Desde apiladores a gramáticas

Clase 27

IIC2223 / IIC2224

Prof. Cristian Riveros

Outline

Desde CFG a PDA (clase anterior)

Desde PDA a CFG

Outline

Desde CFG a PDA (clase anterior)

Desde PDA a CFG

Desde CFG a un PDA

Teorema

Para toda gramática libre de contexto $\mathcal{G}$,
existe un **autómata apilador alternativo** $\mathcal{D}$:

$$\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{D})$$

¿cómo **construimos** $\mathcal{D}$ desde $\mathcal{G}$?

CFG → PDA: Construcción

Sea $\mathcal{G} = (V, \Sigma, P, S)$ una CFG.

Construimos un PDA alternativo $\mathcal{D}$ que acepta $\mathcal{L}(\mathcal{G})$:

$$\mathcal{D} \;=\; \big(V \cup \Sigma \cup \{q_0, q_f\}, \Sigma, \Delta, q_0, \{q_f\}\big)$$

La relación de transición Δ se define como:

$$\begin{array}{rcll} \Delta & = & \{\,(q_0, \epsilon, S \cdot q_f)\,\} & \cup \\ & & \{\,(X, \epsilon, \gamma) \mid X \to \gamma \in P\,\} & \cup \quad (\textbf{Expandir}) \\ & & \{\,(a, a, \epsilon) \mid a \in \Sigma\,\} & \quad (\textbf{Reducir}) \end{array}$$

¿qué esta haciendo el autómata apilador $\mathcal{D}$?

CFG → PDA: Demostración

Por demostrar:

$$\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{D})$$

Dos direcciones:

1. $\mathcal{L}(\mathcal{G}) \subseteq \mathcal{L}(\mathcal{D})$ ✓
2. $\mathcal{L}(\mathcal{D}) \subseteq \mathcal{L}(\mathcal{G})$

CFG → PDA: Demostración $\mathcal{L}(\mathcal{D}) \subseteq \mathcal{L}(\mathcal{G})$

Para cada $w \in \mathcal{L}(\mathcal{D})$ debemos encontrar
un árbol de derivacíon $\mathcal{G}$ para w.

¿cómo encontramos un árbol de derivación para w?

Idea

Si tenemos una ejecución de $\mathcal{D}$ sobre w de la forma:

$$(X \cdot q_f, w) \ \vdash_{\mathcal{D}}^{*} \ (q_f, \epsilon)$$

entonces $X \overset{\star}{\underset{\mathcal{G}}{\Rightarrow}} w$

Inducción en la cantidad de pasos de la ejecución.

CFG $\rightarrow$ PDA: Demostración $\mathcal{L}(\mathcal{D}) \subseteq \mathcal{L}(\mathcal{G})$

Hipótesis de inducción

Para toda ejecución de $\mathcal{D}$ sobre w de largo k de la forma:

$$(X \cdot q_f, w) = (\gamma_0, w_0) \vdash_{\mathcal{D}} (\gamma_1, w_1) \vdash_{\mathcal{D}} \cdots \vdash_{\mathcal{D}} (\gamma_k, w_k) = (q_f, \epsilon)$$

entonces $X \overset{\star}{\underset{\mathcal{G}}{\Rightarrow}} w$.

Ejercicio: terminar la demostración.

Outline

Desde CFG a PDA (clase anterior)

Desde PDA a CFG

Recordatorio: Autómatas apiladores

Definición

Un **autómata apilador** (PushDown Automata, PDA) es una estructura:

$$\mathcal{P} \quad = \quad (Q, \Sigma, \Gamma, \Delta, q_0, \perp, F)$$

- Q es un conjunto finito de **estados**.
- Σ es el alfabeto de **input**.
- $q_0 \in Q$ es el estado **inicial**.
- F es el conjunto de estados **finales**.

$$+$$

- Γ es el alfabeto de **stack**.
- $\perp \in \Gamma$ es el símbolo **inicial de stack**.
- $\Delta \subseteq \big(Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma\big) \times \big(Q \times \Gamma^*\big)$ es una relación finita de transición.

Recordatorio: Autómatas apiladores

Definición

Un **autómata apilador** (PushDown Automata, PDA) es una estructura:

$$\mathcal{P} \quad = \quad (Q, \Sigma, \Gamma, \Delta, q_0, \perp, F)$$

Intuitivamente, la transición:

$$\big((p, a, A), \, (q, B_1 B_2 \cdots B_k) \big) \in \Delta$$

si el autómata apilador está:

- en el estado **p**,
- leyendo **a**, y
- en el tope del stack hay una **A**

entonces:

- cambia al estado **q**, y
- modifico el tope **A** por $\mathbf{B_1 B_2 \cdots B_k}$.

Recordatorio: Ejecución de un autómata apilador

Sea $\mathcal{P} = (Q, \Sigma, \Gamma, \Delta, q_0, \bot, F)$ un autómata apilador.

Definición

Se define la relación $\vdash_{\mathcal{P}}$ de **siguiente-paso** entre configuraciones de $\mathcal{P}$:

$$(q_1 \cdot \gamma_1, w_1) \ \vdash_{\mathcal{P}} \ (q_2 \cdot \gamma_2, w_2)$$

si, y solo si, existe una transición $(q_1, a, A, q_2, \alpha) \in \Delta$ y $\gamma \in \Gamma^*$ tal que:

- $w_1 = a \cdot w_2$
- $\gamma_1 = A \cdot \gamma$
- $\gamma_2 = \alpha \cdot \gamma$.

Se define $\vdash_{\mathcal{P}}^*$ como la clausura **refleja** y **transitiva** de $\vdash_{\mathcal{P}}$.

$(q_1\gamma_1, w_1) \vdash_{\mathcal{P}}^* (q_2\gamma_2, w_2)$ si uno puede ir de $(q_1\gamma_1, w_1)$ a $(q_2\gamma_2, w_2)$ en **0 o más pasos**.

Recordatorio: Lenguajes de un autómata apilador

Sea $\mathcal{P} = (Q, \Sigma, \Gamma, \Delta, q_0, \perp, F)$ un autómata apilador y $w \in \Sigma^*$.

Definiciones

- $\mathcal{P}$ **acepta** w si, y solo si, $(q_0\perp, w) \vdash_{\mathcal{P}}^{*} (q_f, \epsilon)$ para algún $q_f \in F$.
- El **lenguaje** aceptado por $\mathcal{P}$ se define como:

$$\mathcal{L}(\mathcal{P}) = \{w \in \Sigma^* \mid \mathcal{P} \text{ acepta } w\}$$

Desde PDA a un CFG

Teorema

Para todo autómata apilador $\mathcal{P}$, existe una gramática libre de contexto $\mathcal{G}$:

$$\mathcal{L}(\mathcal{P}) = \mathcal{L}(\mathcal{G})$$

Estrategia de la demostración

Sea $\mathcal{P} = (Q, \Sigma, \Gamma, \Delta, q_0, \perp, F)$ un PDA (normal).

1. Convertir $\mathcal{P}$ a un PDA $\mathcal{P}'$ con **UN solo estado**.
2. Convertir $\mathcal{P}'$ a una gramática libre de contexto $\mathcal{G}$.

¿cómo hacemos cada paso?

PDA $\rightarrow$ CFG: Demostración

Paso 2: Convertir $\mathcal{P}'$ a una CFG $\mathcal{G}$

Sea $\mathcal{P}' = \big(\{q\}, \Sigma, \Gamma, \Delta, q, \perp, \{q\}\big)$ con **UN solo estado**.

Construimos la gramática:

$$\mathcal{G} \;=\; (V, \Sigma, P, \perp)$$

- $V = \Gamma$.
- Si $qA \stackrel{\epsilon}{\rightarrow} q\alpha \in \Delta$ entonces: $A \rightarrow \alpha \in P$
- Si $qA \stackrel{a}{\rightarrow} q\alpha \in \Delta$ entonces: $A \rightarrow a\,\alpha \in P$

Demostración: ejercicio.

PDA $\rightarrow$ CFG: Demostración

Paso 1: Convertir $\mathcal{P}$ a un PDA $\mathcal{P}'$ con UN solo estado

Sea $\mathcal{P} = (Q, \Sigma, \Gamma, \Delta, q_0, \perp, F)$ un PDA.

¿cómo guardamos la información de los estados en el stack?

Pregunta principal

"Si el PDA esta en el estado p y en el tope del stack hay una A entonces, ¿a cuál estado llegaré al remover A del stack?"

Solución

Podemos **adivinar** (no-determinismo) el estado que vamos a llegar cuando removamos A del stack.

PDA $\rightarrow$ CFG: Demostración

Paso 1: Convertir $\mathcal{P}$ a un PDA $\mathcal{P}'$ con UN solo estado

Sea $\mathcal{P} = (Q, \Sigma, \Gamma, \Delta, q_0, \perp, F)$ un PDA.

Sin perdida de generalidad podemos asumir que:

1. Todas las transiciones son de la forma:

$$qA \xrightarrow{c} pB_1B_2 \quad \text{o} \quad qA \xrightarrow{c} p\epsilon$$

   con $c \in \left(\Sigma \cup \{\epsilon\}\right)$. ¿por qué?

2. Existe $q_f \in Q$ tal que si $w \in \mathcal{L}(\mathcal{P})$ entonces:

$$(q_0\perp, w) \vdash_{\mathcal{D}}^{*} (q_f, \epsilon)$$

   ¿por qué?

Siempre llegamos al **mismo estado** q_f.

PDA $\rightarrow$ CFG: Demostración

Paso 1: Convertir $\mathcal{P}$ a un PDA $\mathcal{P}'$ con UN solo estado

Sea $\mathcal{P} = (Q, \Sigma, \Gamma, \Delta, q_0, \perp, F)$ un PDA.

Construimos el autómata apilador $\mathcal{P}'$ con **un solo estado**:

$$\mathcal{P}' = (\{q\}, \Sigma, \Gamma', \Delta', \{q\}, \perp', \{q\})$$

- $\Gamma' = Q \times \Gamma \times Q$.

  "$(p, A, q) \in \Gamma'$ si desde p leyendo A en el tope de stack llegamos a q al hacer pop de A"

- $\perp' = (q_0, \perp, q_f)$

  "El autómata parte en q_0 y al hacer pop de $\perp$ llegará a q_f"

PDA $\rightarrow$ CFG: Demostración

Paso 1: Convertir $\mathcal{P}$ a un PDA $\mathcal{P}'$ con UN solo estado

Sea $\mathcal{P} = (Q, \Sigma, \Gamma, \Delta, q_0, \perp, F)$ un PDA.

Construimos el autómata apilador $\mathcal{P}'$ con **un solo estado**:

$$\mathcal{P}' = (\{q\}, \Sigma, \Gamma', \Delta', \{q\}, \perp', \{q\})$$

- Si $pA \stackrel{c}{\rightarrow} p'B_1B_2 \in \Delta$ con $c \in \left(\Sigma \cup \{\epsilon\}\right)$, entonces **para todo** $p_1, p_2 \in Q$:

$$q\,(p, A, p_2) \stackrel{c}{\rightarrow} q\,(p', B_1, p_1)(p_1, B_2, p_2) \in \Delta'$$

- Si $pA \stackrel{c}{\rightarrow} p' \in \Delta$ con $c \in \left(\Sigma \cup \{\epsilon\}\right)$, entonces:

$$q\,(p, A, p') \stackrel{c}{\rightarrow} q \in \Delta'$$

PDA $\rightarrow$ CFG: Demostración $\mathcal{L}(\mathcal{P}) = \mathcal{L}(\mathcal{P}')$

Hipótesis de inducción (en el número de pasos n)

Para todo $p, p' \in Q$, $A \in \Gamma$, y $w \in \Sigma^*$ se cumple que:

$$(pA, w) \vdash^n_{\mathcal{P}} (p', \epsilon) \quad \text{si, y solo si,} \quad (q(p, A, p'), w) \vdash^n_{\mathcal{P}'} (q, \epsilon)$$

donde $\vdash^n_{\mathcal{P}}$ es la relación de **siguiente-paso** de $\mathcal{P}$ n-veces.

Si demostramos esta hipótesis habremos demostrado que $\mathcal{L}(\mathcal{P}) = \mathcal{L}(\mathcal{P}')$

¿por qué?

PDA $\rightarrow$ CFG: Demostración $\mathcal{L}(\mathcal{P}) = \mathcal{L}(\mathcal{P}')$

Caso base: $n = 1$

Para todo $p, p' \in Q$ y $A \in \Gamma$ se cumple que:

$$(pA, c) \vdash_{\mathcal{P}} (p', \epsilon) \quad \text{si, y solo si,} \quad (q(p, A, p'), c) \vdash_{\mathcal{P}'} (q, \epsilon)$$

para todo $c \in \big(\Sigma \cup \{\epsilon\}\big)$.

¿por qué?

PDA $\rightarrow$ CFG: Demostración $\mathcal{L}(\mathcal{P}) = \mathcal{L}(\mathcal{P}')$

Caso inductivo:

Sin perdida de generalidad,
suponga que $pA \overset{a}{\rightarrow} p_1 A_1 A_2$ y $w = auv$, entonces:

$$(pA, \underbrace{auv}_{w}) \vdash_{\mathcal{P}}^{n} (p', \epsilon) \quad \text{ssi} \quad (pA, auv) \vdash_{\mathcal{P}} (p_1 A_1 A_2, uv) \vdash_{\mathcal{P}}^{i} (p_2 A_2, v) \vdash_{\mathcal{P}}^{j} (p', \epsilon)$$

$$\text{ssi} \quad (p_1 A_1, u) \vdash_{\mathcal{P}}^{i} (p_2, \epsilon) \quad \text{y} \quad (p_2 A_2, v) \vdash_{\mathcal{P}}^{j} (p', \epsilon)$$

$$\text{ssi} \quad (q(p_1, A_1, p_2), u) \vdash_{\mathcal{P}'}^{i} (q, \epsilon) \quad \text{y} \quad (q(p_2, A_2, p'), v) \vdash_{\mathcal{P}'}^{j} (q, \epsilon)$$

$$\text{ssi} \quad (q(p, A, p'), auv) \vdash_{\mathcal{P}} (q(p_1, A_1, p_2)(p_2, A_2, q)), uv) \vdash_{\mathcal{P}}^{i+j} (q, \epsilon)$$

$\blacksquare$

Cierre de clase

En esta clase:

1. Terminamos equivalencia entre gramáticas y apiladores.
2. Apiladores pueden recorrer los árboles de derivación en profundidad.
3. Gramáticas pueden simular apiladores al remover el uso de estados.

Próxima clase: Parsing