Pharos University in Alexandria
Faculty of Computer Science & Artificial Intelligence
Course Title: Theory of Computation
Code: CS 307

Theory of Computation

Lecturer: Sherine Shawky

Text Books

1. Introduction to formal languages and automata, Peter Linz, 6th edition, 2017.

Week 10

Pushdown Automata

Pushdown Automata

- In the discussion of regular languages, we saw that there were several ways of exploring regular languages: finite automata, regular grammars, and regular expressions.
- Having defined context-free languages via context-free grammars, we now ask if there are other options for context-free languages.
- It turns out that there is no analog of regular expressions, but that pushdown automata are the automata associated with context free languages.

Pushdown Automata

- Pushdown automata are essentially finite automata with a stack as storage.
- Since a stack by definition has infinite length, this overcomes the limitation on finite automata arising from a bounded memory.
- Pushdown automata are equivalent to context-free grammars, as long as we allow them to be nondeterministic.
- We can also define deterministic pushdown automata, but the language family associated with them is a proper subset of the context-free languages.

Pushdown Automata

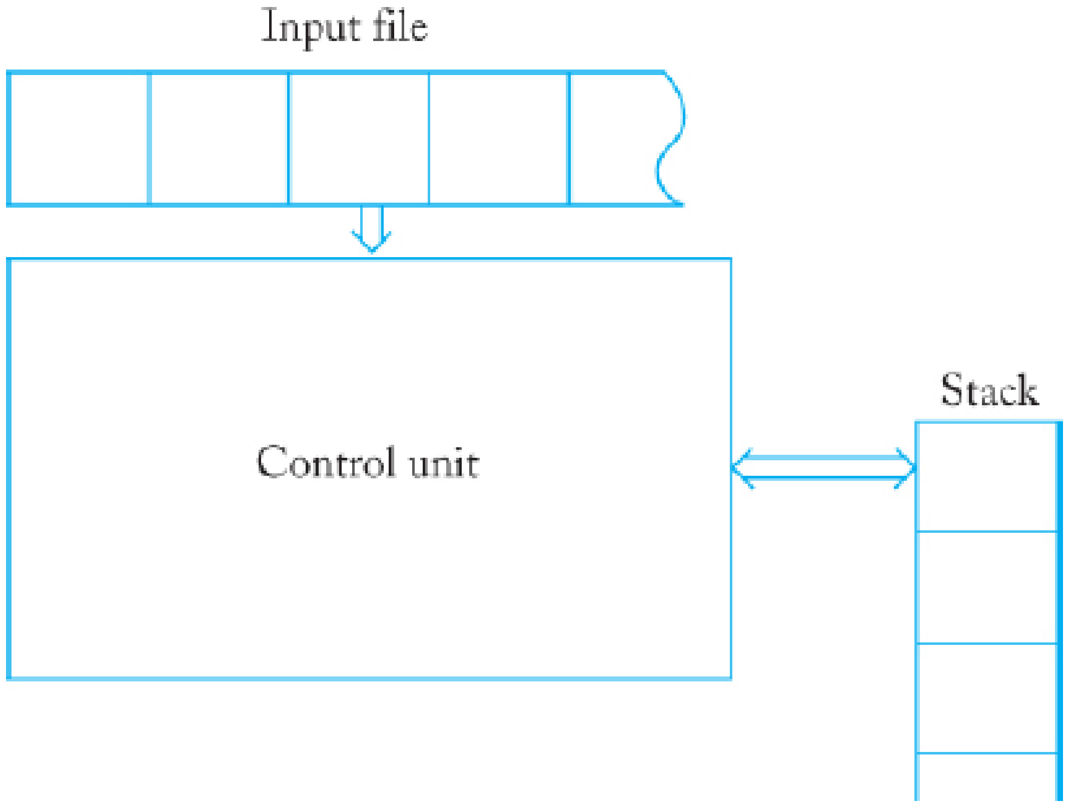
- Context-free language may require storing an unbounded amount of information.
- For example, when scanning a string from the language $L = \{a^n b^n : n \geq 0\}$, we must not only check that all a's precede the first b, we must also count the number of a's.
- Since n is unbounded, this counting cannot be done with a finite memory.
- We want a machine that can count without limit.
- But as we see from other examples, such as $\{ww^R\}$, we need more than unlimited counting ability: We need the ability to store and match a sequence of symbols in reverse order.

Pushdown Automata

- This suggests that we might try a stack as a storage mechanism, allowing unbounded storage that is restricted to operating like a stack. This gives us a class of machines called pushdown automata (PDA).
- The class of deterministic pushdown automata defines a new family of languages, the deterministic context-free languages, forming a proper subset of the context-free languages.
- Since this is an important family for the treatment of programming languages, we conclude the chapter with a brief introduction to the grammars associated with deterministic context-free languages.

NDPA

- A schematic representation of a pushdown automaton is given in the Figure.



NDPA

- Each move of the control unit reads a symbol from the input file, while at the same time changing the contents of the stack through the usual stack operations.
- Each move of the control unit is determined by the current input symbol as well as by the symbol currently on top of the stack.
- The result of the move is a new state of the control unit and a change in the top of the stack.

Definition of a Pushdown Automaton

- A NPDA is defined by the septuple

  $M = (Q, \Sigma, \Gamma, \delta, q0, z, F)$,

  where

  - Q is a finite set of internal states of the control unit,
  - Σ is the input alphabet,
  - Γ is a finite set of symbols called the stack alphabet,
  - $\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow$ set of finite subsets of $Q \times \Gamma^*$ is the transition function,
  - $q0 \in Q$ is the initial state of the control unit,
  - $z \in \Gamma$ is the stack start symbol,
  - $F \subseteq Q$ is the set of final states.

Pushdown Automata

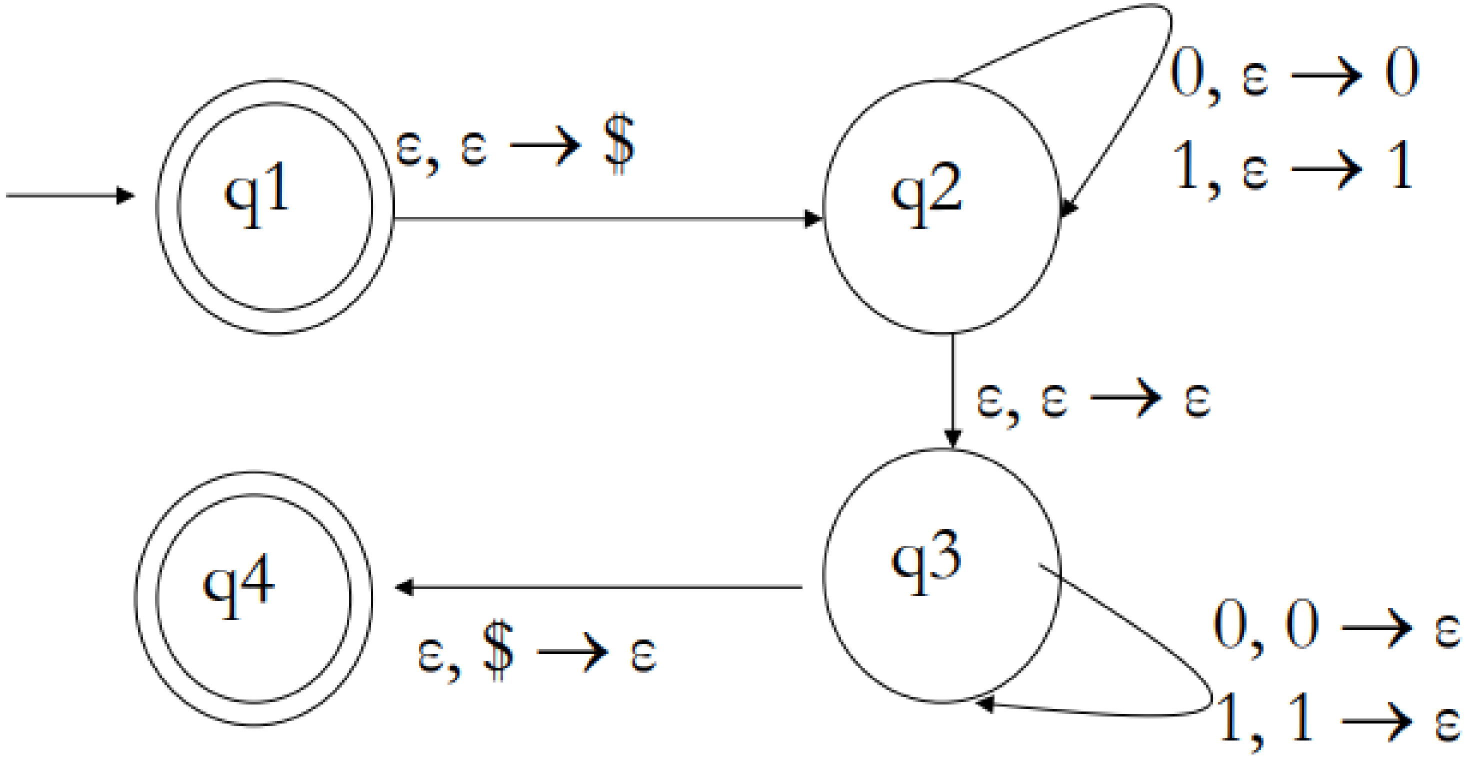
- The complicated formal appearance of the domain and range of δ merits a closer examination.
- The arguments of δ are the current state of the control unit, the current input symbol, and the current symbol on top of the stack.
- The result is a set of pairs (q, x), where q is the next state of the control unit and x is a string that is put on top of the stack in place of the single symbol there before.
- Note that the second argument of δ may be λ, indicating that a move that does not consume an input symbol is possible.
- We will call such a move a λ-transition.
- Note also that δ is defined so that it needs a stack symbol; no move is possible if the stack is empty.
- Finally, the requirement that the elements of the range of δ be a finite subset is necessary because $Q \times \Gamma^*$ is an infinite set and therefore has infinite subsets.
- While an npda may have several choices for its moves, this choice must be restricted to a finite set of possibilities.

Pushdown Automata

- Example: Construct an NPDA for accepting the language $L = \{ww^R : w \in \{a, b\}^+\}$,

Pushdown Automata

- Example: Construct an NPDA for accepting the language $L = \{ww^R : w \in \{a, b\}^+\}$,



Pushdown Automata

- we use the fact that the symbols are retrieved from a stack in the reverse order of their insertion.
- When reading the first part of the string, we push consecutive symbols on the stack.
- For the second part, we compare the current input symbol with the top of the stack, continuing as long as the two match.
- Since symbols are retrieved from the stack in reverse of the order in which they were inserted, a complete match will be achieved if and only if the input is of the form ww^R.

Pushdown Automata

- For the NPDA language

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

  Check strings “abba” and “abab” are accepting in this language or not.

Chomsky Normal Form

- One kind of normal form we can look for is one in which the number of symbols on the right of a production is strictly limited.
- In particular, we can ask that the string on the right of a production consist of no more than two symbols.
- One instance of this is the Chomsky normal form.

Chomsky Normal Form

- DEFINITION 6.4
- A context-free grammar is in Chomsky normal form if all productions are of the form
  - $A \rightarrow BC$
  - or
  - $A \rightarrow a$,

where A, B, C are in V, and a is in T.

Chomsky Normal Form

- Example 1:
- The grammar
  - $S \rightarrow AS|a,$
  - $A \rightarrow SA|b$

  is in Chomsky normal form.

Chomsky Normal Form

- Example 2:
- The grammar
  - $S \rightarrow AS|AAS,$
  - $A \rightarrow SA|aa$

is not; both productions $S \rightarrow AAS$ and $A \rightarrow aa$ violate the conditions of Definition 6.4.

Chomsky Normal Form

- THEOREM 6.6
- Any context-free grammar $G = (V, T, S, P)$ with $\lambda \notin L(G)$ has an equivalent grammar $\hat{G} = (\hat{V}, \hat{T}, S, \hat{P})$ in Chomsky normal form.
- The Chomsky normal form places restrictions on the *length* and the *composition* of the **right-hand side** of a rule.

Chomsky Normal Form

- **Definition 6.4:**
- A CFG is in **Chomsky normal form** if each production rule has one of the following forms:
  - $A \to a$
  - $A \to BC$
  - $S \to \varepsilon$

  where A, B, $C \in$ V and a $\in$L

Algorithm to Convert into CNF

- **Step 1** − If the start symbol **S** occurs on some right side, create a new start symbol **S'** and a new production **S'**→ **S**.
- **Step 2** − Remove Null productions. (Using the Null production removal algorithm discussed earlier)
- **Step 3** − Remove unit productions. (Using the Unit production removal algorithm discussed earlier)
- **Step 4** − Replace each production $\mathbf{A \rightarrow B_1 \ldots B_n}$ where $\mathbf{n > 2}$ with $\mathbf{A \rightarrow B_1 C}$ where $\mathbf{C \rightarrow B_2 \ldots B_n}$. Repeat this step for all productions having two or more symbols in the right side.
- **Step 5** − If the right side of any production is in the form **A** → **aB** where a is a terminal and **A, B** are non-terminal, then the production is replaced by **A** → **XB** and **X** → **a**. Repeat this step for every production which is in the form **A** → **aB**.

Chomsky Normal Form

- **Problem**
- Convert the following CFG into CNF

  $S \to ASA \mid aB,\ A \to B \mid S,\ B \to b \mid \varepsilon$

Chomsky Normal Form

- **Solution**
- **(1)** Since **S** appears in R.H.S, we add a new state $\mathbf{S_0}$ and $\mathbf{S_0 \rightarrow S}$ is added to the production set and it becomes −

  $S_0 \rightarrow S$, $S \rightarrow ASA \mid aB$, $A \rightarrow B \mid S$, $B \rightarrow b \mid \in$
- **(2)** Now we will remove the null productions −

  $B \rightarrow \in$ and $A \rightarrow \in$

  After removing $B \rightarrow \varepsilon$, the production set becomes −

  $S_0 \rightarrow S$, $S \rightarrow ASA \mid aB \mid a$, $A \rightarrow B \mid S \mid \in$, $B \rightarrow b$

  After removing $A \rightarrow \in$, the production set becomes −

  $S_0 \rightarrow S$, $S \rightarrow ASA \mid aB \mid a \mid AS \mid SA \mid S$, $A \rightarrow B \mid S$, $B \rightarrow b$

Chomsky Normal Form

- **(3)** Now we will remove the unit productions.

  After removing S → S, the production set becomes −
  S_0→S, S→ ASA | aB | a | AS | SA, A → B | S, B → b
  After removing S_0→ S, the production set becomes −
  S_0→ ASA | aB | a | AS | SA, S→ ASA | aB | a | AS | SA
  A → B | S, B → b
  After removing A→ B, the production set becomes −
  S_0 → ASA | aB | a | AS | SA, S→ ASA | aB | a | AS | SA
  A → S | b
  B → b
  After removing A→ S, the production set becomes −
  S_0 → ASA | aB | a | AS | SA, S→ ASA | aB | a | AS | SA
  A → b |ASA | aB | a | AS | SA, B → b

Chomsky Normal Form

- **(4)** Now we will find out more than two variables in the R.H.S

  Here, S_0→ ASA, S → ASA, A→ ASA violates two Non-terminals in R.H.S.

  Hence we will apply step 4 and step 5 to get the following final production set which is in CNF −

  S_0→ AX | aB | a | AS | SA

  S→ AX | aB | a | AS | SA

  A → b |AX | aB | a | AS | SA

  B → b

  X → SA

Chomsky Normal Form

- **(5)** We have to change the productions $S_0 \rightarrow aB$, $S \rightarrow aB$, $A \rightarrow aB$

  And the final production set becomes −

  $S_0 \rightarrow AX \mid YB \mid a \mid AS \mid SA$

  $S \rightarrow AX \mid YB \mid a \mid AS \mid SA$

  $A \rightarrow b\ A \rightarrow b \mid AX \mid YB \mid a \mid AS \mid SA$

  $B \rightarrow b$

  $X \rightarrow SA$

  $Y \rightarrow a$