Formal Languages and Automata

Chapter 7
Pushdown Automata

Chuan-Ming Liu

`cmliu@csie.ntut.edu.tw`

Department of Computer Science and Information Engineering
National Taipei University of Technology
Taipei, TAIWAN

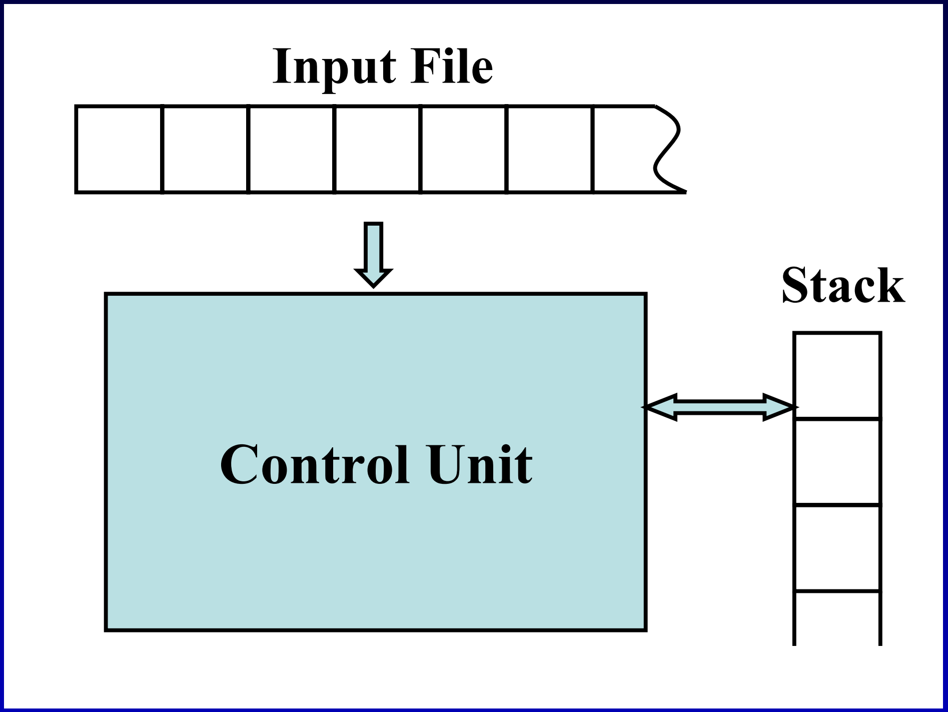
Objectives

- Study the connection between pushdown automata (pda's) and context-free languages
- Pushdown automata:
  - ***nondeterministic*** $\longrightarrow$ context-free languages
  - ***deterministic*** $\longrightarrow$ a subset of context-free languages

Contents

- Nondeterministic Pushdown Automata
- Pushdown Automata and Context-free Languages
- Deterministic Pushdown Automata and Deterministic Context-free Languages
- Grammars for Deterministic Context-free Languages

Nondeterministic PDA’s



Definition

A ***nondeterministic pushdown automata*** (npda) is defined by the septuple $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ where

Q: finite set of states
Σ: input alphabet
Γ: stack alphabet
$\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \longrightarrow$ finite subsets of $Q \times \Gamma^*$ is the ***transition function***
q_0: initial state
z: stack start symbol
F: set of final states

Notes on Transition Functions

1. the second argument of δ may be λ, indicating that a move that does not consume an input symbol is possible. (λ-transition)
2. no move is possible if the stack is empty
3. the range of δ is finite
4. the insertion of a string into a stack is done symbol by symbol from right to left
5. Au unspecific transition is to the null set and represents a ***dead configuration*** for the npda.

Example 1

Consider an nondeterministic pushdown automaton with $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{a, b\}$, $\Gamma = \{0, 1\}$, $z = \emptyset$, and $F = \{q_3\}$ with initial state q_0 and

$$\begin{aligned}
\delta(q_0, a, 0) &= \{(q_1, 10), (q_3, \lambda)\} \\
\delta(q_0, \lambda, 0) &= \{(q_3, \lambda)\} \\
\delta(q_1, a, 1) &= \{(q_1, 11)\} \\
\delta(q_1, b, 1) &= \{(q_2, \lambda)\} \\
\delta(q_2, b, 1) &= \{(q_2, \lambda)\} \\
\delta(q_2, \lambda, 0) &= \{(q_3, \lambda)\}
\end{aligned}$$

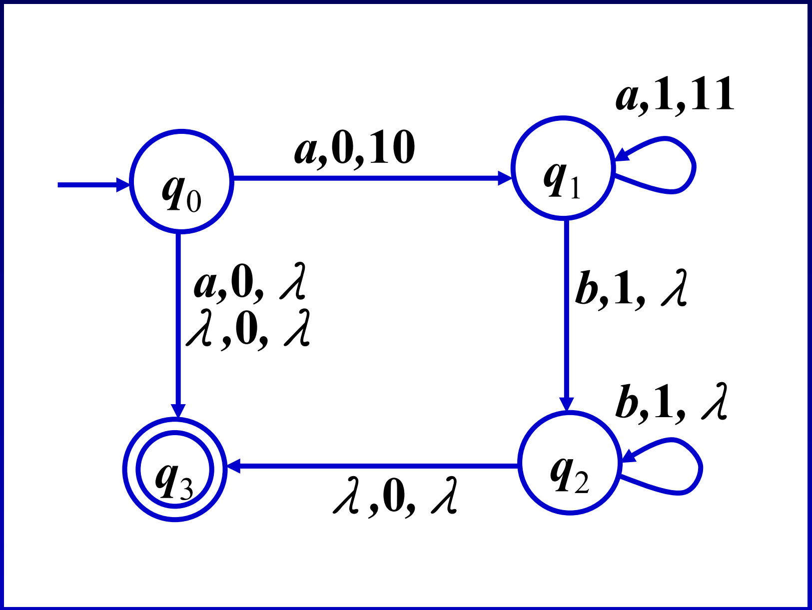
Transition Graphs for Npda's

One can also use ***transition graphs*** to represent npda's (in a similar way to fa) and each edge is labeled with

1. current input symbol,
2. symbol at the top of the stack, and
3. string that replaces the top of the stack

Example 2

Consider the npda in Example 1. The transition graph is as follows.



Instantaneous Description

An ***instantaneous description*** of a pushdown automaton is the triplet (q, w, u), where

q: the state of the control unit
w: the unread part of the input string
u: the stack contents

Note:

1. notation to describe the successive configurations
2. a move from one instantaneous description to another is denoted by $\vdash$, and

$$(q_1, aw, bx) \vdash (q_2, w, yx) \Leftrightarrow (q_2, y) \in \delta(q_1, a, b)$$

3. A series of steps is denoted by $\vdash^*$

Languages Accepted by a PDA

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ be a npda. The language accepted by M is the set

$$L(M) = \{w \in \Sigma^* : (q_0, w, z) \vdash_M^* (p, \lambda, u), p \in F, u \in \Gamma^*\}.$$

Note:

1. the language accepted by M is the set of all strings that can put M into a finite state at the end of the string.
2. the final stack content u is irrelevant.

Example 3

Construct an npda for

$$L = \{w \in \{a, b\}^* : n_a(w) = n_b(w)\}.$$

Example 4

Construct an npda for

$$L = \{ww^R : w \in \{a, b\}^+\}.$$

Contents

- Nondeterministic Pushdown Automata
- Pushdown Automata and Context-free Languages
- Deterministic Pushdown Automata and Deterministic Context-free Languages
- Grammars for Deterministic Context-free Languages

PDA's and CFL's

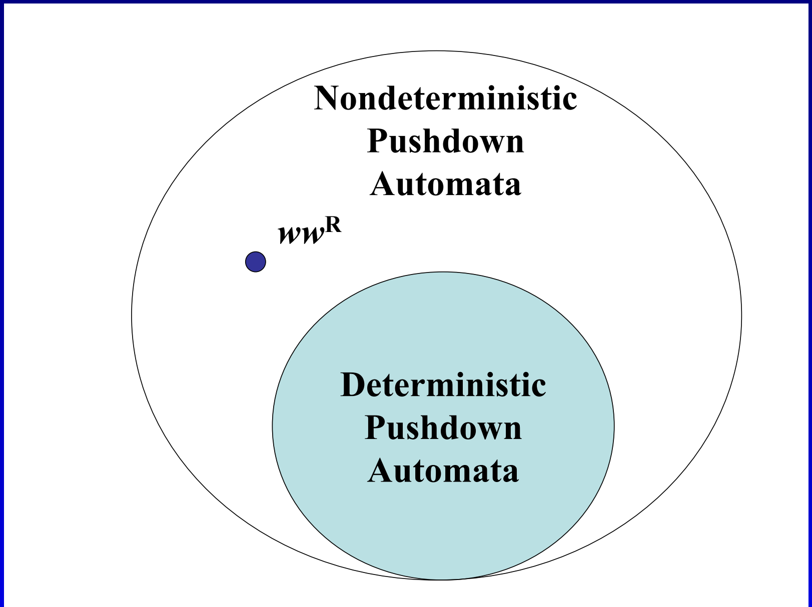
- For any context-free language, there exists an nondeterministic pushdown automaton that accepts, and the language accepted by any nondeterministic pushdown automaton is context-free
- For finite automata, the deterministic and nondeterministic models are equivalent with respect to the languages accepted. However, this is NOT TRUE for pushdown automata.

NPDA *v.s.* DPDA

Consider

$$L = \{ww^R : w \in \{a, b\}^+\}.$$

This language is accepted by an npda but no dpda.



Acceptance by Empty Stack

- Recall that the definition about the language L accepted by a pushdown automaton:

$$L(M) = \{w \in \Sigma^* : (q_0, w, z) \vdash_M^* (p, \lambda, u), p \in F, u \in \Gamma^*\}.$$

The final stack content u is then ignored.

- Actually, we can define the language accepted by a pda as the set of all inputs for which some sequence of moves cause the pda to empty the stack – accepted by empty stack.

Definitions

For a pushdown automaton M,

1. $T(M)(= L(M))$ is the language accepted by final states.
2. $N(M)$ is the language accepted by empty stack.

Note: When accepting by empty stack, the set of final states is irrelevant and can be the empty set, *i.e.*,

$$M = (Q, \Sigma, \Gamma, \delta, q_0, z, \emptyset).$$

Theorem 1

$L = N(M_1)$ for some pda $M_1 \Leftrightarrow L = T(M_2)$ for some pda M_2

Theorem 2

For any context-free language L, there exists an nondeterministic pushdown automaton M such that

$$L = L(M).$$

Example 5

Consider the grammar

$$\begin{aligned} S &\rightarrow aA \\ A &\rightarrow aABC|bB|a \\ B &\rightarrow b \\ C &\rightarrow c \end{aligned}$$

Construct an npda M.

Theorem 3

If $L = N(M)$ for some pda M, the L is a context-free language.

Example 6

Let $M = (\{q_0, q_1\}, \{0, 1\}, \{X, Z_0\}, \delta, q_0, Z_0, \emptyset)$ where δ includes

$$
\begin{array}{ll}
\delta(q_0, 0, Z_0) = \{(q_0, XZ_0)\} & \delta(q_1, 1, X) = \{(q_1, \lambda)\} \\
\delta(q_0, 0, X) = \{(q_0, XX)\} & \delta(q_1, \lambda, X) = \{(q_1, \lambda)\} \\
\delta(q_0, 1, X) = \{(q_1, \lambda)\} & \delta(q_1, \lambda, Z_0) = \{(q_1, \lambda)\}
\end{array}
$$

Construct a context-free grammar G such that $L(G) = N(M)$.

Example 7

Consider an nondeterministic pushdown automaton

$$\begin{aligned}
\delta(q_0, a, z) &= \{(q_0, Az)\} \\
\delta(q_0, a, A) &= \{(q_0, A)\} \\
\delta(q_0, b, A) &= \{(q_1, \lambda)\} \\
\delta(q_1, \lambda, z) &= \{(q_2, \lambda)\}
\end{aligned}$$

Construct a context-free grammar G.

Notes

The following three statements are equivalent:

1. L is a context-free language
2. $L = N(M_1)$ for some pushdown automata M_1
3. $L = T(M_2)$ for some pushdown automata M_2

Contents

- Nondeterministic Pushdown Automata
- Pushdown Automata and Context-free Languages
- Deterministic Pushdown Automata and Deterministic Context-free Languages
- Grammars for Deterministic Context-free Languages

DPDA's and DCFL's

Definition:(*deterministic pushdown automaton*)
A dpda $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ is a pushdown automaton in which

1. for each $q \in Q$ and $z \in \Gamma$, whenever $\delta(q, \lambda, z)$ is nonempty, then $\delta(q, a, z)$ is empty for all $a \in \Sigma$.
2. for no $q \in Q$, $z \in \Gamma$, and $a \in \Sigma \cup \{\lambda\}$ does $\delta(q, a, z)$ contain more than one element.

Definition: (*deterministic context-free languages*)
A language L is said to be deterministic context-free if and only if there exists a dpda M such that $L = L(M)$.

Example 6

The language

$$L = \{a^n b^n : n \geq 0\}$$

is a dcfl, where the dpda
$M = (\{q_0, q_1, a_2\}, \{a, b\}, \{1, 0\}, \delta, q_0, 0, \{q_0\}\}$ with

$$\begin{aligned}
\delta(q_0, a, 0) &= \{(q_1, 10)\} \\
\delta(q_1, a, 1) &= \{(q_1, 11)\} \\
\delta(q_1, b, 1) &= \{(q_2, \lambda)\} \\
\delta(q_2, b, 1) &= \{(q_2, \lambda)\} \\
\delta(q_2, \lambda, 0) &= \{(q_0, \lambda)\}
\end{aligned}$$

Contents

Grammars for dcfl's

- The importance of deterministic context-free languages lies in the fact that they can be parsed efficiently.
- Consider to parse a string w top-down for deriving the leftmost derivation

$$\begin{array}{rcl} w & : & a_1a_2\cdots a_{i-1}|a_i\cdots a_n \\ \text{sentential form} & : & a_1a_2\cdots a_{i-1}|A \end{array}$$

Is there a production rule for each step?
Question: whether are there grammars that allow us to do so?

S-Grammars

Recall the S-grammar where $A \longrightarrow aX$ and the pair (A, a) appears once in the productions.

- given a string $w = w_1 w_2$ and we have derived $w = w_1 A x$.
- suppose the leftmost symbol of w_2 is a.
- If there is a rule $A \longrightarrow ay$, then the approach continues; otherwise, $w \notin L(G)$.

So, S-grammar is possible but restrictive for the syntax of programming languages.

LL-Grammars

- by looking ahead part of the input, one can predict exactly which production rule must be used
- the first L stands for the fact that the input is scanned from left to right
- the second L indicates that leftmost derivation is considered.

Example 8: $S \to aSb|ab$ is not an s-grammar, but is an LL-grammar.

Definition

Let $G = (V, T, S, P)$ be a cfg. If for each pair of leftmost derivations

$$\begin{aligned} S &\Rightarrow^* & w_1 A x_1 \Rightarrow w_1 y_1 x_1 \Rightarrow^* w_1 w_2 \\ S &\Rightarrow^* & w_1 A x_2 \Rightarrow w_1 y_2 x_2 \Rightarrow^* w_1 w_3 \end{aligned}$$

with $w_1, w_2, w_3 \in T^*$, the equality of the k leftmost symbols of w_2 and w_3 implies $y_1 = y_2$, then G is said to be an *LL(k)* grammar.
In other words, a grammar is an $LL(k)$ grammar if we can uniquely identify the correct production, given the current scanned symbol and a "look-ahead" of the next $k - 1$ symbols.