Equivalence of CFG's and PDA's

The title says it all.

- We'll show a language \( L \) is \( L(G) \) for some CFG if and only if it is \( N(P) \) for some PDA \( P \).

Only If (CFG to PDA)

Let \( L = L(G) \) for some CFG \( G = (V, \Sigma, P, S) \).

- Idea: have PDA \( A \) simulate LM derivations in \( G \), where a left-sentential form is represented by:

  1. The sequence of input symbols that \( A \) has consumed from its input, followed by
  2. \( A \)'s stack, topmost.

- Example: If \( (q, abcd, S) \overset{*}{\rightarrow} (q, cd, ABC) \), then the LSF represented is \( abABC \).

Moves of \( A \)

- If a terminal \( a \) is on top of the stack, then \( A \) consumes \( a \) from the input and pops it from the stack, if so.
  - The LSF represented doesn't change!
- If a variable \( B \) is on top of the stack, then PDA \( A \) has a choice of replacing \( B \) on the stack by the body of any production with head \( B \).

Formal Construction of \( A \)

\[
A = (\{q\}, \Sigma, V \cup \Sigma, \delta, q_0, S), \quad \text{where } \delta \text{ is defined by:}
\]

1. If \( B \) is in \( V \), then \( \delta(q, \epsilon, B) = \{(q, \alpha) \mid B \rightarrow \alpha \text{ is in } P\} \).
2. If \( a \) is in \( \Sigma \), then \( \delta(q, a, a) = \{(q, \epsilon)\} \).

Example

\[
G = (\{S, A\}, \{0, 1\}, P, S), \quad \text{where } P \text{ consists of } S \rightarrow AS \mid \epsilon; \quad A \rightarrow 0A1 \mid A1 \mid 01.
\]

- \( A = (\{q\}, \{0, 1\}, \{0, 1, A, S\}, \delta, q_0, S), \) where \( \delta \) is defined by:
  - \( \delta(q, \epsilon, S) = \{(q, AS), (q, \epsilon)\} \)
  - \( \delta(q, \epsilon, A) = \{(q, 0A1), (q, A1), (q, 01)\} \)
  - \( \delta(q, 0, 0) = \{(q, \epsilon)\} \)
  - \( \delta(q, 1, 1) = \{(q, \epsilon)\} \)

Only-If Proof (i.e., Grammar \( \Rightarrow \) PDA)

- Prove by induction on the number of steps in the derivation \( \overset{*}{\rightarrow} \) that for any \( x \),
  \( (q, wx, S) \overset{*}{\rightarrow} (q, x, \beta) \), where
  1. \( w\beta = \alpha \).
  2. \( \beta \) is the suffix of \( \alpha \) that begins at the leftmost variable (\( \beta = \epsilon \) if there is no variable).
- Also prove the converse, that if \( (q, wx, S) \overset{*}{\rightarrow} (q, x, \beta) \), then \( S \Rightarrow w\beta \).
- Inductive proofs in reader.
- As a consequence, if \( y \) is a terminal string, then \( S \Rightarrow y \) iff \( (q, y, S) \overset{*}{\rightarrow} (q, \epsilon, \epsilon) \), i.e., \( y \) is in \( L(G) \) iff \( y \) is in \( N(A) \).

PDA to CFG

Assume \( L = L(P) \), where \( P = (Q, \Sigma, \delta, q_0, Z_0) \).

- Key idea: units of PDA action have the net effect of popping one symbol from the stack, consuming some input, and making a state change.
- The triple \( [qZp] \) is a CFG variable that generates exactly those strings \( w \) such that \( P \) can read \( w \) from the input, pop \( Z \) (net effect), and go from state \( p \) to state \( q \).
  - More precisely, \( (q, w, Z) \overset{*}{\rightarrow} (p, \epsilon, \epsilon) \).
  - As a consequence of above, \( (q, wx, Za) \overset{*}{\rightarrow} (p, x, \alpha) \) for any \( x \) and \( \alpha \).
- It's a Zen thing: \( [qZp] \) is at once a triple involving states and symbols of \( P \), and yet to the CFG we construct it is a single, indivisible object.
  - OK; I know that's not a Zen thing, but you get the point.
- Complete proof is in the reader. We'll just give some examples and the idea behind the construction.
- Example: a popping rule, e.g., \( (p, \epsilon) \) in \( \delta(q, a, Z) \).
  - \( [qZp] \rightarrow a \)
• A rule that replaces one symbol and state by others, e.g., \((p, Y)\) in \(\delta(q, a, Z)\).
  • For all states \(r\): \([qZr] \rightarrow a[pZr]\)
• A rule that replaces one stack symbol by two, e.g., \((p, XY)\) in \(\delta(q, a, Z)\).
  • For all states \(r\) and \(s\): \([qZs] \rightarrow a[pXr][rYs]\)

**Deterministic PDA’s**

Intuitively: never a choice of move.
• \(\delta(q, a, Z)\) has at most one member for any \(q, a, Z\) (including \(a = \epsilon\)).
• If \(\delta(q, \epsilon, Z)\) is nonempty, then \(\delta(q, a, Z)\) must be empty for all input symbols \(a\).

**Why Care?**

Parsers, as in YACC, are really DPDA’s.
• Thus, the question of what languages a DPDA can accept is really the question of what programming language syntax can be parsed conveniently.

**Some Language Relationships**

• Acceptance by empty stack is hard for a DPDA.
  • Once it accepts, it dies and cannot accept any continuation.
  • Thus, \(N(P)\) has the prefix property: if \(w\) is in \(N(P)\), then \(wx\) is NOT in \(N(P)\) for any \(x \neq \epsilon\).
• However, parsers do accept by emptying their stack.
  • Trick: they really process strings followed by a unique endmarker (typically \(\$\)) e.g., if they accept \(w\$\), they consider \(w\) to be a correct program.
• If \(L\) is a regular language, then \(L\) is a DPDA language.
  • A DPDA can simulate a DFA, without using its stack (acceptance by final state).
• If \(L\) is a DPDA language, then \(L\) is a CFL that is not inherently ambiguous.
  • A DPDA yields an unambiguous grammar in the standard construction.