COMPSCI/SFWRENG 2FA3

Discrete Mathematics with Applications II

Winter 2020

5 Push-Down Automata and Context-Free Languages

William M. Farmer

Department of Computing and Software
McMaster University

March 19, 2020

McMaster University

Regular Languages

- Regular languages are very special sets of strings that can be specified by:
  1. DFAs,
  2. NFAs,
  3. NFAs with ϵ-transitions,
  4. Regular expressions, and.
  5. Regular grammars.
- Most sets of strings are not regular.
- Examples of nonregular languages:
  1. $\{a^n b^n \mid n \geq 0\}$.
  2. $\{x \in \Sigma^* \mid x \text{ is a palindrome}\}$.
  3. $\{x \in \{[,]\}^* \mid \text{parentheses are balanced in } x\}$.

Context-Free Languages

- Context-free languages are specified by:
  1. Context-free grammars.
  2. Nondeterministic push-down automata.
- Context-free languages include the three examples of nonregular languages on the previous slide as well as most programming languages.
- Most sets of strings are not context-free.
- Example of a non-context-free language:

  $\{a^n b^n c^n \mid n \geq 0\}$.

Example 1: Grammar for English

- Suppose we want to develop a context-free grammar for English. We could start with the following rules:

  $\langle \text{sentence} \rangle \to \langle \text{noun phrase} \rangle \langle \text{verb phrase} \rangle$
  $\langle \text{noun phrase} \rangle \to \langle \text{adjective} \rangle \langle \text{noun phrase} \rangle$
  $\langle \text{noun phrase} \rangle \to \langle \text{noun} \rangle$
  $\langle \text{noun} \rangle \to \texttt{cat}$
  $\langle \text{adjective} \rangle \to \texttt{black}$
  $\langle \text{verb phrase} \rangle \to \langle \text{adverb} \rangle \langle \text{verb phrase} \rangle$
  $\langle \text{verb phrase} \rangle \to \langle \text{verb} \rangle$
  $\langle \text{verb} \rangle \to \texttt{sleeps}$
  $\langle \text{adverb} \rangle \to \texttt{quietly}$

- Then the string

  `black cat quietly sleeps`

  would be produced by this grammar.

Grammar for English (iClicker)

Can a grammar of this kind be constructed for the English language?

1. Yes.
2. No.

Example 2: BNF [1/3]

This is a context-free grammar for a Pascal-like programming language given in Backus-Naur form (BNF):

⟨stmt⟩ ::= ⟨if-stmt⟩ | ⟨while-stmt⟩ | ⟨begin-stmt⟩ | ⟨assg-stmt⟩

⟨if-stmt⟩ ::= `if` ⟨bool-expr⟩ `then` ⟨stmt⟩ `else` ⟨stmt⟩

⟨while-stmt⟩ ::= `while` ⟨bool-expr⟩ `do` ⟨stmt⟩

⟨begin-stmt⟩ ::= `begin` ⟨stmt-list⟩ `end`

⟨stmt-list⟩ ::= ⟨stmt⟩ | ⟨stmt⟩ ; ⟨stmt-list⟩

⟨assg-stmt⟩ ::= ⟨var⟩ `:=` ⟨arith-expr⟩

⟨bool-expr⟩ ::= ⟨arith-expr⟩⟨compare-op⟩⟨arith-expr⟩

⟨compare-op⟩ ::= < | > | <= | => | = | !=

⟨arith-expr⟩ ::= ⟨var⟩ | ⟨const⟩ | (⟨arith-expr⟩⟨arith-op⟩⟨arith-expr⟩)

⟨arith-op⟩ ::= + | - | * | /

⟨const⟩ ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

⟨var⟩ ::= a | b | c | ⋯ | x | y | z

Example 2: BNF [2/3]

- The context-free grammar consists of a finite set of production rules.
  - The rules can be recursive.
  - Alternatives are separated by vertical bars (|).
- The expressions of the form $\langle x \rangle$ are called nonterminal symbols; the other symbols are called terminal symbols.
  - Each nonterminal generates a set of strings over a finite alphabet Σ.
- A derivation of a string w is a sequence of expressions starting with $\langle \text{stmt} \rangle$ and ending with w containing no nonterminals.
  - Each expression in the sequence is obtained from the previous expression by replacing one nonterminal $\langle x \rangle$ with one of the alternatives on the right side of the production for $\langle x \rangle$.

Example 2: BNF [3/3]

- The following is a derivation of `x := 5`:

  ⟨stmt⟩
  ⟨assg-stmt⟩
  ⟨var⟩ `:=` ⟨arith-expr⟩
  `x :=` ⟨arith-expr⟩
  `x :=` ⟨const⟩
  `x := 5`

- The language of this grammar is the set of all strings generated by the grammar.

BNFs (iClicker)

A BNF defines the _____ of a programming language.

A. Parser.
B. Syntax.
C. Semantics.
D. Syntax and semantics.

Context-Free Grammars [1/3]

- A context-free grammar (CFG) is a tuple $G = (N, \Sigma, P, S)$ where:
  1. N is a finite set of symbols called nonterminal symbols.
  2. Σ is a finite set of symbols called terminal symbols.
  3. N and Σ are disjoint.
  4. P is a finite subset of $N \times (N \cup \Sigma)^*$ called productions.
  5. $S \in N$ is the start symbol.
- $A, B, C, \ldots$ are syntactic variables ranging over nonterminals.
- $a, b, c, \ldots$ are syntactic variables ranging over terminals.
- $\alpha, \beta, \gamma, \ldots$ are syntactic variables ranging over strings in $(N \cup \Sigma)^*$.
- A production (A, α) is written as $A \to \alpha$.
- Productions $A \to \alpha_1$, $A \to \alpha_2$, $A \to \alpha_3$ may be written succinctly as $A \to \alpha_1 \mid \alpha_2 \mid \alpha_3$.

Context-Free Grammars (iClicker)

Context-free grammars can have productions of the form

$$aAb \to \alpha?$$

A. Yes.

B. No.

Context-Free Grammars [2/3]

- β is derivable from α in one step in G, written $\alpha \xrightarrow[G]{1} \beta$, if β can be obtained from α by replacing some occurrence of a nonterminal A in α with γ where $A \to \gamma$ is in P.
- $\alpha \xrightarrow[G]{n} \beta$ and $\alpha \xrightarrow[G]{*} \beta$ are defined by:
  1. $\alpha \xrightarrow[G]{0} \alpha$ for any α.
  2. $\alpha \xrightarrow[G]{n+1} \beta$ if there is a γ such that $\alpha \xrightarrow[G]{n} \gamma$ and $\gamma \xrightarrow[G]{1} \beta$.
  3. $\alpha \xrightarrow[G]{*} \beta$ if $\alpha \xrightarrow[G]{n} \beta$ for some $n \geq 0$.
- $\alpha \xrightarrow[G]{*} \beta$ is the reflexive transitive closure of $\alpha \xrightarrow[G]{1} \beta$.
- A sentential form is a string derivable from the start symbol, while a sentence is a sentential form that consists of only terminal symbols.

Context-Free Grammars [3/3]

- The language generated by G, written $L(G)$, is the set of all strings $x \in \Sigma^*$ such that $S \xrightarrow[G]{*} x$.
- $A \subseteq \Sigma^*$ is a context-free language (CFL) if $A = L(G)$ for some context-free grammar G.
- Two grammars G_1 and G_2 are equivalent if $L(G_1) = L(G_2)$.
- Context-free grammars and languages were introduced by the linguist Noam Chomsky (1928–) in the late 1950s.

Example 3: $\{a^n b^n \mid n \geq 0\}$

- Let $G = (N, \Sigma, P, S)$ where:
  1. $N = \{S\}$.
  2. $\Sigma = \{a, b\}$.
  3. $P = \{S \to aSb, S \to \epsilon\}$.
- Proposition 1. $L(G) = \{a^n b^n \mid n \geq 0\}$.
- Corollary 1. $\{a^n b^n \mid n \geq 0\}$ is a CFL.

Example 4: Palindromes

- Let $G = (N, \Sigma, P, S)$ where:
  1. $N = \{S\}$.
  2. $\Sigma = \{a, b\}$.
  3. P is the set of productions
     $$S \to aSa \mid bSb \mid a \mid b \mid \epsilon.$$
- Proposition 2. $L(G) = \{x \in \Sigma^* \mid x \text{ is a palindrome}\}$.
- Corollary 2. $\{x \in \Sigma^* \mid x \text{ is a palindrome}\}$ is a CFL.

Admin — March 10

- Midterm Test 2 will be held on Wednesday, March 11, at 7:00–9:00 PM in MDCL 1305.
  - Same format as Midterm Test 1.
  - Will cover the first four topics.
  - The lecture on March 11 will be a review session.
  - TA review today at 4:30 in BSB 106.
  - A sample test is on Avenue; solutions will be posted tonight.
- Penalties:
  - 15% penalty for incorrect or missing student numbers or version numbers.
  - 5% penalty for incomplete erasures.
- Office hours: To see me please send me a note with times.
- Are there any questions?

Midterm Course Review: Changes to the Course

1. Bonus marks will be awarded to the class according to the level of participation in the online course evaluation.
2. Exercises will be posted with two solutions, but the full solutions will be posted after the assignment is marked.
3. During discussion sessions, a student's name will be read a second time if the student doesn't respond.
4. Tutorials will start with questions from the students about the week's exercises.
5. There will be an exercises channel on Discord.
6. I will explain assignment solutions in the lectures.

Assignment 5 Solutions

Question 1. Construct a deterministic finite automaton M for the alphabet $\Sigma = \{a\}$ such that $L(M)$ is the set of all strings in Σ^* whose length is divisible by either 2 or 5. Present M as a transition diagram.

Question 2. Construct a deterministic finite automaton M for the alphabet $\Sigma = \{0, 1\}$ such that $L(M)$ is the set of all strings x in Σ^* for which $\#0(x)$ is divisible by 2 and $\#1(x)$ is divisible by 3. Present M as a transition diagram.

Assignment 7

Question 1. Let Σ be the set of lower and uppercase letters in the Roman alphabet plus the hyphen symbol "-". Let a *simple English word* be a nonempty string $x \in \Sigma^*$ such that:

1. An uppercase letter can only occur as the first symbol in x.
2. A hyphen cannot occur as either the first or the last symbol in x.
3. Two hyphens cannot occur in a row.

Write a regular expression that matches the set of simple English words (as defined above).

Question 2. Construct an NFA (possibly with ϵ-transitions) that is equivalent to the regular expression $01((10)^* + 11)^*$. Present the NFA as both a transition table (with a column for ϵ transitions) and a transition diagram.

Extra Credit Assignment 4

Design a program that takes a regular expression α and a text file F as input and returns as output all the strings in F that match α. Your design should include the following:

1. A language for writing the regular expressions that are inputted into the program.
2. An algorithm A for compiling the regular expression into an equivalent NFA (possibly with ϵ-transitions).
3. An algorithm B for simulating the NFA on a string.
4. An algorithm C with B as a parameter for finding all the strings in the file that match the regular expression.
5. One or more methods for presenting the output to the user.

Present your design as a 1–2 page paper appended to the end of this assignment.

Review

- Regular languages.
- Context-free languages.
- Grammar for English example.
- BNF example.
- Context-free grammars.

Example 5: $\{x \in \{a, b\}^+ \mid \#a(x) = \#b(x)\}$ [1/2]

- Problem: Find a CFG whose language is
$$\{x \in \{a, b\}^+ \mid \#a(x) = \#b(x)\}.$$
- Let $G = (N, \{a, b\}, P, S)$. What should N and P be?

Example 5: $\{x \in \{a, b\}^+ \mid \#a(x) = \#b(x)\}$ [1/2]

- Problem: Find a CFG whose language is
  $$\{x \in \{a, b\}^+ \mid \#a(x) = \#b(x)\}.$$
- Let $G = (N, \{a, b\}, P, S)$. What should N and P be?
- Productions in P:
  $$S \to aB$$
  $$S \to bA$$

Example 5: $\{x \in \{a, b\}^+ \mid \#a(x) = \#b(x)\}$ [1/2]

- Problem: Find a CFG whose language is
  $$\{x \in \{a, b\}^+ \mid \#a(x) = \#b(x)\}.$$
- Let $G = (N, \{a, b\}, P, S)$. What should N and P be?
- Productions in P:
  $$S \to aB$$
  $$S \to bA$$
  $$A \to a$$
  $$A \to aS$$
  $$A \to bAA$$

Example 5: $\{x \in \{a, b\}^+ \mid \#a(x) = \#b(x)\}$ [1/2]

- Problem: Find a CFG whose language is
  $$\{x \in \{a, b\}^+ \mid \#a(x) = \#b(x)\}.$$
- Let $G = (N, \{a, b\}, P, S)$. What should N and P be?
- Productions in P:
  $$S \to aB$$
  $$S \to bA$$
  $$A \to a$$
  $$A \to aS$$
  $$A \to bAA$$
  $$B \to b$$
  $$B \to bS$$
  $$B \to aBB$$

  So $N = \{S, A, B\}$.

Example 5: $\{x \in \{a, b\}^+ \mid \#a(x) = \#b(x)\}$ [2/2]

- For $x \in \{a, b\}^+$ define:
  1. $P_S(x)$ to be $S \xrightarrow[G]{*} x$ iff $\#a(x) = \#b(x)$.
  2. $P_A(x)$ to be $A \xrightarrow[G]{*} x$ iff $\#a(x) = \#b(x) + 1$.
  3. $P_B(x)$ to be $B \xrightarrow[G]{*} x$ iff $\#b(x) = \#a(x) + 1$.
- Theorem 2. $\forall\, x \in \{a, b\}^+ \,.\, P_S(x)$.

  Proof by induction on $|x|$ fails!
- Corollary 3. $L(G) = \{x \in \{a, b\}^+ \mid \#a(x) = \#b(x)\}$.

  Proof. Immediate.
- Lemma 1. $\forall\, x \in \{a, b\}^+ \,.\, (P_S(x) \wedge P_A(x) \wedge P_B(x)))$.

  Proof. By weak induction on $|x|$.
- Proof of Theorem 2. Immediate from Lemma 1.

Example 5 (iClicker)

Do you understand the need for strengthening the induction hypothesis to prove that

$$L(G) = \{x \in \{a, b\}^+ \mid \#a(x) = \#b(x)\}?$$

A. Yes.

B. No.

Proof of Lemma 1 [1/2]

- Let $P(n) \equiv \forall\, x \in \{a, b\}^n \,.\, (P_S(x) \wedge P_A(x) \wedge P_B(x))$.
- Lemma 1 is then equivalent to:

  Lemma 2. $\forall\, n \geq 1 \,.\, P(n)$.

  Proof. We will prove Lemma 2 by weak induction.

  Base case: $n = 1$. We must prove $P(1)$.

  Subcase BC.1: $x = a$. We must show $P_S(a)$, $P_A(a)$, and $P_B(a)$. $P_S(a)$ and $P_B(a)$ hold since a is derivable from neither S nor B. $P_A(a)$ holds since $A \xrightarrow[G]{1} a$.

  Subcase BC.2: $x = b$. Similar to Subcase BC.1.

Proof of Lemma 1 [2/2]

Induction step: $n \geq 1$. Assume $P(n)$. We must prove $P(n+1)$.

Subcase IS.1: $x = ay$ with $|y| = n$.

Part IS.1.1. Show $P_S(x)$. ($\Rightarrow$) Assume $S \xrightarrow[G]{*} x$. Then the derivation must begin with $S \xrightarrow[G]{1} aB$ and end with $B \xrightarrow[G]{*} y$. By the induction hypothesis, $\#b(y) = \#a(y) + 1$, and so $\#a(x) = \#b(x)$,
($\Leftarrow$) Assume $\#a(x) = \#b(x)$. Then $\#b(y) = \#a(y) + 1$. By the induction hypothesis, $B \xrightarrow[G]{*} y$, and so $S \xrightarrow[G]{1} aB \xrightarrow[G]{*} ay = x$.

Part IS.1.2. Show $P_A(x)$. Similar to Part IS.1.1.

Part IS.1.3. Show $P_B(x)$. Similar to Part IS.1.1.

Subcase IS.2: $x = by$ with $|y| = n$. Similar to Subcase IS.1.

Simplification of CFGs

- Let $G = (N, \Sigma, P, S)$ is a CFG.
- A nonterminal $A \in N$ is useful if there is a derivation $S \xrightarrow[G]{*} \alpha A \beta \xrightarrow[G]{*} x$ for some $x \in \Sigma^*$.
- An ϵ-production is a production of the form $A \to \epsilon$.
- A unit production is a production of the form $A \to B$.
- Theorem 3. Every CFL not containing ϵ is generated by some CFG that has no useless nonterminals, ϵ-productions, or unit productions.

Chomsky Normal Form

- A CFG is in Chomsky normal form (CNF) if all its productions have the form

  $$A \to BC \text{ or } A \to a.$$

- Theorem 4. Every CFL not containing ϵ is generated by some CFG in Chomsky normal form.

  Proof. See Lecture 21 in D. Kozen, Automata and Computability.

Example 6: Chomsky Normal Form

- Consider the CFG $G = (\{S, A, B\}, \{a, b\}, P, S)$ given in Example 5 where P contains the following productions:

  $$S \to aB \mid bA$$
  $$A \to bAA \mid aS \mid a$$
  $$B \to aBB \mid bS \mid b$$

- Find an equivalent grammar in CNF.

Greibach Normal Form

- A CFG is in Greibach normal form (GNF) if all its productions have the form

  $$A \to aB_1, \ldots, B_k \text{ where } k \geq 0.$$

- The GNF was introduced by Sheila Greibach (1939-present).
- Theorem 5. Every CFL not containing ϵ is generated by some CFG in Greibach normal form.

  Proof. See Lecture 21 in D. Kozen, Automata and Computability.

Admin — March 13

- Midterm Test 2.
  - Two questions are being removed from the test:
    - Question about BNFs.
    - Question about number of accessible states (9, not 8).
  - The solutions will be posted soon on Avenue.
  - The marks with posted on Avenue early next week.
- A 2FA3 notetaker is still needed for SAS students.
  - For more information and to register, see https://sas.mcmaster.ca/volunteer-notetaking/
  - Send questions to `sasnotes@mcmaster.ca`.
- Office hours: To see me please send me a note with times.
- Are there any questions?

Linear and Regular Grammars

- A CFG is linear if all its productions have at most one nonterminal on the right-hand side.
- A linear language is a language generated by some linear CFG.
- A CFG is right linear if all its productions have the form

  $$A \to xB \text{ or } A \to x$$

- A CFG is left linear if all its productions have the form

  $$A \to Bx \text{ or } A \to x$$

- A CFG is regular if it is right linear or left linear.
- Theorem 6. A language is regular iff it is generated by a regular CFG.
- Theorem 7. There are linear languages that are not regular.

Pumping Lemma for Context-Free Languages

- The pumping lemma for regular languages says that every sufficiently long string in a regular language contains a short substring that can be "pumped".
- The pumping lemma for CFLs says that every sufficiently long string in a CFL can be divided into five segments such that the middle three segments are short and the second and fourth can be simultaneously "pumped".
- Lemma 5 (Pumping Lemma for CFGs). Let L be a CFL. Then there is a constant n_L such that, if z is in L with $|z| \geq n_L$, then $z = uvwxy$ where
  1. $|vx| \geq 1$,
  2. $|vwx| \leq n_L$, and
  3. for all $i \geq 0$, $uv^i wx^i y$ is in L.

  Proof. See the proof given on pp. 150–3 of D. Kozen, Automata and Computability.

Example 7: $\{a^n b^n c^n \mid n \geq 0\}$

- $L = \{a^n b^n c^n \mid n \geq 0\}$.
- We will use the pumping lemma to show that L is not a CFL.
  1. Suppose L is a CFL.
  2. Let n_L be the constant of Lemma 5 and $z = a^{n_L} b^{n_L} c^{n_L}$.
  3. Then $|z| \geq n_L$ and so $z = uvwxy$ where u, v, w, x, y satisfy the three properties of Lemma 5.
  4. Since $|vwx| \leq n_L$, vwx can only contain instances of at most two of the three symbols.
  5. Hence, by Lemma 5, pumping increases the number of at most two symbols which contradicts the structure of L.

Example 8: Balanced Parentheses [1/4]

- Let PAREN =

$$\{x \in \{[,]\}^* \mid \text{parentheses are balanced in } x\}.$$

- Let $G = (\{S\}, \{[,]\}, P, S)$ be the CFG where P contains the following productions:

$$S \to [S] \mid SS \mid \epsilon.$$

- We want to show $L(G) =$ PAREN?
- First, we must formalize what "parentheses are balanced" means.
- $x \in \{[,], S\}^*$ is balanced iff
  B1. $L(x) = R(x)$ and
  B2. for all prefixes y of x, $L(y) \geq R(y)$
  where $L(x) = \#[(x)$ and $R(x) = \#](x)$.
- So now we want to prove
  $L(G) = \{x \in \{[,]\} \mid x \text{ satisfies } B1 \text{ and } B2\}$.

Example 8: Balanced Parentheses [2/4]

- Lemma 3. $S \xrightarrow[G]{*} \alpha$ implies α satisfies B1 and B2.

  Proof. We will prove Lemma 3 by weak induction on the length of the derivation of α.

  Base case: $S \xrightarrow[G]{0} \alpha$. Then $\alpha = S$ and so α satisfies B1 and B2 trivially.

  Induction step: $S \xrightarrow[G]{n+1} \alpha$. Then $S \xrightarrow[G]{n} \beta \xrightarrow[G]{1} \alpha$ and, by the induction hypothesis, β satisfies B1 and B2. There are three possibilities for the last production.

  Case 1: $S \to SS$. Easy since number and order of parentheses are not changed.

  Case 2: $S \to \epsilon$. Easy since number and order of parentheses are not changed.

Example 8: Balanced Parentheses [3/4]

Case 3: $S \to [S]$. Then $\beta = \beta_1 S \beta_2$ and $\alpha = \beta_1 [S] \beta_2$. $L(\alpha) = L(\beta) + 1 = R(\beta) + 1 = R(\alpha)$ since $L(\beta) = R(\beta)$ by the induction hypothesis. Hence α satisfies B1. Let γ be a prefix of α. We need to show that $L(\gamma) \geq R(\gamma)$ to prove that α satisfies B2. There are three cases:

- Case 3.1: γ is a prefix of β_1 and thus of β. So γ satisfies B2 by the induction hypothesis.
- Case 3.2: γ is a prefix of $\beta[S$ but not of β. Then $L(\gamma) = L(\beta_1) + 1 \geq R(\beta_1) + 1 > R(\beta_1) = R(\gamma)$ since $L(\beta_1) \geq R(\beta_1)$ by the induction hypothesis.
- Case 3.3: $\gamma = \beta[S]\delta$ where δ is a prefix of β_2. Then $L(\gamma) = L(\beta_1 S \delta) + 1 \geq R(\beta_1 S \delta) + 1 = R(\gamma)$ since $L(\beta_1 S \delta) \geq R(\beta_1 S \delta)$ by the induction hypothesis.

Hence α satisfies B2.

Example 8: Balanced Parentheses [4/4]

- Lemma 4. x satisfies B1 and B2 implies $S \xrightarrow[G]{*} x$.

  Proof. By strong induction on $|x|$. See the proof given on pp. 138–9 of D. Kozen, Automata and Computability.
- Theorem 8. $L(G) = \{x \in \{[,]\} \mid x \text{ satisfies } B1 \text{ and } B2\}$.

  Proof. Follows immediately from Lemmas 3 and 4.

Context-Sensitive Grammars

- A context-sensitive grammar (CSG) is a tuple $G = (N, \Sigma, P, S)$ where:
  1. N is a finite set of symbols called nonterminal symbols.
  2. Σ is a finite set of symbols called terminal symbols.
  3. N and Σ are disjoint.
  4. P is a finite set of productions of the form
     $$\alpha A \beta \to \alpha \gamma \beta$$
     where $A \in N$, $\alpha, \beta \in (N \cup \Sigma)^*$, and $\gamma \in (N \cup \Sigma)^+$.
  5. $S \in N$ is the start symbol.
- The production $S \to \epsilon$ is also allowed if S does not appear on the right side of any production.
- $A \subseteq \Sigma^*$ is a context-sensitive language (CSL) if A is generated by some context-sensitive grammar.
- Example of a CSL that is not a CFL: $\{a^n b^n c^n \mid n \geq 0\}$.

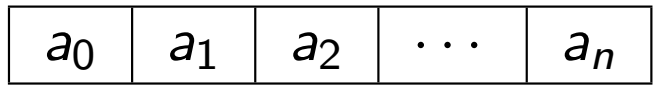
A Stack as a Value

- Abstractly, a stack can be viewed as a value in an inductive set Stack defined by the following constructors:
  1. bottom : Stack.
  2. push : Stack $\times$ Element $\rightarrow$ Stack.
- The accessors of the Stack are:
  1. pop : Stack $\rightarrow$ Stack.
     $\mathsf{pop}\,(\mathsf{push}(s, e) = s$.
  2. top : Stack $\rightarrow$ Element.
     $\mathsf{top}\,(\mathsf{push}(s, e) = e$.
- Note: Access is done in a last in, first out (LIFO) fashion.

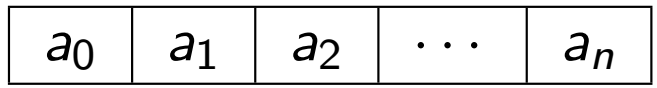
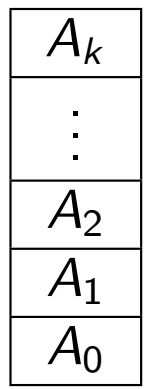
A Stack as a Machine

- Abstractly, a stack can be viewed as a machine consisting of a finite collection S of values of type Element and the following three operations:
  1. push takes an x : Element as input and adds x to S.
  2. pop removes from S the most recently added element.
  3. top returns as output the most recently added element.
- Note: Access is done in a last in, first out (LIFO) fashion.

An NFA as a Machine

- An NFA has the following components:
  1. Input tape: 
  2. Memory:
     a. State: $\boxed{\text{q}} \in Q$.
  3. Program: $\boxed{\Delta}$, a transition function.
- The tape is read left to right and is read only.
- The memory is finite.
- The program is nondeterministic.

An NPDA as a Machine

- A nondeterministic push-down automaton (NPDA) has the following components:
  1. Input tape: 
  2. Memory:
     a. State: $\boxed{\text{q}} \in Q$.
     b. Stack: 
  3. Program: $\boxed{\delta}$, a transition relation.
- The tape is read left to right and is read only.
- The memory is unbounded but accessed as a stack.
- The program is nondeterministic.

Nondeterministic Push-Down Automata [1/4]

- A nondeterministic push-down automaton (NPDA) is a tuple $M = (Q, \Sigma, \Gamma, \delta, s, \bot, F)$ where:
  1. Q is a finite set of elements called states.
  2. Σ is a finite set of symbols called the input alphabet.
  3. Γ is a finite set of symbols called the stack alphabet.
  4. $\delta \subseteq (Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma) \times (Q \times \Gamma^*)$ is the transition relation.
  5. $s \in Q$ is the start state.
  6. $\bot \in \Gamma$ is the initial stack symbol.
  7. $F \subseteq Q$ is the set of final states.

Nondeterministic Push-Down Automata [2/4]

- $((p, a, A), (q, B_1 B_2 \cdots B_k)) \in \delta$ means that, if the NPDA is in state p reading input a on the input tape and A is on the top of the stack, then the machine can pop A off the stack, push $B_1 B_2 \cdots B_k$ onto the stack, move its read head right one cell past a, and enter state q.
- $((p, \epsilon, A), (q, B_1 B_2 \cdots B_k)) \in \delta$ means that, if the NPDA is in state p and A is on the top of the stack, then the machine can pop A off the stack, push $B_1 B_2 \cdots B_k$ onto the stack, not move its read head, and enter state q.
- A configuration of an NPDA M is a member of

  $$Q \times \Sigma^* \times \Gamma^*$$

  that describes M's current state, unread input string, and stack contents.
- The start configuration on input x is $(s, x, \bot)$.

Nondeterministic Push-Down Automata [3/4]

- The next configuration relation $C \xrightarrow[M]{1} D$ is defined by:
  1. If $((p, a, A), (q, \gamma)) \in \delta$, then, for all $y \in \Sigma^*, \beta \in \Gamma^*$,
     $$(p, ay, A\beta) \xrightarrow[M]{1} (q, y, \gamma\beta).$$
  2. If $((p, \epsilon, A), (q, \gamma)) \in \delta$, then, for all $y \in \Sigma^*, \beta \in \Gamma^*$,
     $$(p, y, A\beta) \xrightarrow[M]{1} (q, y, \gamma\beta).$$
- $C \xrightarrow[M]{n} D$ and $C \xrightarrow[M]{*} D$ are defined by:
  1. $C \xrightarrow[M]{0} C$.
  2. $C \xrightarrow[M]{n+1} D$ if there is some E such that $C \xrightarrow[M]{n} E \xrightarrow[M]{1} D$.
  3. $C \xrightarrow[M]{*} D$ if there is some $n \geq 0$ such that $C \xrightarrow[M]{n} D$.
- $C \xrightarrow[M]{*} D$ is the reflexive transitive closure of $C \xrightarrow[M]{1} D$.

Nondeterministic Push-Down Automata [4/4]

- There are two definitions of acceptance:
  1. M accepts x by final state if
     $$(s, x, \bot) \xrightarrow[M]{*} (q, \epsilon, \gamma)$$
     for some $q \in F$ and $\gamma \in \Gamma^*$. Let $L_{fs}(M)$ be the language M accepts by final state.
  2. M accepts x by empty stack if
     $$(s, x, \bot) \xrightarrow[M]{*} (q, \epsilon, \epsilon)$$
     for some $q \in Q$. Let $L_{es}(M)$ be the language M accepts by empty stack.
- Theorem 9. Let $L \subseteq \Sigma*$. Then $L = L_{fs}(M_1)$ for some NPDA M_1 iff $L = L_{es}(M_2)$ for some NPDA M_2
  Proof. See Lecture E of D. Kozen, Automata and Computability.

Example 9 [1/2]

- Let $M = (Q, \Sigma, \Gamma, \delta, s, \bot, \emptyset)$ be the NPDA where:

  $Q = \{s\}$.
  $\Sigma = \{\texttt{[}, \texttt{]}\}$.
  $\Gamma = \{\bot, \texttt{[}\}$.
  δ contains the following transitions:

  1. $((s, \texttt{[}, \bot), (s, \texttt{[}\bot))$.
  2. $((s, \texttt{[}, \texttt{[}), (s, \texttt{[[}))$.
  3. $((s, \texttt{]}, \texttt{[}), (s, \epsilon))$.
  4. $((s, \epsilon, \bot), (s, \epsilon))$.

- M accepts by empty stack the set of balanced strings of parentheses in Σ^*.

Example 9 [2/2]

- M accepts ϵ by empty stack:

	$(s,$	$\epsilon,$	$\bot)$	start configuration
$\rightarrow$	$(s,$	$\epsilon,$	$\epsilon)$	$((s, \epsilon, \bot),\ (s, \epsilon))$

- M accepts `[[][]]` by empty stack:

	$(s,$	`[[][]],`	$\bot)$	start configuration
$\rightarrow$	$(s,$	`[][]],`	`[`$\bot$)	$((s,$ `[`$, \bot),\ (s,$ `[`$\bot))$
$\rightarrow$	$(s,$	`][]],`	`[[`$\bot$)	$((s,$ `[`$,$ `[`$),\ (s,$ `[[`$)$
$\rightarrow$	$(s,$	`[]],`	`[`$\bot$)	$((s,$ `]`$,$ `[`$),\ (s, \epsilon))$
$\rightarrow$	$(s,$	`]],`	`[[`$\bot$)	$((s,$ `[`$,$ `[`$),\ (s,$ `[[`$)$
$\rightarrow$	$(s,$	`],`	`[`$\bot$)	$((s,$ `]`$,$ `[`$),\ (s, \epsilon))$
$\rightarrow$	$(s,$	$\epsilon,$	$\bot)$	$((s,$ `]`$,$ `[`$),\ (s, \epsilon))$
$\rightarrow$	$(s,$	$\epsilon,$	$\epsilon)$	$((s, \epsilon, \bot),\ (s, \epsilon))$

Example 9 (iClicker)

In what configuration will M get stuck?

A. $(s, [x, \epsilon)$.

B. $(s,]x, \epsilon)$.

C. $(s,]x, \bot)$.

D. $(s,]x, [\gamma)$.

Example 9 (iClicker)

In what configuration will M get stuck?

A. $(s, [x, \epsilon)$.

B. $(s,]x, \epsilon)$.

C. $(s,]x, \bot)$.

D. $(s,]x, [\gamma)$.

Example 10 [1/2]

- Let $M = (Q, \Sigma, \Gamma, \delta, q_0, \bot, \emptyset)$ be the NPDA where:

  $Q = \{q_0, q_1\}$.
  $\Sigma = \{a, b\}$.
  $\Gamma = \{\bot, a\}$.
  δ contains the following transitions:

  1. $((q_0, \epsilon, \bot),\ (q_0, \epsilon))$.
  2. $((q_0, a, \bot),\ (q_0, a))$.
  3. $((q_0, a, a),\ (q_0, aa))$.
  4. $((q_0, \epsilon, a),\ (q_1, a))$.
  5. $((q_1, b, a),\ (q_1, \epsilon))$.

- M accepts by empty stack $\{a^n b^n \mid n \geq 0\}$.

Example 10 (iClicker)

How many configurations are in a computation of M that accepts $a^n b^n$?

A. $2n$.

B. $2n + 1$.

C. $2n + 2$.

D. $2n + 3$.

Example 10 (iClicker)

How many configurations are in a computation of M that accepts $a^n b^n$?

A. $2n$.

B. $2n + 1$.

C. $\boxed{2n + 2.}$

D. $2n + 3$.

Example 10 [2/2]

- M accepts $a^0 b^0 = \epsilon$ by empty stack:

	$(q_0,$	$\epsilon,$	$\bot)$	start configuration
$\to$	$(q_0,$	$\epsilon,$	$\epsilon)$	$((q_0, \epsilon, \bot),\ (q_0, \epsilon))$

- M accepts $a^3 b^3 = aaabbb$ by empty stack:

	$(q_0,$	$aaabbb,$	$\bot)$	start configuration
$\to$	$(q_0,$	$aabbb,$	$a)$	$((q_0, a, \bot),\ (q_0, a))$
$\to$	$(q_0,$	$abbb,$	$aa)$	$((q_0, a, a),\ (q_0, aa))$
$\to$	$(q_0,$	$bbb,$	$aaa)$	$((q_0, a, a),\ (q_0, aa))$
$\to$	$(q_1,$	$bbb,$	$aaa)$	$((q_0, \epsilon, a),\ (q_1, a))$
$\to$	$(q_1,$	$bb,$	$aa)$	$((q_1, b, a),\ (q_1, \epsilon))$
$\to$	$(q_1,$	$b,$	$a)$	$((q_1, b, a),\ (q_1, \epsilon))$
$\to$	$(q_1,$	$\epsilon,$	$\epsilon)$	$((q_1, b, a),\ (q_1, \epsilon))$

Equivalence of CFGs and NPDAs

- Theorem 10. Let G be a CFG. Then there is an NPDA M such that $L(G) = L_{es}(M)$.
  Proof. See Lecture 24 in D. Kozen, Automata and Computability. We sketch the proof below.
- Theorem 11. Let M be an NPDA. Then there is a CFG G such that $L(G) = L_{es}(M)$.
  Proof. See Lecture 25 in D. Kozen, Automata and Computability.
- Corollary 4. CFGs generate and NPDAs accept the same class of languages — the class of context-free languages.

Sketch of the Proof of Theorem 10 [1/2]

1. Let $G = (N, \Sigma, P, S)$ be a CFG.
2. We may assume without loss of generality that all the productions of G are of the form

   $$A \to cB_1B_2 \cdots B_k$$

   where $c \in \Sigma \cup \{\epsilon\}$ and $k \geq 0$.
3. We construct from G the NPDA

   $$M = (\{q\}, \Sigma, N, \delta, q, S, \emptyset)$$

   where δ contains

   $$((q, c, A), (q, B_1B_2 \cdots B_k))$$

   for each production $A \to cB_1B_2 \cdots B_k$ in P.

Sketch of the Proof of Theorem 10 [2/2]

4. Lemma 5. For all $z, y \in \Sigma^*, \gamma \in N^*, A \in N$,

   $$A \xrightarrow[G]{n} z\gamma \text{ via a leftmost derivation}$$

   iff

   $$(q, zy, A) \xrightarrow[M]{n} (q, y, \gamma).$$

   Proof. By weak induction on n.

5. Then $L(G) = L_{es}(M)$ follows using Lemma 5 by the following derivation:

$$\begin{aligned} & x \in L(G) \\ & \equiv S \xrightarrow[G]{*} x \text{ by a leftmost derivation} \\ & \equiv (q, x, S) \xrightarrow[M]{*} (q, \epsilon, \epsilon) \\ & \equiv x \in L_{es}(M) \end{aligned}$$

Example 11 [1/2]

- Let $L = \{x \in \{[,]\}^+ \mid \text{parentheses are balanced in } x\}$.
- $G = (\{S\}, \{[,]\}, P, S)$ is a CFG in GNF that accepts L where P contains the following productions:

  $S \to [B \mid [BS \mid [SB \mid [SBS$
  $B \to]$

- Let M be defined as above and so its transitions are:

  $((q, [, S), (q, B))$
  $((q, [, S), (q, BS))$
  $((q, [, S), (q, SB))$
  $((q, [, S), (q, SBS))$
  $((q,], B), (q, \epsilon))$

Example 11 [2/2]

- Here is a leftmost derivation of `[[][]]` in G next to an accepting computation of M on `[[][]]`:

S	$(q,$ `[[][]]`$,$ $S)$
`[`SB	$(q,$ `[][]]`$,$ $SB)$
`[[`BSB	$(q,$ `][]]`$,$ $BSB)$
`[[]`SB	$(q,$ `[]]`$,$ $SB)$
`[[][`BB	$(q,$ `]]`$,$ $BB)$
`[[][]`B	$(q,$ `]`$,$ $B)$
`[[][]]`	(q, ϵ, ϵ)

Closure Properties of Context-Free Languages

Context-free languages are closed under:

1. Union.
   - L_1 and L_2 are context-free implies $L_1 \cup L_2$ is context-free.
2. Concatenation.
   - L_1 and L_2 are context-free implies $L_1 L_2$ is context-free.
3. Asterate.
   - L is context-free implies L^* is context-free.

Context-free languages are not closed under:

1. Complementation.
   - L is context-free does not imply $\sim L$ is context-free.
2. Intersection.
   - L_1 and L_2 are context-free does not imply $L_1 \cap L_2$ is context-free.

Deterministic Push-Down Automata

- A deterministic push-down automaton (DPDA) is similar to a NDPA except that exactly one transition applies to each configuration.
  - See Lecture F in D. Kozen, Automata and Computability.
- A deterministic context-free language (DCFL) is a language accepted by a DPDA.
- Theorem 12. Every DCFL is a CFL, but not every CFL is a DCFL.
- Theorem 13. DCFLs are closed under complement (unlike CFLs).
- Example of a CFL that is not a DCFL:
  $L = \{xx \mid x \in \{a, b\}^*\}$ is not a CFL, but $\sim L$ is a CFL, so $\sim L$ is not a DCFL by Theorem 13.

Parsing

- Let L be a programming language.
- A parser for L is a program that takes a program in L as input and returns a parse tree of the program if the program is syntactically correct and otherwise identifies the syntax error(s) in the program.
  - Thus a parser recognizes what strings are programs in L.
- In a compiler for L, the parser is called after the lexical analyzer finishes.
- If the syntax of L is specified by a CFG, then a parser for L can be implemented as a DPDA which is usually generated automatically from the grammar for L.