Equivalence of CFG’s and PDA’s

Mridul Aanjaneya

Stanford University

July 24, 2012
A PDA is an automaton equivalent to the CFG in language-defining power.

Only the nonterministic PDA’s define all possible CFL’s.

But the deterministic version models parsers.

Most programming languages have deterministic PDA’s.
Recap: Intuition

- Think of an $\varepsilon$-NFA with the additional power that it can manipulate a stack.
- Its moves are determined by:
  1. The current state (of its NFA).
  2. The current input symbol (or $\varepsilon$), and
  3. The current symbol on top of its stack.
Recap: Intuition

- Being **nondeterministic**, the PDA can have a **choice** of next moves.
- In each choice, the PDA can:
  1. **Change state**, and also
  2. **Replace** the **top symbol** on the stack by a sequence of **zero or more symbols**.
     - Zero symbols = **pop**.
     - Many symbols = sequence of **pushes**.
A PDA is described by:

1. A finite set of states \( Q \), typically.
2. An input alphabet \( \Sigma \), typically.
3. A stack alphabet \( \Gamma \), typically.
4. A transition function \( \delta \), typically.
5. A start state \( q_0 \), in \( Q \), typically.
6. A start symbol \( Z_0 \), in \( \Gamma \), typically.
7. A set of final states \( F \subseteq Q \), typically.
Recap: The Transition Function

- Takes three arguments:
  1. A state in $Q$.
  2. An input which is either a symbol in $\Sigma$ or $\varepsilon$.
  3. A stack symbol in $\Gamma$.

- $\delta(q,a,Z)$ is a set of zero or more actions of the form $(p, \alpha)$.
  - $p$ is a state, $\alpha$ is a string of stack symbols.
Recap: Actions of the PDA

- If $\delta(q,a,Z)$ contains $(p,\alpha)$ among its actions, then one thing the PDA can do in state $q$, with $a$ at the front of the input, and $Z$ on top of the stack is:
  1. Change the state to $p$.
  2. Remove $a$ from the front of the input (but $a$ may be $\varepsilon$).
  3. Replace $Z$ on the top of the stack by $\alpha$. 
• Design a PDA to accept \( \{0^n1^n \mid n \geq 1\} \).

• The states:
  - \( q = \) start state. We are in state \( q \) if we have seen only 0’s so far.
  - \( p = \) we’ve seen at least one 1 and may now proceed only if the inputs are 1’s.
  - \( f = \) final state; accept.
• The stack symbols:
  • $Z_0 = \text{start symbol}$. Also marks the bottom of the stack, so we know we have counted the same number of 1’s as 0’s.
  • $X = \text{marker}$, used to count the number of 0’s seen on the input.
Example: PDA

- The transitions:
  - $\delta(q,0,Z_0) = \{(q,XZ_0)\}$. 
  - $\delta(q,0,X) = \{(q,XX)\}$. These two rules cause one $X$ to be pushed onto the stack for each 0 read from the input.
  - $\delta(q,1,X) = \{(p,\varepsilon)\}$. When we see a 1, go to state $p$ and pop one $X$.
  - $\delta(p,1,X) = \{(p,\varepsilon)\}$. Pop one $X$ per 1.
  - $\delta(p,\varepsilon,Z_0) = \{(f,Z_0)\}$. Accept at bottom.
Actions of the Example PDA

Mridul Aanjaneya  Automata Theory  11/ 53
Actions of the Example PDA
Actions of the Example PDA
Actions of the Example PDA

Mridul Aanjaneya  Automata Theory  15/ 53
Actions of the Example PDA
Actions of the Example PDA
• We can formalize the pictures just seen with an **instantaneous description** (ID).

• An ID is a **triple** \((q, w, \alpha)\), where:
  1. \(q\) is the **current state**.
  2. \(w\) is the **remaining input**.
  3. \(\alpha\) is the **stack contents**, top at the left.
The “Goes-To” Relation

- To say that ID \( I \) can becomes ID \( J \) in one move of the PDA, we can write \( I \models J \).
- Formally, \((q, aw, X\alpha) \models (p, w, \beta\alpha)\) for any \( w \) and \( \alpha \), if \( \delta(q, a, X) \) contains \((p, \beta)\).
- Extend \( \models \) to \( \models^* \), meaning zero or more moves, by:
  - **Basis:** \( I \models^* I \).
  - **Induction:** If \( I \models^* J \) and \( J \models K \), then \( I \models^* K \).
Using the previous example PDA, we can describe the sequence of moves by

\[(q, 000111, Z_0) \vdash (q, 00111, XZ_0) \vdash (q, 0111, XXZ_0) \]
\[\vdash (q, 111, XXXZ_0) \vdash (p, 11, XXZ_0) \]
\[\vdash (p, 1, XZ_0) \vdash (p, \varepsilon, Z_0) \]
\[\vdash (f, \varepsilon, Z_0) \]

Thus, \((q, 000111, Z_0) \vdash^* (f, \varepsilon, Z_0)\).
Using the previous example PDA, we can describe the sequence of moves by

\[(q, 000111, Z_0) \vdash (q, 00111, XZ_0) \vdash (q, 0111, XXZ_0)\]
\[\vdash (q, 111, XXXZ_0) \vdash (p, 11, XXZ_0)\]
\[\vdash (p, 1, XZ_0) \vdash (p, \varepsilon, Z_0)\]
\[\vdash (f, \varepsilon, Z_0)\]

Thus, \((q, 000111, Z_0) \vdash^* (f, \varepsilon, Z_0)\).

Question

What would happen on the input 0001111?
\[(q, 0001111, Z_0) \vdash (q, 001111, XZ_0) \vdash (q, 01111, XXZ_0)\]
\[\vdash (q, 1111, XXXZ_0) \vdash (p, 111, XXZ_0)\]
\[\vdash (p, 11, XZ_0) \vdash (p, 1, Z_0)\]
\[\vdash (f, 1, Z_0)\]

- **Note:** The last action is legal because a PDA can use ε input even if input remains.
- The last ID has no move.
- **0001111** is not accepted, because the input is not completely consumed.
Aside: FA and PDA Notations

- We represented moves of a FA by an extended $\delta$, which did not mention the input yet to be read.
- We could have chosen a similar notation for PDA's, where the FA state is replaced by a state-stack combination.
Similarly, we could have chosen a FA notation with ID’s.
  - Just drop the stack notation.

Why the difference?

- FA tend to models thinks like protocols with infinitely long inputs.
- PDA model parsers, which are given a fixed program to process.
The common way to define the language of a PDA is by final state.

If $P$ is a PDA, then $L(P)$ is the set of strings $w$ such that $(q_0, w, Z_0) \vdash^* (f, \varepsilon, \alpha)$ for final state $f$ and any $\alpha$. 
Another language defined by the same PDA is by empty stack.

If $P$ is a PDA, then $N(P)$ is the set of strings $w$ such that $(q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon)$ for any state $q$. 
Equivalence of Language Definitions

1. If $L = L(P)$, then there is another PDA $P'$ such that $L = N(P')$.

2. If $L = N(P)$, then there is another PDA $P''$ such that $L = L(P'')$. 
• P’ will simulate P.
• If P accepts, P’ will empty its stack.
• P’ has to avoid accidentally emptying its stack, so it uses a special bottom marker to catch the case where P empties its stack without accepting.
Proof: \( L(P) \rightarrow N(P') \)

- \( P' \) has all the states, symbols, and moves of \( P \), plus:
  1. Stack symbol \( X_0 \), used to guard the stack bottom against accidental emptying.
  2. New start state \( s \) and erase state \( e \).
  3. \( \delta(s, \varepsilon, X_0) = \{(q_0, Z_0 X_0)\} \). Get \( P \) started.
  4. \( \delta(f, \varepsilon, X) = \delta(e, \varepsilon, X) = \{(e, \varepsilon)\} \) for any final state \( f \) of \( P \) and any stack symbol \( X \).
Proof: $N(P) \rightarrow L(P'')$ Intuition

- $P''$ simulates $P$.
- $P''$ has a special bottom-marker to catch the situation where $P$ empties its stack.
- If so, $P''$ accepts.
Proof: \( N(P) \to L(P'') \)

- \( P'' \) has all the states, symbols, and moves of \( P \), plus:
  1. Stack symbol \( X_0 \), used to guard the stack bottom.
  2. New start state \( s \) and final state \( f \).
  3. \( \delta(s, \varepsilon, X_0) = \{(q_0, Z_0X_0)\} \). Get \( P \) started.
  4. \( \delta(q, \varepsilon, X_0) = \{(f, \varepsilon)\} \) for any state \( q \) of \( P \).
To be **deterministic**, there must be **at most** one choice of move for any state \( q \), input symbol \( a \), and stack symbol \( X \).

In addition, there must **not** be a choice between using input \( \varepsilon \) or **real input**.

Formally, \( \delta(q, a, X) \) and \( \delta(q, \varepsilon, X) \) **cannot** both be **nonempty**.
• When we talked about closure properties of regular languages, it was useful to be able to jump between RE and DFA representations.

• Similarly, CFG’s and PDA’s are both useful to deal with properties of CFL’s.
Also, PDA’s, being algorithmic, are often easier to use when arguing that a language is a CFL.

**Example:** It is easy to see how a PDA can recognize balanced parentheses, not so easy as a grammar.

But all depends on knowing that CFG’s and PDA’s both define the CFL’s.
• Let $L = L(G)$.
• Construct PDA $P$ such that $N(P) = L$.
• $P$ has:
  • One state $q$.
  • Input symbols = terminals of $G$.
  • Stack symbols = all symbols of $G$.
  • Start symbol = start symbol of $G$. 
Intuition about P

- Given input $w$, $P$ will step through a leftmost derivation of $w$ from the start symbol $S$.
- Since $P$ can’t know what this derivation is, or even what the end of $w$ is, it uses nondeterminism to guess the production to use at each step.
Intuition about $P$

- At each step, $P$ represents some left-sentential form (step of a leftmost derivation).
- If the stack of $P$ is $\alpha$, and $P$ has so far consumed $x$ from its input, then $P$ represents left-sentential form $x\alpha$.
- At empty stack, the input consumed is a string in $L(G)$. 
Transition Function of P

1. \( \delta(q,a,a) = (q,\varepsilon) \). (Type 1 rules)
   - This step does not change the LSF represented, but moves responsibility for \( a \) from the stack to the consumed input.

2. If \( A \rightarrow \alpha \) is a production of \( G \), then \( \delta(q,\varepsilon,A) \) contains \( (q,\alpha) \). (Type 2 rules)
   - Guess a production for \( A \), and represent the next LSF in the derivation.
Proof that $N(P) = L(G)$

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

- Consider \( n \) moves of \( P: (q,wx,S) \vdash^* (q,x,\alpha) \) and assume the \textbf{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.
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, \(S \Rightarrow_{lm}^* ya\alpha\).

But \(ya = w\), so \(S \Rightarrow_{lm}^* w\alpha\).
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 \(A \rightarrow \gamma\) is a production and \(\alpha = \gamma\beta\).
- By the IH applied to the first \(n-1\) steps, \(S \Rightarrow^*_l wA\beta\).
- Thus, \(S \Rightarrow^*_l w\gamma\beta = w\alpha\).
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.
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 \(x = \alpha = \varepsilon\).

Then \((q,w,S) \vdash^* (q,\varepsilon,\varepsilon)\) if and only if \(S \Rightarrow^*_{lm} w\).

That is, \(w \in N(P)\) if and only if \(w \in L(G)\).
Now assume $L = N(P)$.

We’ll construct a CFG $G$ such that $L = L(G)$.

**Intuition:** $G$ will have variables generating exactly the inputs that cause $P$ to have the net effect of popping a stack symbol $X$ while going from state $p$ to state $q$.

- $P$ never gets below this $X$ while doing so.
• G’s variables are of the form \([pXq]\).
• This variable generates all and only the strings \(w\) such that
  \[(p, w, X) \vdash^* (q, \varepsilon, \varepsilon)\]
• Also a start symbol \(S\) we’ll talk about later.
Productions of $G$  

- Each production for $[pXq]$ comes from a move of $P$ in state $p$ with stack symbol $X$.
- **Simplest case:** $\delta(p,a,X)$ contains $(q,\varepsilon)$.
- Then the production is $[pXq] \rightarrow a$.
  - Note that $a$ can be an input symbol or $\varepsilon$.
- Here, $[pXq]$ generates $a$, because reading $a$ is one way to pop $X$ and go from $p$ to $q$. 
Next simplest case: $\delta(p,a,X)$ contains $(r,Y)$ for some state $r$ and symbol $Y$.

$G$ has production $[pXq] \rightarrow a[rYq]$.

- 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: $[pXq] \Rightarrow^* aw$ whenever $[rYq] \Rightarrow^* w$. 
• **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 $q$. 
Action of P

Diagram:

1. a wx
   - p
   - X
   - Y
   - Z
2. wx
   - r
   - Y
   - Z
3. x
   - s
   - Z
4. q
• Since we do not know state $s$, we must generate a family of productions:

$$[pXq] \rightarrow a[rYs][sZq]$$

• It follows $[pXq] \Rightarrow^* awx$ whenever $[rYs] \Rightarrow^* w$ and $[sZq] \Rightarrow^* x$. 
• Suppose $\delta(p, a, X)$ contains $(r, Y_1, \ldots, Y_k)$ for some state $r$ and $k \geq 3$.

• Generate family of productions

$$[pXq] \rightarrow a[rY_1s_1][s_1Y_2s_2] \ldots [s_{k-2}Y_{k-1}s_{k-1}][s_{k-1}Y_kq]$$
Completion of the Construction

- We can prove that \((q_0, w, Z_0) \vdash^* (p, \varepsilon, \varepsilon)\) iff \([q_0Z_0p] \Rightarrow^* w\).
  - Proof is two easy inductions. Left as exercises.
- 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 \(p\).