Autómatas de Pila II
(*pushdown automata*)

Alan Reyes-Figueroa
Teoría de la Computación **(Aula 17) 28.septiembre.2022**

Equivalencia de PDAs y CFGs
Conversion of CFG to PDA
Conversion of PDA to CFG

Overview

- Cuando hablamos de propiedades de cerradura de lenguajes regulares, fue muy útil saltar entre las representaciones *regexp* y los DFA.
- Similarlmente, para las CFGs y los PDAs, ambos son útiles para mostrar propiedades relacionadas con los lenguajes libres del contexto.

Overview

- Además, los PDAs, siendo "algorítmicos," son más fáciles de usarcuando argumentamos que un lenguajes debe ser CFL.
- Ejemplo: Es más fácil ver cómo un PDA puede reconocer paréntesis balanceados; y menos vía una CFG.
- But all depends on knowing that CFG's and PDA's both define the CFL's.

Convirtiendo una CFG a PDA

- Sea L = L(G).
- Construimos un autómata de pila P tal que N(P) = L.
- P tiene:
  - Un estado q.
  - Símbolos input = terminales de G.
  - Símbolos stack = todos los símbolos de G.
  - Símbolo stack inicial = símbolo inicial de G.

Intuición acerca de P

- Dado el input w, P pasará por una derivación más a la izquierda de w desde el símbolo inicial S.
- Dado que P no puede saber cuál es esta derivación, ni siquiera cuál es el final de w, utiliza el no determinismo para "adivinar" la producción a utilizar en cada paso.

Intuición acerca de P

- En cada paso, P representa una *forma sentencial a la izquierda* (*left-sentential form*) (paso de una derivación leftmost).
- Si el stack de P es α, y P ha consumido una parte x de su input, entonces P representa la forma left-sentential xα.
- Cuando el stack sea vacío, el input consumido es una cadena en L(G).

Función de Transición de P

1. $\delta(q, a, a) = (q, \epsilon)$. (Reglas *Tipo 1*)
   - Este paso no cambia el LSF representado, sino que "mueve" la responsabilidad de *a* de la pila a la entrada consumida.
2. Si A -> α es una producción de G, luego $\delta(q, \epsilon, A)$ contiene (q, α). (Reglas *Tipo 2*)
   - Adivinar una producción para A y representar el siguiente LSF en la derivación.

Proof That L(P) = L(G)

- We need to show that $(q, wx, S) \vdash^* (q, x, \alpha)$ for any x if and only if $S =>^*_{lm} w\alpha$.
- Part 1: "only if" is an induction on the number of steps made by P.
- Basis: 0 steps.
  - Then $\alpha = S$, $w = \epsilon$, and $S =>^*_{lm} S$ is surely true.

Induction for Part 1

- Consider n moves of P: (q, wx, S) $\vdash$* (q, x, α) and assume the IH for sequences of n-1 moves.
- There are two cases, depending on whether the last move uses a Type 1 or Type 2 rule.

Use of a Type 1 Rule

- The move sequence must be of the form $(q, yax, S) \vdash^* (q, ax, a\alpha) \vdash (q, x, \alpha)$, where $ya = w$.
- By the IH applied to the first n-1 steps, $S =>^*_{lm} ya\alpha$.
- But $ya = w$, so $S =>^*_{lm} w\alpha$.

Use of a Type 2 Rule

- The move sequence must be of the form $(q, wx, S) \vdash^* (q, x, A\beta) \vdash (q, x, \gamma\beta)$, where $A \to \gamma$ is a production and $\alpha = \gamma\beta$.
- By the IH applied to the first n-1 steps, $S \Rightarrow^*_{lm} wA\beta$.
- Thus, $S \Rightarrow^*_{lm} w\gamma\beta = w\alpha$.

Proof of Part 2 (“if”)

- We also must prove that if $S \Rightarrow^*_{lm} w\alpha$, then $(q, wx, S) \vdash^* (q, x, \alpha)$ for any x.
- Induction on number of steps in the leftmost derivation.
- Ideas are similar; read in text.

Proof – Completion

- We now have $(q, wx, S) \vdash^* (q, x, \alpha)$ for any x if and only if $S \Rightarrow^*_{lm} w\alpha$.
- In particular, let $x = \alpha = \epsilon$.
- Then $(q, w, S) \vdash^* (q, \epsilon, \epsilon)$ if and only if $S \Rightarrow^*_{lm} w$.
- That is, w is in N(P) if and only if w is in L(G).

Convertir de PDA a CFG

- Ahora, supongamos que L = N(P).
- Construirmos una CFG G tal que L = L(G).
- Intuición: G tendrá variables que generan exactamente las entradas que hacen que P tenga el efecto de quitar un símbolo de pila X mientras pasa del estado p al q.
  - P nunca cae por debajo de esta X mientras lo hace.

Variables of G

- Las variables de G son de la forma [pXq].
- Esta variable genera todas y sólo aquellas cadenas w, tales que

$$(p, w, X) \vdash^* (q, \epsilon, \epsilon).$$

- Además, añadimos un símbolo inicial S, del cual hablaremos luego.

Productions of G

- Each production for [pXq] comes from a move of P in state p with stack symbol X.
- Simplest case: δ(p, a, X) contains (q, ϵ).
- Then the production is [pXq] -> a.
  - Note *a* can be an input symbol or ϵ.
- Here, [pXq] generates *a*, because reading *a* is one way to pop X and go from p to q.

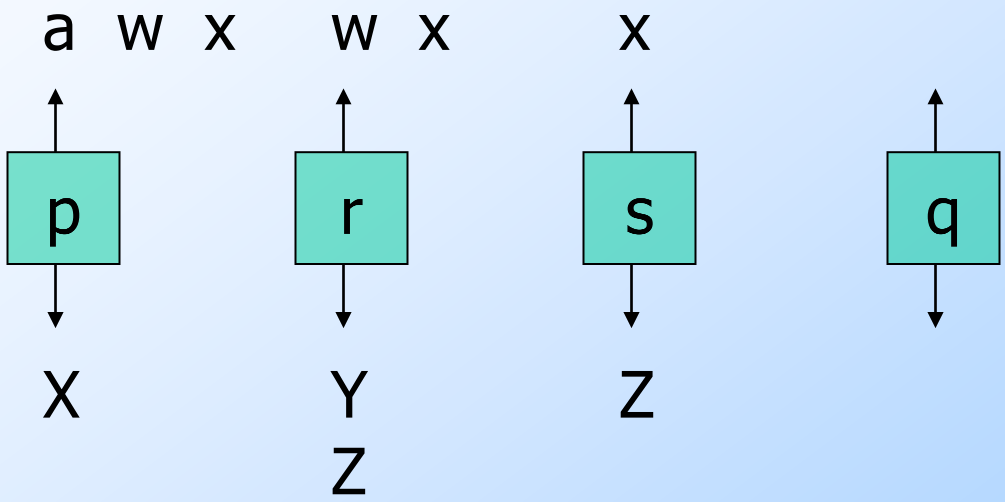
Productions of G – (2)

- Next simplest case: δ(p, a, X) contains (r, Y) for some state r and symbol Y.
- G has production [pXq] -> a[rYq].
  - We can erase X and go from p to q by reading *a* (entering state r and replacing the X by Y) and then reading some w that gets P from r to q while erasing the Y.
- Note: [pXq] =>* aw whenever [rYq] =>* w.

Productions of G – (3)

- Third simplest case: $\delta(p, a, X)$ contains (r, YZ) for some state r and symbols Y and Z.
- Now, P has replaced X by YZ.
- To have the net effect of erasing X, P must erase Y, going from state r to some state s, and then erase Z, going from s to q.

Picture of Action of P



Third-Simplest Case – Concluded

- Since we do not know state s, we must generate a family of productions:

  [pXq] -> a[rYs][sZq]

  for all states s.
- [pXq] =>* awx whenever [rYs] =>* w and [sZq] =>* x.

Productions of G: General Case

- Suppose $\delta(p, a, X)$ contains $(r, Y_1,\ldots Y_k)$ for some state r and $k \geq 3$.
- Generate family of productions

$$[pXq] \to a[rY_1s_1][s_1Y_2s_2]\ldots[s_{k-2}Y_{k-1}s_{k-1}][s_{k-1}Y_kq]$$

Completion of the Construction

- We can prove that $(q_0, w, Z_0) \vdash^* (p, \epsilon, \epsilon)$ if and only if $[q_0Z_0p] \Rightarrow^* w$.
  - Proof is in text; it is two easy inductions.
- But state p can be anything.
- Thus, add to G another variable S, the start symbol, and add productions $S \rightarrow [q_0Z_0p]$ for each state p.