CS 133 : Automata Theory and Computability

Lecture Slides[1]

Context-Free Grammars and Languages

Nestine Hope S. Hernandez

Algorithms and Complexity Laboratory
Department of Computer Science
University of the Philippines, Diliman
nshernandez@dcs.upd.edu.ph

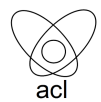
Day 12

[1] Reference: Intro to the Theory of Computation (2nd Ed), Sipser

Context-Free Grammars and Languages

Equivalence of PDAs and CFGs

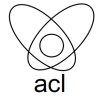
Deterministic Pushdown Automata



Context-Free and Non-Context Free Languages

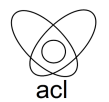
Equivalence of PDAs and CFGs

Deterministic Pushdown Automata



Theorem

A language is context free if and only if some (nondeterministic) pushdown automaton recognizes it.



From PDA to CFG

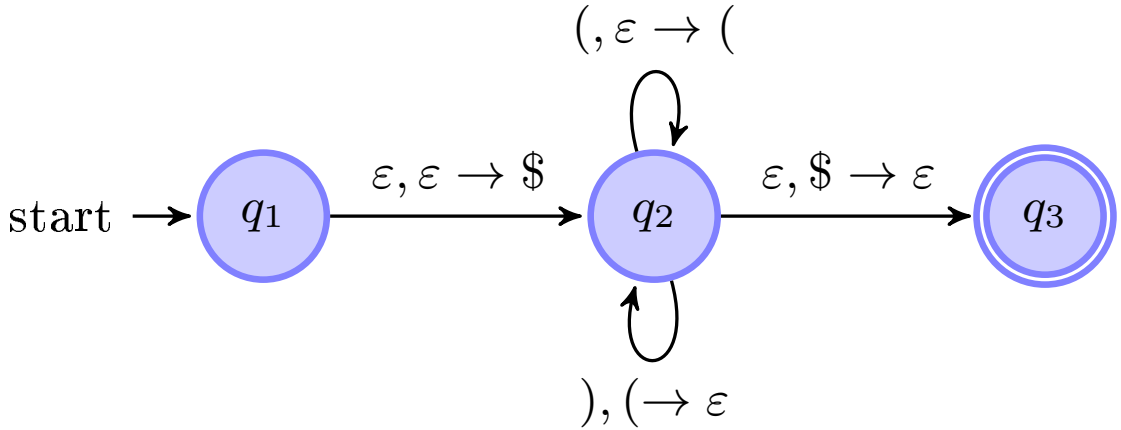
First, we slightly modify P to give it the following features:

1. It has a single accept state, q_{accept}.
2. It empties its stack before accepting.
3. Each transition either pushes a symbol onto the stack or pops one off the stack, but it does not do both at the same time.
   (Note: This also applies to epsilon moves, i.e. if P has an epsilon move, modify P such that some symbol (not in Σ) is pushed onto the stack and is popped off the stack immediately after.)

Let the PDA $P = (Q, \Sigma, \Gamma, \delta, q_{start}, \{q_{accept}\})$.
We construct the CFG G as follows:
The variables of G are $\{A_{pq} \mid p, q \in Q\}$.
The start variable is $A_{q_{start}, q_{accept}}$.
The rules of G are described as follows:

- For each $p, q, r, s \in Q, t \in \Gamma$, and $a, b \in \Sigma_\varepsilon$, if $\delta(p, a, \varepsilon)$ contains (r, t) and $\delta(s, b, t)$ contains (q, ε), put the rule $A_{pq} \to aA_{rs}b$ in G.
- For each $p, q, r, \in Q$, put the rule $A_{pq} \to A_{pr}A_{rq}$ in G.
- Finally, for each $p \in Q$, put the rule $A_{pp} \to \varepsilon$ in G.

An Example



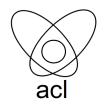
The variables of G are $\{A_{pq} \mid p, q \in Q\}$.
The start variable is $A_{q_{start}, q_{accept}}$.
The rules of G are described as follows:

- For each $p, q, r, s \in Q$, $t \in \Gamma$, and $a, b \in \Sigma_\varepsilon$, if $\delta(p, a, \varepsilon)$ contains (r, t) and $\delta(s, b, t)$ contains (q, ε), put the rule $A_{pq} \to aA_{rs}b$ in G.
- For each $p, q, r, \in Q$, put the rule $A_{pq} \to A_{pr}A_{rq}$ in G.
- Finally, for each $p \in Q$, put the rule $A_{pp} \to \varepsilon$ in G.

$$A_{13} \to \varepsilon A_{22} \varepsilon$$
$$A_{22} \to (A_{22})$$
$$A_{22} \to A_{22}A_{22}$$
$$A_{22} \to \varepsilon$$

Theorem

A language is context free if and only if some (nondeterministic) pushdown automaton recognizes it.



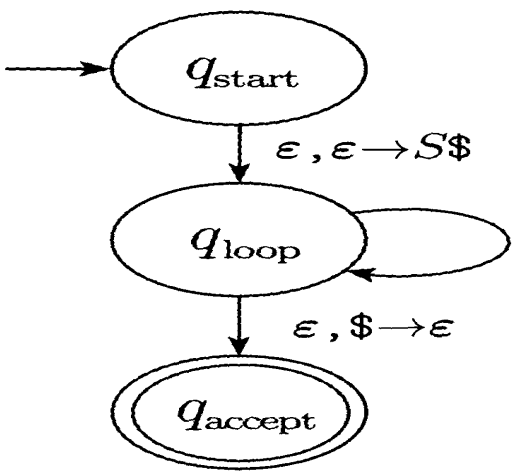
From CFG to PDA

1. Place the marker symbol \$ and the start variable on the stack.
2. Repeat the following steps forever.
   a. If the top of stack is a variable symbol A, nondeterministically select one of the rules for A and substitute A by the string on the right-hand side of the rule.
   b. If the top of the stack is a terminal symbol a, read the next symbol from the input and compare it to a. If they match, repeat. If they do not match, reject on this branch of the nondeterminism.
   c. If the top of the stack is the symbol \$, enter the accept state. Doing so accepts the input if it has all been read.



From CFG to PDA

1. Place the marker symbol \$ and the start variable on the stack.
2. Repeat the following steps forever.
   a. If the top of stack is a variable symbol A, nondeterministically select one of the rules for A and substitute A by the string on the right-hand side of the rule.
   b. If the top of the stack is a terminal symbol a, read the next symbol from the input and compare it to a. If they match, repeat. If they do not match, reject on this branch of the nondeterminism.
   c. If the top of the stack is the symbol \$, enter the accept state. Doing so accepts the input if it has all been read.



$\varepsilon, A \to w$ for rule $A \to w$

$a, a \to \varepsilon$ for terminal a

An Example

Construct a PDA from the following CFG:

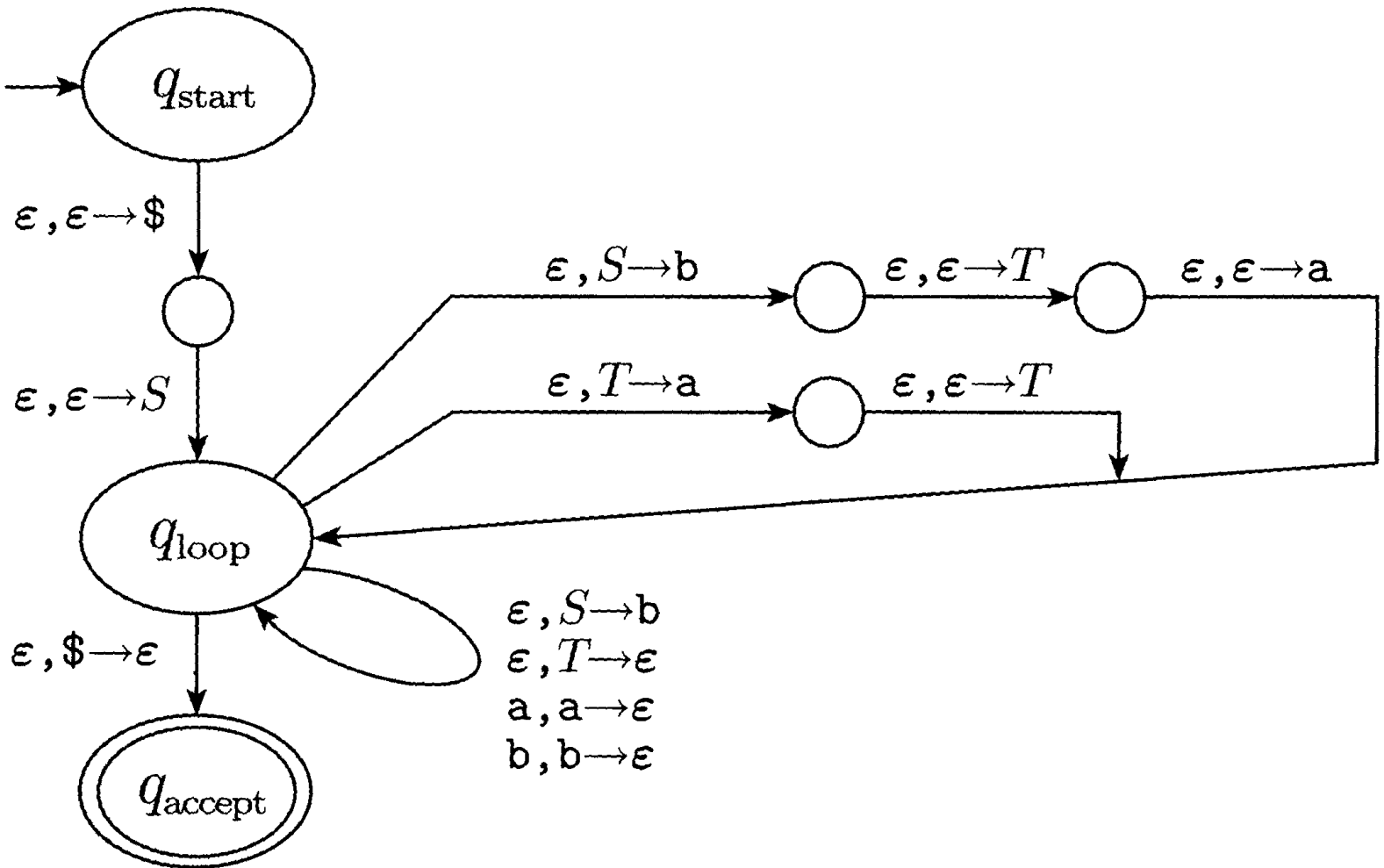
$$S \to aTb \mid b$$
$$T \to Ta \mid \varepsilon$$



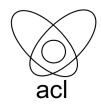

An Example

Construct a PDA from the following CFG:

$$S \to aTb \mid b$$
$$T \to Ta \mid \varepsilon$$



Context-Free and Non-Context Free Languages

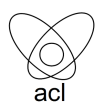


dPDA

Definition

A pushdown automaton $(Q, \Sigma, \Gamma, \delta, q_0, F)$, is deterministic if and only if the following conditions are met:

1. $\delta(q, a, t)$ has at most one element for any $q \in Q$, $a \in \Sigma_\varepsilon$, and $t \in \Gamma$.
2. If $\delta(q, a, t)$ is nonempty for some $a \in \Sigma$, then $\delta(q, \varepsilon, t)$ must be empty.



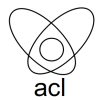
dPDA

Definition

A pushdown automaton $(Q, \Sigma, \Gamma, \delta, q_0, F)$, is deterministic if and only if the following conditions are met:

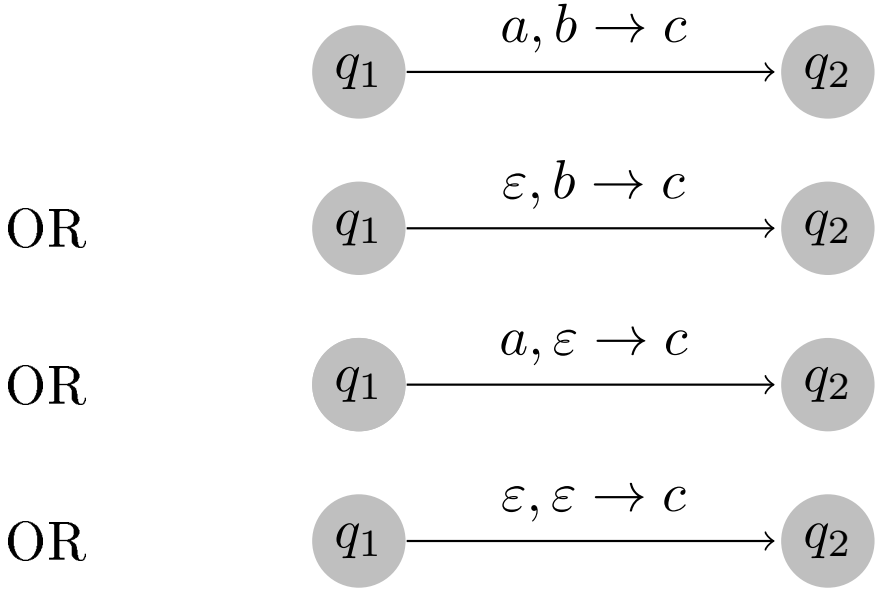
1. $\delta(q, a, t)$ has at most one element for any $q \in Q$, $a \in \Sigma_\varepsilon$, and $t \in \Gamma$.
2. If $\delta(q, a, t)$ is nonempty for some $a \in \Sigma$, then $\delta(q, \varepsilon, t)$ must be empty.

That is, a PDA is deterministic if it never has a choice of move for a given state, input symbol (including ε), and stack symbol. Also, it never has a choice between making a move using a true input and a move using ε input.

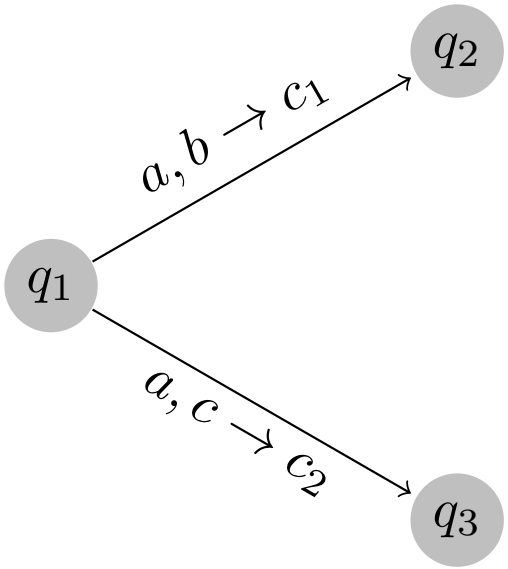


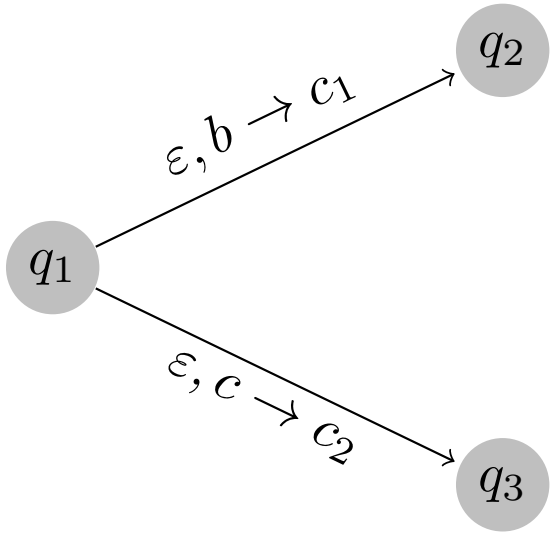
Allowed transitions in dPDA

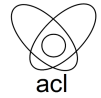
For any $a \in \Sigma$ and $b, c \in \Gamma$, only of these is possible:



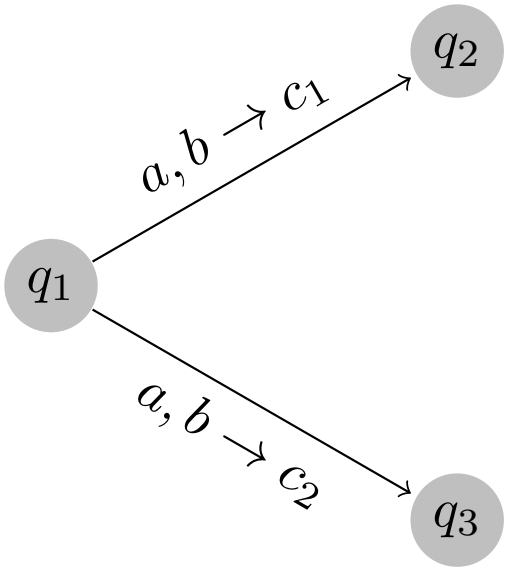
Allowed in dPDA: deterministic choices

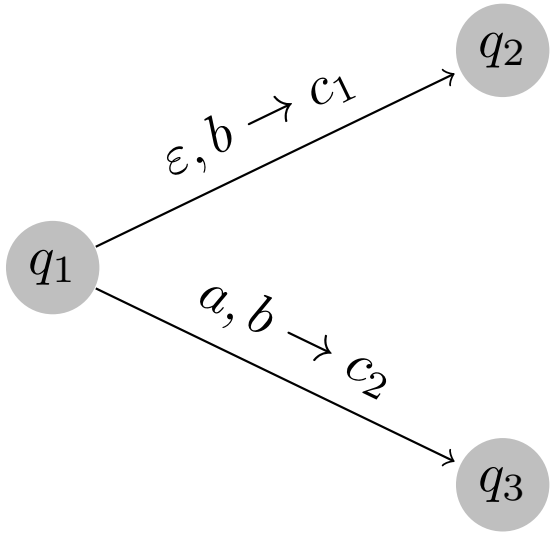






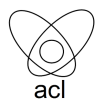
Not allowed in dPDA: nondeterministic choices



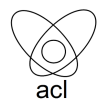


Acceptance by dPDA

Note that:
The two modes of acceptance, final state and empty stack, are not the same for dPDAs. Rather, the languages accepted by empty stack are exactly those of the languages accepted by final state that have the prefix property (i.e. no string in the language is a prefix of another word in the language).

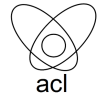


The languages recognized by dPDA



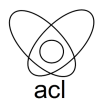
The languages recognized by dPDA

- All the regular languages are accepted (by final state) by dPDAs.



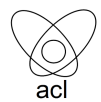
The languages recognized by dPDA

- All the regular languages are accepted (by final state) by dPDAs. But there are nonregular languages recognized by dPDAs.



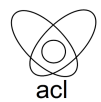
The languages recognized by dPDA

- All the regular languages are accepted (by final state) by dPDAs. But there are nonregular languages recognized by dPDAs.
- The dPDA languages are context-free languages, and in fact are languages that have unambiguous CFGs.



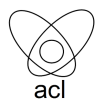
The languages recognized by dPDA

- All the regular languages are accepted (by final state) by dPDAs. But there are nonregular languages recognized by dPDAs.
- The dPDA languages are context-free languages, and in fact are languages that have unambiguous CFGs. But the dPDA languages are not exactly equal to the subset of the CFLs that are not inherently ambiguous.



The languages recognized by dPDA

- All the regular languages are accepted (by final state) by dPDAs. But there are nonregular languages recognized by dPDAs.
- The dPDA languages are context-free languages, and in fact are languages that have unambiguous CFGs. But the dPDA languages are not exactly equal to the subset of the CFLs that are not inherently ambiguous.
- Thus, the dPDA languages lie strictly between the regular languages and the context-free languages.



Questions?

See you next meeting!