CS3063 Theory of Computing

Semester 4 (21 Intake), Jan – May 2024

Lecture 8

Context-Free Languages: Session 3

Sanath Jayasena

Previous Lecture

- Context-free Languages
  - Ambiguous CFGs (continued)
  - Simplified Forms and Normal Forms
  - Pushdown Automata (PDA)
    - Definition
    - Acceptance
  - Examples: *SimplePal*, *Pal* languages and corresponding PDA

Today's Outline:

Lecture 8

Context-free Languages (CFLs) - 3

- **Acceptance by PDA**
  - **Review & Further Discussion**
- **Non-determinism in PDA**
- **PDA for a CFG**
  - **Top-Down Approach**
  - **Bottom-up Approach**

PART 1

Outline:

Lecture 8

Context-free Languages (CFLs) - 3

- **Acceptance by PDA**
  - **Review & Further Discussion**
- **Non-determinism in PDA**
- **PDA for a CFG**
  - **Top-Down Approach**
  - **Bottom-up Approach**

Recall: Acceptance by a PDA

- **Definition**: If M=$(Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$ is a PDA and x is in Σ^*, x is accepted by M if

  $$(q_0, x, Z_0) \vdash^*_M (q, \Lambda, \alpha)$$

  for some α in Γ^* and some q in A

  – The stack may or may not be empty because $\alpha=\Lambda$ or $\alpha\neq\Lambda$

More on Acceptance

- Here acceptance depends only on the state, not the stack contents
  - *Accepting configuration* is any configuration in which the state is an accepting state

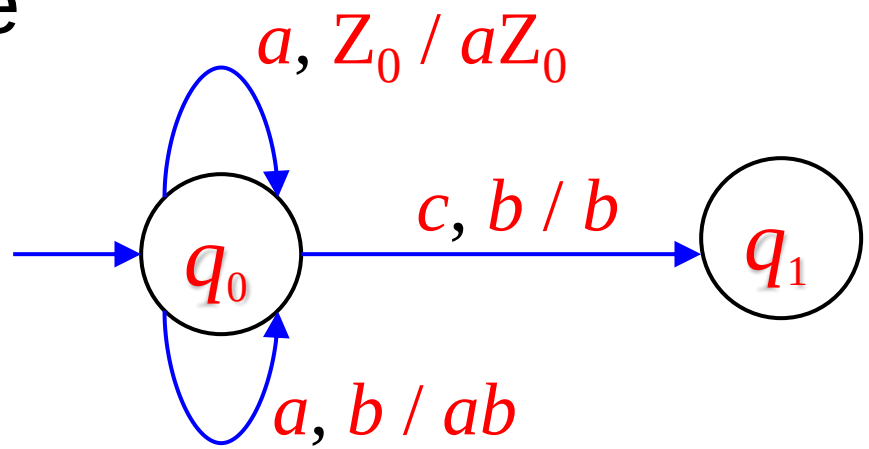
- This is "acceptance by final state"

More on Acceptance …contd

- Can define "acceptance by empty stack"
  - If we reach a configuration with empty stack
  - Regardless of whether we reach an accepting state
- Two types of acceptance are equivalent
  - If a language is accepted by a PDA using one mode of acceptance, there is another PDA using the other mode that accepts the language

Transition Diagram for a PDA?

- More complicated than that for an FA
- Example



  - a, X / β : transition (move) occur at state q_0 for input a with a stack symbol X; replace X by β on stack
  - To follow arrows, keep track of stack contents

Transition Table for a PDA?

- Alternative to the transition diagram
- Example

Move #	State	Input	Stack Symbol	Move(s)
1	q_0	a	Z_0	(q_0 , aZ_0)
2	q_0	b	Z_0	(q_0 , bZ_0)
3	q_0	a	a	(q_0 , aa)
…	….	…	…	….
12	q_1	Λ	Z_0	(q_2 , Z_0)

PART 2

Outline:

Lecture 8

Context-free Languages (CFLs) - 3

- Acceptance by PDA
  - Review & Further Discussion
- **Non-determinism in PDA**
- PDA for a CFG
  - Top-Down Approach
  - Bottom-up Approach

Non-determinism in PDA

- Recall 2 recent examples we discussed
  - Example 1 ***SimplePal*** (p. 251, Example 7.1)
    - What is the PDA to accept odd length palindromes over {a, b} with c as the middle symbol?
  - Example 2 ***Pal*** (p. 257, Example 7.2)
    - What is the PDA to accept all palindromes (odd or even length) over {a, b}?

Example 1: *SimplePal*

- Consider the previous CFG again

$$S \rightarrow aSa \mid bSb \mid c$$

  – Generates odd-length palindromes of {a, b} with c being the middle symbol

- What is the PDA to accept the language?
  – Discussion from E.g. 7.1 (p. 251)

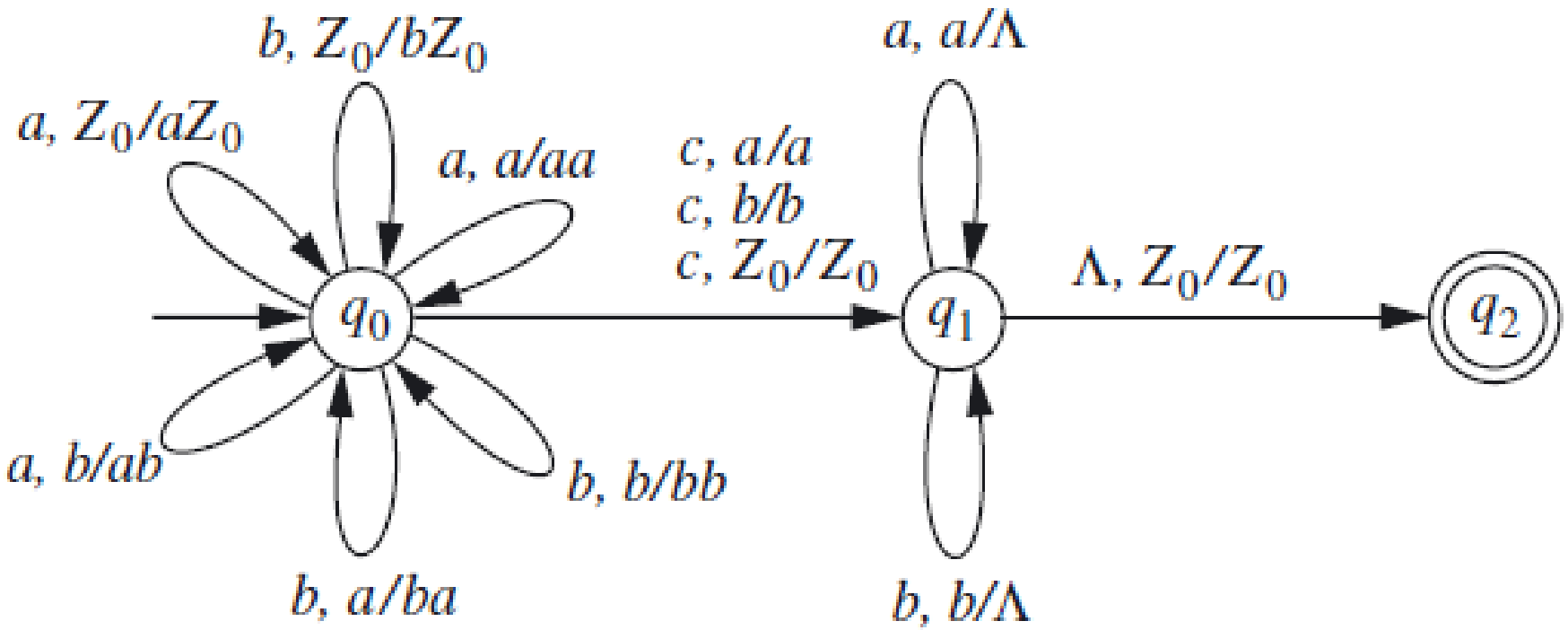
Example 1: *SimplePal*

- Simple palindrome recognizer
  - 3 states $Q=\{q_0, q_1, q_2\}$; q_0 initial state, q_2 accepting state
  - Input alphabet $\Sigma=\{a, b, c\}$
  - Stack alphabet $\Gamma=\{a, b, Z_0\}$
  - Transition function δ in Table 7.1 (p. 254)
    - Next slide (→all other combinations: "none" move)
  - "Transition diagram" in Fig. 7.1 (p. 255)

Transition Table

Move #	State	Input	Stack Symbol	Move(s)
1	q_0	a	Z_0	(q_0 , aZ_0)
2	q_0	b	Z_0	(q_0 , bZ_0)
3	q_0	a	a	(q_0 , aa)
4	q_0	b	a	(q_0 , ba)
5	q_0	a	b	(q_0 , ab)
6	q_0	b	b	(q_0 , bb)
7	q_0	c	Z_0	(q_1 , Z_0)
8	q_0	c	a	(q_1 , a)
9	q_0	c	b	(q_1 , b)
10	q_1	a	a	(q_1 , Λ)
11	q_1	b	b	(q_1 , Λ)
12	q_1	Λ	Z_0	(q_2 , Z_0)

Transition Diagram



Example 2: *Pal*

- Example 7.2 in text book p. 257
- A PDA accepting the language of all palindromes over {a, b} even or odd-length
  - When we reach the mid-point, if odd-length, discard mid symbol
  - Push all symbols in first half to stack
  - Match second half symbols to symbols on stack

Example 2: *Pal*

- How does PDA know mid-point reached?
  - PDA has to **guess**
  - Guessing OK if non-palindromes not accepted
  - First a sequence of "***not yet***" guesses
    - Push each symbol to stack (first half)
  - Then a "***yes***" guess stops the "not yet" series
    - Odd-length (xsx^r): discard next, match after that
    - Even-length (xx^r): : match from next
  - After that no more guesses

Example 2: *Pal*

- Consequences of above guessing
  - Non-palindromes will not be accepted
  - Can accept all palindromes
    - make the correct "yes" guess at the right time
  - PDA may also guess at the wrong time for a given palindrome
    - May not accept or accept a different palindrome
- Table 7.2, Fig. 7.2 (pp. 258, 259)
    - Next slide (→all other combinations: "none" move)

Transition Table

Choices (Guessing) in moves:
1-6: Not yet vs. Yes-Odd

1,2 vs. **7**:
3,**4** vs. **8**:
5,**6** vs. **9**:
Yes-Odd vs. Yes-Even

Move #	State	Input	Stack Symbol	Move(s)
1	q_0	a	Z_0	$(q_0 , aZ_0), (q_1 , Z_0)$
2	q_0	b	Z_0	$(q_0 , bZ_0), (q_1 , Z_0)$
3	q_0	a	a	$(q_0 , aa), (q_1 , a)$
4	q_0	b	a	$(q_0 , ba), (q_1 , a)$
5	q_0	a	b	$(q_0 , ab), (q_1 , b)$
6	q_0	b	b	$(q_0 , bb), (q_1 , b)$
7	q_0	Λ	Z_0	(q_1 , Z_0)
8	q_0	Λ	a	(q_1 , a)
9	q_0	Λ	b	(q_1 , b)
10	q_1	a	a	(q_1 , Λ)
11	q_1	b	b	(q_1 , Λ)
12	q_1	Λ	Z_0	(q_2 , Z_0)

Example 2: *Pal*

Non-determinism in this PDA

1. First 6 rows (moves 1-6) → 2 possible moves
   - Non-determinism with two choices
     - "Not yet" guess: push input to stack, remain in q_0
     - "Yes-Odd" guess: input symbol is middle (discard it), go to q_1
2. Rows 1-2 vs. 7, rows 3-4 vs. 8, rows 5-6 vs. 9:
   - Guess: Yes-Odd vs. Yes-Even, go to q_1
   - Odd: discard input, Even: not discard (Λ-transition)

- State q_0 → all guessing occurs here
- State q_1 → all comparison-making here

Non-determinism in PDA

- Example 1: ***SimplePal***
  - PDA to accept odd length palindromes over {a, b} with c as the middle symbol
    - Never has a choice of more than one move
    - This is a **deterministic PDA** (or **DPDA**)
- Example 2: ***Pal***
  - PDA to accept all palindromes (odd or even length) over {a, b}?
    - Has **two forms of non-determinism**

Non-determinism in PDA

- Two types of non-determinism can exist
  1. *There are two or more moves involving the same combination of state, stack symbol and input symbol*
  2. *For some combination of state and stack symbol, the PDA has a choice of reading an input symbol or making a Λ-transition*
- A *deterministic PDA* (or **DPDA**) has no configuration with a choice of > 1 move

Non-determinism in PDA

- A CFL is a *deterministic CFL* (or **DCFL**) if it there is a DPDA accepting it
- Recall that a regular language can be accepted by an NFA or an FA
- But not every CFL is accepted by a DPDA
  - Language of palindromes requires non-determinism in the PDA
  - Language of palindromes is not a DCFL

PART 3

Outline:

Lecture 8

Context-free Languages (CFLs) - 3

- Acceptance by PDA
  - Review & Further Discussion
- Non-determinism in PDA
- **PDA for a CFG**
  - **Top-Down Approach**
  - Bottom-up Approach

PDA for a CFG

- Every CFG can be recognized by a PDA

- For a given CFG, G, we want to build a PDA to determine whether a given (arbitrary) string can be derived from G
  - Strategy: *simulate a derivation*
  - Can involve guessing (non-determinism)

PDA for a CFG …contd

- A step in simulation = constructing a portion of the derivation tree
- 2 natural ways to simulate a derivation (using order of constructing the portions)
  - Top-down
  - Bottom-up

Top-Down Approach

- Start by pushing the start symbol S (at the top of derivation tree) onto the stack

- Next, at each step, replace a non-terminal on the stack (a node in the tree) by the RHS of a production that has that non-terminal on LHS
  - i.e., add children of that node to the tree

Top-Down Approach …contd

- Stack contains a portion of the current string in the derivation
- 2 types of moves by PDA, after push S
  1. Replace non-terminal A on top of stack by RHS α of some production $A \rightarrow \alpha$ (may involve guessing)
  2. Pop a terminal from stack if it matches the next input symbol and then discard both

Top-Down Approach …contd

- At each step, current string in derivation = the string of input symbols read so far followed by contents of stack

- A non-terminal appears on top of stack means terminals preceding it have already been matched and it is the *leftmost* non-terminal in current string
  - We are simulating a leftmost derivation

Top-Down Approach …contd

- Suppose G=(V, Σ, S, P) is a CFG
- Top-down PDA M that accepts L(G) is:
  - M=$(Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$ such that
  - Q ={q_0, q_1, q_2}, A={q_2}, Γ= V U Σ U {Z_0}, Z_0 is not in V U Σ
  - Initially, place S on stack and move to q_1: $\delta(q_0, \Lambda, Z_0)$ = {(q_1, SZ_0)}

Top-Down Approach …contd

- PDA M that accepts L(G) is: (…contd)
  - Only move to accepting state q_2 is from q_1, when stack is empty, except for Z_0 : $\delta(q_1, \Lambda, Z_0) = \{(q_2, Z_0)\}$
  - Otherwise, the only 2 moves of M are:
  1. For every X in V, $\delta(q_1, \Lambda, X) = \{(q_1, \alpha) \mid X \rightarrow \alpha$ is in P$\}$
  2. For every a in Σ, $\delta(q_1, a, a) = \{(q_1, \Lambda)\}$

Example

- See Example 7.5 on p. 269
  - Strings over {a, b} with more a's than b's
- CFG given as:

$$S \rightarrow a \mid aS \mid bSS \mid SSb \mid SbS$$

- Our PDA is, $M=(Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$ where:
  - $Q = \{q_0, q_1, q_2\}$, $A = \{q_2\}$
  - $\Sigma = \{a, b\}$, $\Gamma = \{S, a, b, Z_0\}$

Example ...contd

- Transition function δ defined by moves:

State	Input	Stack Symbol	Move(s)
q_0	Λ	Z_0	(q_1 , SZ_0)
q_1	Λ	S	(q_1 , a), (q_1, aS), (q_1, bSS), (q_1, SSb), (q_1, SbS)
q_1	a	a	(q_1 , Λ)
q_1	b	b	(q_1 , Λ)
q_1	Λ	Z_0	(q_2 , Z_0)
All other combinations			none

– Exercise: Check if the string *abbaaa* is in L(G)

PART 4

Outline:

Lecture 8

Context-free Languages (CFLs) - 3

- Acceptance by PDA
  - Review & Further Discussion
- Non-determinism in PDA
- **PDA for a CFG**
  - Top-Down Approach
  - **Bottom-up Approach**

Bottom-Up Approach

- Opposite counterparts to previous moves
  - Instead of replacing a non-terminal A on the stack by the RHS α of some production $A \rightarrow \alpha$, remove α from stack and replace it by A
  - This is called "reducing α to A"

- Derivation tree is built bottom-up
  - Start at leaves and extend upwards

Bottom-Up Approach ...contd

- Contents of stack represents a portion of string in the derivation being simulated
- The PDA "shifts" a terminal from the input to the end of this portion to prepare for a "reduction"

- Note: shifting reverses the order of input
  - string α (to be reduced to A) will appear on the stack in reverse

Bottom-Up Approach ...contd

- A reduction here requires a sequence of moves (while top-down method requires one move to apply a production)

- Process ends when start symbol S, which is on the stack due to the last reduction, is popped off so that only Z_0 is left in it

Bottom-Up Approach ...contd

- A derivation of input string in reverse order
- At each step, current string in derivation is the contents of stack (in reverse) followed by the string of unread input
- After each reduction, the non-terminal at top of stack is the *rightmost*
  - We are simulating a **rightmost derivation** (in reverse)

Example

- See Example 7.6 on p. 271
  - Given the CFG

    $S \rightarrow S + T \mid T$

    $T \rightarrow T * a \mid a$
  - How will a bottom-up PDA process the input string $a + a * a$?
  - Rightmost derivation:

    $S \Rightarrow S + T \Rightarrow S + T * a \Rightarrow S + a * a \Rightarrow T + a * a \Rightarrow a + a * a$

Move	Production	Stack	Unread input
-		Z_0	$a+a*a$
Shift		aZ_0	$+a*a$
Reduce	T → a	TZ_0	$+a*a$
Reduce	S → T	SZ_0	$+a*a$
Shift		$+SZ_0$	$a*a$
Shift		$a+SZ_0$	$*a$
Reduce	T → a	$T+SZ_0$	$*a$
Shift		$*T+SZ_0$	a
Shift		$a*T+SZ_0$	-
Reduce	T → T * a	$T+SZ_0$	-
Reduce	S → S + T	SZ_0	-
(pop S)		Z_0	-
(accept)			

CFG for a PDA

- We can construct a CFG for a given PDA
- Procedure outlined in the book
  - pp. 273-280
- Approach based on "acceptance by empty stack" mentioned earlier

Conclusion

- We discussed today
  - Acceptance by PDA
  - Non-determinism in PDA
  - PDA for a given CFG
    - Top-down approach
    - Bottom-up approach