CS 133: Automata and Computability (Hierarchy of Formal Languages)[1]

Henry N. Adorna & Nestine Hope S. Hernandez
Department of Computer Science
(Algorithms and Complexity)
University of the Philippines - Diliman

24 January 2013

[1]This Lecture is based on the book entitled Introduction to Formal Languages and Automata (4ed, 2006) by Peter Linz

Outline of the Talk

Definitions

- Symbols, $a, b, x, y, \ldots$.
- Alphabet Σ : a finite set of symbols.
- Words $w = ababaa$: a sequence of symbols from a given alphabet.
- Σ^*, set of all possible words from Σ.
- Operation on words:
  1. length of a word w: $|w|$, number of symbols(including repeated ones) in a word
  2. concatenation of words w and x : wx, juxtaposition of words; NOTE $wx \neq xw$.
  3. reversal of a word w : w^R, w with symbols in reverse order.
  4. etc. ...

Three Basic Concepts

1. Languages,
   $L \subseteq \Sigma^*$.
2. Grammar,
   $G = (V, T, S, P)$, where
   $V :=$ finite set of variables
   $T :=$ finite set of terminal symbols
   $S \in V :=$ start symbol
   $P :=$ set of productions
   $L(G)$, set of words generated by G.
3. Automata, (accepts or rejects an input word, deterministically or non-deterministically)
   $L(M)$, is the set of strings/words accepted by an automaton M.

Some Applications

1. Formal languages and grammars: used widely in connection with programming languages

$$
\begin{array}{lcl}
< id > & \rightarrow & < letter >< rest > \mid < undrscr >< rest > \\
< rest > & \rightarrow & < letters >< rest > \mid < digit >< rest > \\
 & & \mid < undrscr >< rest > \mid \epsilon \\
< letter > & \rightarrow & a|b|c|\cdots|A|B|\cdots|X|Y|Z \\
< digit > & \rightarrow & 0|1|2|\cdots|9 \\
< undrscr > & \rightarrow & -
\end{array}
$$

2. Another area is digital design, where tranducer concepts are prevalent.
   In principle, any digital computer can be viewed as an automaton: (not necessarily appropriate)

Exercise

1. Let $L = \{ab, aa, baa\}$. Which of the following words are in L^* : $abaabaaabaa$, $aaaabaaaa$, $baaaaabaaaab$, $baaaaabaa$? Which words are in L^4?
2. Find the grammar for $\Sigma = \{a, b\}$ that generate the set of all words with at least three a's. Please provide a convincing reason.
3. What language does the grammar with the following productions below generate?
   $S \rightarrow Aa$
   $A \rightarrow B$
   $B \rightarrow Aa$.

Answer

1. $abaabaaabaa, baaaaabaa \in L^*$, while $aaaabaaaa, baaaaabaa \in L^4$.
2. Required grammar: $S \longrightarrow AaAaAaA$, $A \longrightarrow aA|bA|\epsilon$.
3. $L = \emptyset$.

Regular Languages: Type 3

Definition

Let Σ be a given alphabet. Then

1. $\emptyset, \epsilon$, and $a \in \Sigma$ are all regular expressions. (primitive)
2. If R_1 and R_2 are regular expressions, then so are $R_1 + R_2, R_1 R_2, R_1^*$, and (R_1).
3. A word is a regular expression iff it can be derived via (1) and (2).

Definition

The language $L(R)$ denoted by any regular expression R is defined by the following rules:

1. $\emptyset$ is a regular expression denoting the empty set,
2. ϵ is a regular expression denoting $\{\epsilon\}$,
3. $\forall a \in \Sigma$, a is a regular expression denoting $\{a\}$.
4. If R_1 and R_2 are regular expressions, then
   1. $L(R_1 + R_2) = L(R_1) \cup L(R_2)$
   2. $L(R_1 \cdot R_2) = L(R_1)L(R_2)$
   3. $L((R_1)) = L(R_1)$
   4. $L(R_1^*) = (L(R_1))^*$

Example

If

$$R = (aa)^*(bb)^*b,$$

then

Example

If

$$R = (aa)^*(bb)^*b,$$

then

$$L(R) = \{a^{2n}b^{2m+1} \mid n, m \geq 0\}.$$

Example

If

$$R = (aa)^*(bb)^*b,$$

then

$$L(R) = \{a^{2n}b^{2m+1} \mid n, m \geq 0\}.$$

Example

Let R be a regular expression on some alphabet Σ, such that

$$L(R) = \{w \in \Sigma^* \mid w \text{ has at least one pair of consecutive zeroes}\},$$

then

Example

If

$$R = (aa)^*(bb)^*b,$$

then

$$L(R) = \{a^{2n}b^{2m+1} \mid n, m \geq 0\}.$$

Example

Let R be a regular expression on some alphabet Σ, such that

$$L(R) = \{w \in \Sigma^* \mid w \text{ has at least one pair of consecutive zeroes}\},$$

then

$$R = (0+1)^*00(0+1)^*.$$

Ways of Describing Type 3 Langugages

1. Finite Automaton:

   A language L is **regular** if and only if there exists some (deterministic) finite automaton that accepts all words in L, that is, $L = L(M)$.

2. Regular Expression:

   Let L be a regular language if and only if there exists a regular expression R such that $L = L(R)$.

3. Left- or Right-Linear Grammar (Regular Grammar):

   A grammar $G = (V, T, S, P)$ is right(left)-linear if all productions are of the form

   $$a \rightarrow xB(Bx)$$

   $$A \rightarrow x$$

   The language L is regular if and only if there exists a right(left)-linear grammar G such that $L = L(G)$.

Example

Construct a finite automaton that accepts the language generated by the grammar G with rules

$$V_0 \to aV_1, V_1 \to abV_0|b.$$

what is the language accepted by the automaton which is generated by the grammar G.

Example

Construct a finite automaton that accepts the language generated by the grammar G with rules

$$V_0 \to aV_1, V_1 \to abV_0|b.$$

what is the language accepted by the automaton which is generated by the grammar G.

$$L((aab)^*ab).$$

Example

Construct a right-linear grammar for $L(aab^*a)$.

Example

Construct a right-linear grammar for $L(aab^*a)$.

$$\begin{array}{lll} \delta(q_0, a) = \{q_1\} & - & q_0 \rightarrow aq_1 \\ \delta(q_1, a) = \{q_2\} & - & q_1 \rightarrow aq_2 \\ \delta(q_2, b) = \{q_2\} & - & q_2 \rightarrow bq_2 \\ \delta(q_2, a) = \{q_f\} & - & q_2 \rightarrow q_f \\ q_f \in F & - & q_f \rightarrow \epsilon \end{array}$$

Properties of Type 3 Languages

1. Closure Properties of Regular Languages
2. Pumping Lemma for Regular Languages:

   Every sufficiently long string in L can be broken into three parts in such away that an arbitrary number of repetitions of the middle part yields another word in L. (we say that the middle word is "pumped.")

Exercise

It is known that there exists an algorithm for determining whether a regular language, given in standard representation is empty, finite or infinite.

Let L be any regular language on some alphabet $\{a, b\}$. Show that an algorithm exists for determining if L contains any string of even length.

Exercise

It is known that there exists an algorithm for determining whether a regular language, given in standard representation is empty, finite or infinite.

Let L be any regular language on some alphabet $\{a, b\}$. Show that an algorithm exists for determining if L contains any string of even length.

Hint:

Check

$$L \cap L((aa + bb + ab + ba)^*) = \emptyset.$$

The Pumping Lemma

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geq m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leq m,$$

and

$$|y| \geq 1,$$

such that

$$w_i = xy^i z,$$

is also in L for all $i = 0, 1, 2, \ldots$

The Pumping Lemma

The correct argument can be visualized as a gmae we play against an opponent. Our goal is to win the game by establishing a contradiction of the pumpng lemma, while the opponent tries to foil us.

Moves in the Game:

1. The opponent picks m.
2. Given m, we pick a string w in L of length equal or greater than m. We are free to choose any w, subject to $w \in L, |w| \geq m$.
3. The opponent chooses the decomposition xyz, subject to $|xy| \leq m, |y| \geq 1$.
4. We try to pick i in such a way that the pumped word w_i is not in L. If so, you win, otherwise the opponent won.

Example

The languages

1. $L = \{a^n b^n | n \geq 0\}$
2. $L = \{ww^R | w \in \Sigma^*\}$.

are not regular languages via Pumping Lemma.

Non-Regular Grammar: Linear Grammar

Example

$$G := S \rightarrow A, A \rightarrow aB|\epsilon, B \rightarrow Ab.$$

Although every production is either in right-linear or left-linear form, the grammar itself is neither right-linear nor left-linear. It is called Linear Grammar.

REMARK:

1. Regular grammar has at most one variable appears on the rightside of any production. The variable must consistently be either the rightmost or leftmost symbol of the right side of any production.
2. A linear grammar is a grammar in which at most one variable can occur on the right side of any production, without restriction on the position of the variable.
3. A regular grammar is always linear but not all linear grammars are regular.

Context-Free Languages: Type 2

Ways of Describing Type 2 Languages

1. Context-Free Grammar:

   A grammar $G = (V, T, S, P)$ is context-free if all productions in P have the form

   $$A \to x,$$

   where $A \in V$ and $x \in (V \cup T)^*$.
   A language L is a context-free language (CFL) if and only if there is a context-free grammar G such that $L = L(G)$.

2. Pushdown Automata:

   A finite automaton with *stack.*

Example

The languages

1. $L = \{a^n b^n | n \geq 0\}$
2. $L = \{ww^R | w \in \Sigma^*\}$.

are context-free languages.

Example

The languages

1. $L = \{a^n b^n | n \geq 0\}$
2. $L = \{ww^R | w \in \Sigma^*\}$.

are context-free languages.

1. $G_{ww^R} := S \rightarrow aSa|bSb|\epsilon$

Example

The languages

1. $L = \{a^n b^n | n \geq 0\}$
2. $L = \{ww^R | w \in \Sigma^*\}$.

are context-free languages.

1. $G_{ww^R} := S \rightarrow aSa|bSb|\epsilon$
2. $G_{a^n b^n} := S \rightarrow aSb|\epsilon$

CFG but Not Linear

Example

$$S \rightarrow aSb|SS|\epsilon$$

Example

$$S \rightarrow AS_1|S_1B$$

$$S_1 \rightarrow aS_1b|\epsilon$$

$$A \rightarrow aA|a$$

$$B \rightarrow bB|b$$

A CFL L is linear if there is a linear CFG, G such that $L = L(G)$.

REMARK:

$$L_{REG} \subseteq L_{LIN} \subseteq L_{CFL}$$

Exercise

1. Find a context-free grammar for the set of regular expressions on the alphabet $\{a, b\}$.
2. Let
$$L = \{a^n b^n | n \geq 0\}.$$
Show that L^2 is context-free.

Answer:

Exercise

1. Find a context-free grammar for the set of regular expressions on the alphabet $\{a, b\}$.
2. Let
$$L = \{a^n b^n | n \geq 0\}.$$
Show that L^2 is context-free.

Answer:

1. $E \rightarrow E + E|E.E|E^*|(E)|\epsilon|\emptyset|a|b.$
2. If S derives L, then SS derives L^2.

Properties of Type 2 Languages

1. The family of CFL is closed under union, concatenation, and star-closure.
2. The family of CFL is not closed under intersection and complementation.
3. Let $L_1 \in CFL$, and $L_2 \in REG$. Then $L_1 \cap L_2 \in CFL$.

Exercise

Show that

$$L = \{ a^n b^n \mid n \geq 0, n \neq 100 \} \in CFL$$

Answer

Exercise

Show that

$$L = \{ a^n b^n \mid n \geq 0, n \neq 100 \} \in CFL$$

Answer
Let $L_1 = \{ a^{100} b^{100} \}$. Then take $L_1^C \cap \{ a^n b^n \mid n \geq 0 \}$. Then the conclusion follows.

Pumping Lemma for LIN

Let L be an infinite LIN. Then there existssome positive integer m such that any $w \in L$ with $|w| \geq m$ can be decomposed as

$$w = uvxyz,$$

with

$$|uvyz| \leq m,$$

and

$$|vy| \geq 1,$$

such that

$$uv^i xy^i z \in L,$$

for all $i = 0, 1, 2, \ldots$

Pumping Lemma for LIN

Example

The language

$$L = \{w | num_a(w) = num_b(w)\}$$

is not linear.

Pumping Lemma for CFL

Let L be an infinite CFL. Then there existssome positive integer m such that any $w \in L$ with $|w| \geq m$ can be decomposed as

$$w = uvxyz,$$

with

$$|vxy| \leq m,$$

and

$$|vy| \geq 1,$$

such that

$$uv^i xy^i z \in L,$$

for all $i = 0, 1, 2, \ldots$

Example

Show that

$$\{a^n b^n c^n | n \geq 0\} \notin CFL.$$

Recusively Enumerable Languages: Type 0

Way of Describing Type 0 Languages

1. A language L is recursively enumerable if there exists a Turing Machine that accepts it.
2. A language L is recursively enumerable iff there is a grammar G where all the productions are of the form $u \longrightarrow v$, $u \in (V \cup T)^+$ and $v \in (V \cup T)^*$. G is called an unrestricted grammar.
3. A language is recursive if there exists a Turing machine that accepts it and that halts in every input.

REMARK

1. Unrestricted grammar, essentially has no conditions imposed on the production other than disallowing ϵ as the leftside of a production.
2. Unrestricted grammar generates words by a well-defined algorithmic process, so derivations can be done on a Turing Machine, and conversely.

Exercise

What language does the unrestricted grammar
$S \to S_1 B, \quad S_1 \to a S_1 b, \quad bB \to bbbB,$
$aS_1 b \to aa, \quad B \to \epsilon$
derive?
Answer:

Exercise

What language does the unrestricted grammar
$S \rightarrow S_1 B, \quad S_1 \rightarrow a S_1 b, \quad bB \rightarrow bbbB,$
$aS_1 b \rightarrow aa, \quad B \rightarrow \epsilon$
derive?
Answer:

$$L = \{a^{n+1} b^{n+k} | n \geq 1, k = -1, 1, 3, ...\}$$

Context-Sensitive Languages: Type 1

Way of Describing Type 1 Languages

1. A language L is context-sensitive if there is a grammar G such that all productions are of the form

$$x \rightarrow y,$$

where $x, y \in (V \cup T)^+$,

$$|x| \leq |y|,$$

and $L = L(G)$ or $L = L(G) \cup \{\epsilon\}$.

Example

The language $L = \{a^n b^n c^n | n \geq 0\}$ *is generated by a grammar*

$$
\begin{array}{lcl}
S & \longrightarrow & abc|aAbc \\
Ab & \longrightarrow & bA \\
Ac & \longrightarrow & Bbcc \\
bB & \longrightarrow & Bb \\
aB & \longrightarrow & aa|aaA
\end{array}
$$

Properties of Type 1 Languages

1. Every CSL L is recursive.
2. There exists a recursive language that is not CSL.

Inclusion Relation

$$L_{REG} \subseteq L_{CF} \subseteq L_{CS} \subseteq L_{RE}$$

Refinements...

$$L_{REG} \subseteq L_{DCF} \subseteq L_{CF} \subseteq L_{CS} \subseteq L_{REC} \subseteq L_{RE}$$

$$L_{REG} \subseteq L_{LIN} \cap L_{DCF} \subseteq L_{LIN} \cup L_{DCF} \subseteq L_{CF}$$

Salamat Po