolos.
     - Cero símbolos = "pop".
     - Más símbols = secuencia de varios "push".

Formalismo de un PDA

- Un PDA se describe mediante una estructura $A = (Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$, donde:
  1. Un conjunto finito de *estados* (Q).
  2. Un *alfabeto de entrada* (Σ).
  3. Un *alfabeto de pila* (Γ).
  4. Una *función de transición* (δ).
  5. Un *estado inicial* ($q_0 \in Q$).
  6. Un *símbolo inicial* ($Z_0 \in \Gamma$).
  7. Un conjunto de *estados finales* ($F \subseteq Q$).

Convenciones

- a, b, … son símbolos de entrada.
  - Tomar en cuenta que también admitimos como válido el símbolo de entrada ε.
- …, X, Y, Z son símbolos de pila.
- …, w, x, y, z denotan cadenas de símbolos de entradas.
- α, β, … denotan cadenas de símbolos de pila.

La Función de Transición

- Ahora δ: $Q \times \Sigma \times \Gamma \rightarrow Q \times \Gamma^*$.
  La función tomas tres argumentos:
  1. Un estado $q \in Q$.
  2. Un símbolo de entrada a, el cual puede ser un símbolo de Σ, o ser ε.
  3. Un símbolo de pila $Z \in \Gamma$.
- $\delta(q, a, Z)$ denota el conjunto de cero o más acciones de la forma (p, α).
  - $p \in Q$; $\alpha \in \Gamma^*$ cadena de símbolos pila.

Acciones de un PDA

- Si $\delta(q, a, Z)$ contiene (p, α) entre sus acciones, entonces una de las cosas que el PDA puede hacer en el estado q, con símbolo de entrada a, y Z en el top de la pila es:
  1. Cambiar del estado q al estado p.
  2. Remover *a* del input (*a* podría ser ε).
  3. Reemplazar Z en el top de la pila por la cadena α.

Ejemplo: PDA

- Vamos a diseñar un PDA para aceptar las cadenas $\{0^n1^n: n \geq 1\}$.
- Estados:
  - q = estado inicial. Estaremos en el estado q si sólo hemos visto 0s hasta ahora.
  - p = alcanzamos este estado si hemos visto al menos un 1, y procedemos si los inputs sólo son 1s.
  - f = estado final, de aceptación.

Ejemplo: PDA

- Símbolos de entrada (inputs):
  - {0, 1}.
- Símbolos de pila (stack):
  - Z_0 = símbolo inicial. También marca el fondo de la pila, de forma que podemos contar el mismo número de 1s que de 0s.
  - X = marcador. Se usa para contar el número de 0s en la cadena de entrada.

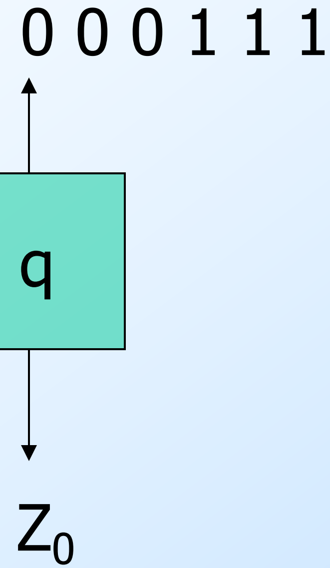
Ejemplo: PDA

- Las transiciones:
  - $\delta(q, 0, Z_0) = \{(q, XZ_0)\}$.
  - $\delta(q, 0, X) = \{(q, XX)\}$.

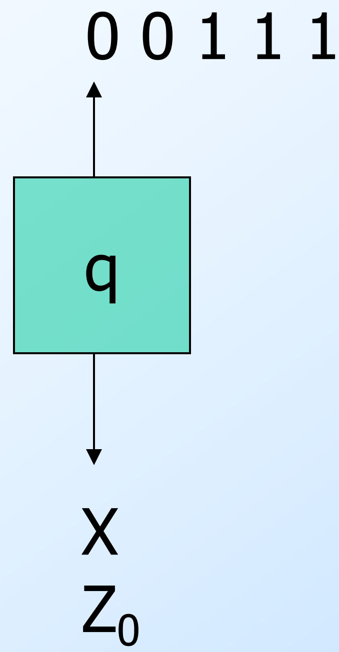
    Estas reglas hacen que se agregue una X a la pila, cada vez que se lee un 0 en la entrada.
  - $\delta(q, 1, X) = \{(p, \epsilon)\}$.

    Al leer un 1, vamos al estado p, y se hace pop de una X.
  - $\delta(p, 1, X) = \{(p, \epsilon)\}$. Pop de una X por cada 1.
  - $\delta(p, \epsilon, Z_0) = \{(f, Z_0)\}$. Aceptar al final.

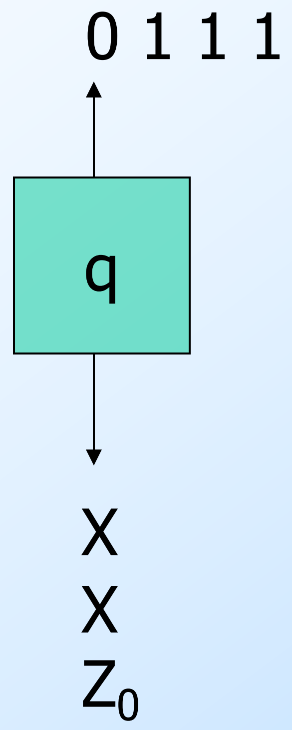
Accciones en el PDA Ejemplo



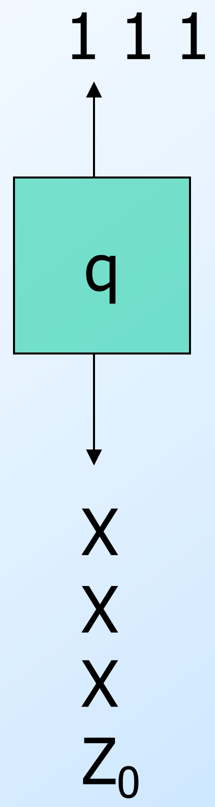
Accciones en el PDA Ejemplo



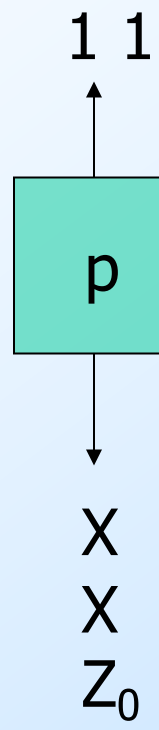
Accciones en el PDA Ejemplo



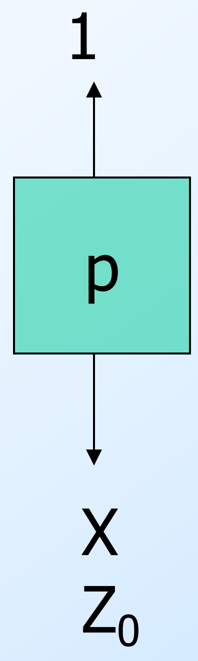
Accciones en el PDA Ejemplo



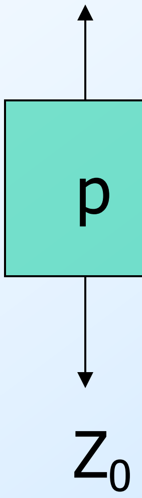
Accciones en el PDA Ejemplo



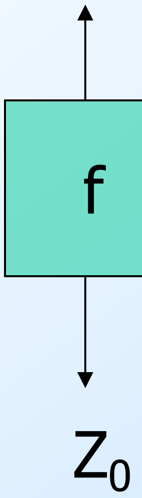
Accciones en el PDA Ejemplo



Accciones en el PDA Ejemplo



Accciones en el PDA Ejemplo



Descripciones Instantáneas

- Formalizamos las figuras anteriores bajo el concepto de *descripción instantánea* (ID).
- Una ID es una tripla (q, w, α), donde:
  1. q es el estado actual.
  2. w es el input remanente.
  3. α es el contenido de la pila (top to bottom).

Relación "Goes-To"

- Denotamos por

$$I \vdash J$$

cuando la descripción I puede convertirse en la descripción J en un movimiento.

- Formalmente,

$$(q, aw, X\alpha) \vdash (p, w, \beta\alpha)$$

para w, α, si $\delta(q, a, X)$ contiene la acción (p, β).

- Extendemos $\vdash$ a $\vdash^*$, "cero o más movidas" por:
  - Base: $I \vdash^* I$.
  - Inducción: Si $I \vdash^* J$ y $J \vdash K$, entonces $I \vdash^* K$.

Ejemplo: "Goes-To"

- Usamos el PDA del ejemplo anterior. Tenemos:

$$
\begin{aligned}
(q, 000111, Z_0) &\vdash (q, 00111, XZ_0)\\
&\vdash (q, 0111, XXZ_0)\\
&\vdash (q, 111, XXXZ_0)\\
&\vdash (p, 11, XXZ_0)\\
&\vdash (p, 1, XZ_0)\\
&\vdash (p, \epsilon, Z_0)\\
&\vdash (f, \epsilon, Z_0)
\end{aligned}
$$

- Así, $(q, 000111, Z_0) \vdash^* (f, \epsilon, Z_0)$.
- Ejercicio: ¿Cómo quedaría si el input fuese 0001111?

Respuesta

Legal ya que un PDA puede usar ϵ input, aún si el input no cambia.

- $(q, 0001111, Z_0) \vdash (q, 001111, XZ_0)$
  $\vdash (q, 01111, XXZ_0)$
  $\vdash (q, 1111, XXXZ_0)$
  $\vdash (p, 111, XXZ_0)$
  $\vdash (p, 11, XZ_0)$
  $\vdash (p, 1, Z_0)$
  $\vdash (f, 1, Z_0)$

- Observe que la última descripción no se mueve.
- 0001111 no es aceptada, ya que el input no es completamente consumido.

Respuesta

- $(q, 000011, Z_0) \vdash (q, 00011, XZ_0)$
  $\vdash (q, 0011, XXZ_0)$
  $\vdash (q, 011, XXXZ_0)$
  $\vdash (q, 11, XXXXZ_0)$
  $\vdash (p, 1, XXXZ_0)$
  $\vdash (p, \epsilon, XXZ_0)$
  ya no se puede continuar.

- Observe que la última descripción no se mueve.
- 000011 no es aceptada, ya que la memoria tiene información (aparte de Z_0).

Autómatas finitos y PDAs

- Hemos representado los movimientos de un autómata finito mediante una función de transición extendida δ, que no menciona al input remanente.
- Podemos representar con una notación similar a los PDAs, en donde el estado del autómata se reemplaza por una combinación estado-stack, como en los diagramas anteriores.

Autómatas finitos y PDAs

- Similarmente, podemos representar un autómata finito con la notación de las descripciones ID.
  - Sólo hay que quitar el componente stack.
- ¿Por qué la diferencia? (Mi teoría):
- Los FA tienden a modelos protocolos, con inputs indefinidamente largos.
- Los PDA modelan *parsers* (analizadores sintácticos) que procesan un programa.

Lenguaje de un PDA

- La manera usual de definir el lenguaje de un autómata de pila es mediante *estados finales*.
- Si P es un autómata de pila, entonces L(P) es el conjunto de cadenas w ϵ Σ, tales que

$$(q_0, w, Z_0) \vdash^* (f, \varepsilon, \alpha),$$

para algún estado final f ε F, y cualquier α.

Lenguaje de un PDA

- Otra manera de definir el lenguaje generado por un PDA es mediante el concepto de *stack vacío*.
- Si P es un autómata de pila, entonces N(P) es el conjunto de cadenas w ϵ Σ, tales que

$$(q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon)$$

para cualquier estado q.

Equivalencia $L(P) = N(P)$

1. Si $L = L(P)$, entonces existe otro autómata de pila P' tal que $L = N(P')$.

2. Si $L = N(P)$, entonces existe otro autómata de pila P'' tal que $L = L(P'')$.

Prueba: L(P) → N(P’) (Idea)

- P’ simulará a P.
- Si P acepta, P’ tendrá su stack vacío.
- P’ debe evitar vaciar su stack accidentalmente, así que se usa un símbolo especial (bottom-marker) para detectar el caso en el que P vacía su pila sin aceptar.

Prueba: $L(P) \rightarrow N(P')$

- P' posee todos los estados, símbolos y movidas de P, más:
  1. Un símbolo stack X_0, usado para guardar el final del stack de vacíos accidentales.
  2. Un nuevo estado inicial s, y
     un nuevo estado "erase" e.
  3. $\delta(s, \varepsilon, X_0) = \{(q_0, Z_0X_0)\}$. (Inicia P)
  4. $\delta(f, \varepsilon, X) = \delta(e, \varepsilon, X) = \{(e, \varepsilon)\}$ para cualquier estado final f de P, y para todo símbolo stack X.

Prueba: N(P) $\rightarrow$ L(P”) (Idea)

- P” simulará a P.
- P” posee un símbolo especial (bottom-marker) para detectar aquellos casos donde P vacía su stack.
- En ese caso, P” acepta.

Prueba: $N(P) \rightarrow L(P'')$

- P'' posee todos los estados, símbolos y movidas de P, más:
  1. Un símbolo stack X_0, usado para guardar el final del stack.
  2. Un nuevos estado inicial s, y un nuevo estado final f.
  3. $\delta(s, \varepsilon, X_0) = \{(q_0, Z_0X_0)\}$. (Inicia P)
  4. $\delta(q, \varepsilon, X_0) = \{(f, \varepsilon)\}$ para todo $q \in P$.

PDAs Deterministas

- Para un PDA ser determinista, debe existir a lo sumo una elección de movida en cada combinación (q,a,X), (estado q, input *a*, símbolo stack X).
- Además, no debe haber una elección entre un input de Σ, y la cadena ε.
  - Formalmente, δ(q, a, X) y δ(q, ϵ, X) no pueden ser ambos no-vacíos.