What is Theory of Computation? (Lecture 1)

- Theory of computation is the branch that deals with whether and how efficiently problems can be solved on a model of computation, using an algorithm.
- The field is divided into three major branches: automata theory, computability theory and computational complexity theory.

What is Automata Theory?

- Automata theory is the study of abstract machines and the computational problems that can be solved using these machines.
- Automaton = an abstract computing devices imply means any machine.
  - Note: A "device" need not even be a physical hardware!
  - It has a mechanism to read input (string over a given alphabet, e.g. strings of 0's and 1's on S = {0,1}) written on an input file.
- An **abstract machine**, also called an **abstract computer**, is a theoretical model of a computer hardware or software system used in Automata theory.
- This automaton consists of:
  - States (represented in the figure by circles).
  - Transitions (represented by arrows).

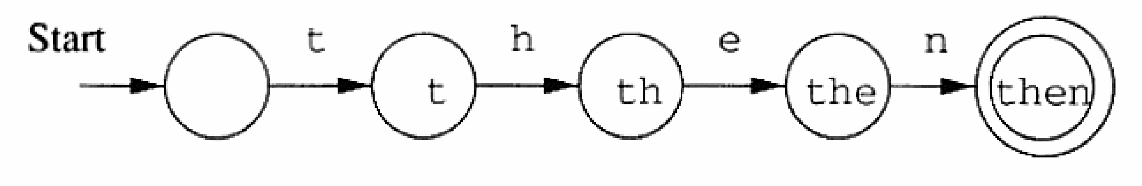

Figure 1.2: A finite automaton modeling recognition of **then**

- As the automaton sees a symbol of input, it makes a *transition* (or *jump*) to another state,
- According to its ***transition function*** (which takes the current state and the recent symbol as its inputs).
- Uses of Automata: compiler design and parsing.

Languages & Grammars

- **Languages**: "*A language is a collection of sentences of finite length all constructed from a finite alphabet of symbols*".
- **Grammars**: "*A grammar can be regarded as a device that enumerates the sentences of a language*" - nothing more, nothing less.

Languages

- An alphabet $\sum$ is a finite set of symbols.

– Example: Common alphabets include the binary alphabet {0, 1}, the English alphabet {A, B… Z, a, b… z}.

– A language is a set of strings

- String: A sequence of letters
  - Examples: “cat”, “dog”, “house” …
  - Defined over an alphabet:

$$\Sigma = \{a, b, c, \ldots, z\}$$

  - The alphabet of a language is normally denoted by ∑.
  - The set of strings, including empty, over an alphabet Σ is denoted by Σ^*.
  - $\Sigma^+ = \Sigma^* - \{\epsilon\}$

Alphabets and Strings

– **Example of alphabet**:

- 0110, 11, 001 are three strings over the binary alphabet {0, 1}.
- aab, abcb, b, cc are four strings over the alphabet { a, b, c }.

– Strings

$$a$$
$$ab$$
$$abba$$
$$baba$$
$$aaabbbaabab$$

$$u = ab$$
$$v = bbbaaa$$
$$w = abba$$

String Operations

$$w = a_1 a_2 \cdots a_n \qquad abba$$

$$v = b_1 b_2 \cdots b_m \qquad bbbaaa$$

– **Concatenation**

$$wv = a_1 a_2 \cdots a_n b_1 b_2 \cdots b_m \qquad abbabbbaaa$$

$w = a_1 a_2 \cdots a_n$ $\quad$ $ababaaabbb$

– **Reverse**

$w^R = a_n \cdots a_2 a_1$ $\quad$ $bbbaaababa$

– **String Length**

$w = a_1 a_2 \cdots a_n$

Length: $|w| = n$

Examples:

$$|abba| = 4$$
$$|aa| = 2$$
$$|a| = 1$$

– **Length of Concatenation**

$|uv| = |u| + |v|$

Example:

$$u = aab, \quad |u| = 3$$
$$v = abaab, \quad |v| = 5$$

$$|uv| = |aababaab| = 8$$
$$|uv| = |u| + |v| = 3 + 5 = 8$$

– **Empty String**

- A string with no letters: λ
- Observations:

$$|\lambda| = 0$$

$$\lambda w = w\lambda = w$$

$$\lambda abba = abba\lambda = abba$$

– Substring

- Substring of string:
  - a subsequence of consecutive characters

String	Substring
$\underline{ab}bab$	ab
$\underline{abba}b$	$abba$
$ab\underline{b}ab$	b
$a\underline{bbab}$	$bbab$

– Prefix and Suffix

$abbab$

Prefixes	Suffixes
λ	$abbab$
a	$bbab$
ab	bab
abb	ab
$abba$	b
$abbab$	λ

$w = uv$

prefix (u), suffix (v)

– Another Operation

$$w^n = \underbrace{ww \cdots w}_{n}$$

Example: $(abba)^2 = abbaabba$

Definition: $w^0 = \lambda$

$$(abba)^0 = \lambda$$

– The * Operation

Σ^* : the set of all possible strings from alphabet Σ

$$\Sigma = \{a, b\}$$

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

– **The + Operation**

Σ^+ : the set of all possible strings from alphabet Σ except λ

$$\Sigma = \{a, b\}$$
$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

$$\Sigma^+ = \Sigma^* - \lambda$$
$$\Sigma^+ = \{a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

Languages

- Languages-A set of strings which are chosen from some Σ*, where Σ is a particular alphabet, is called a language.
  - If Σ is an alphabet, and L subset of Σ*, then L is said to be language over alphabet Σ.
  - For ***example*** the language of all strings consisting of n 0's followed by n 1's for some n>=0: {ϵ,01,0011,000111,-------}
- Another definition of language: A set of letters that called Alphabet. This can be seen in any natural language, for example the alphabet of English can be defined as:

  E = {a, b… z}

- By concatenate letters from alphabet, we get words.
- All words from the alphabet make language.
- Language can be classified into two types as follows:
  - Natural Languages
  - Formal Languages
- As in the natural language not all concatenations make permissible words, the same things happen with the formal languages.

Note: formal language deals with form not meaning.
Note: alphabet could be a set of an empty set (or null string) which is a string of no letters.

- ☒ **Alphabet: finite**
- ☒ **Words: finite**
- ☒ **Language: infinite**

A language is any subset of Σ^*

Example: $\Sigma = \{a, b\}$

$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, \ldots\}$

Languages: $\{\lambda\}$

$\{a, aa, aab\}$

$\{\lambda, abba, baba, aa, ab, aaaaaa\}$

Note that:

Sets	$\varnothing = \{\ \} \neq \{\lambda\}$
Set size	$\lvert\{\ \}\rvert = \lvert\varnothing\rvert = 0$
Set size	$\lvert\{\lambda\}\rvert = 1$
String length	$\lvert\lambda\rvert = 0$

– **Another Example**

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

$$\left.\begin{matrix} \lambda \\ ab \\ aabb \\ aaaaabbbbb \end{matrix}\right\} \in L \qquad abb \notin L$$

Definition (Regular Language): Let ∑ be an alphabet. The following are precisely the regular languages over ∑:

- The empty language Ø; is regular.
- For each a ∈ ∑, {a} is regular.
- Let L_1; L_2 be regular languages over ∑. Then $L_1 \cup L_2$; $L_1 . L_2$, and L_1^* are all regular.

Remark: The operation **.** is string concatenation. Formally, $L_1 . L_2 = \{xy: x \in L_1; y \in L_2\}$.

- **Operations on Languages**

The usual set operations

$$\{a,ab,aaaa\}\cup\{bb,ab\}=\{a,ab,bb,aaaa\}$$
$$\{a,ab,aaaa\}\cap\{bb,ab\}=\{ab\}$$
$$\{a,ab,aaaa\}-\{bb,ab\}=\{a,aaaa\}$$

Complement: $\overline{L}=\Sigma^*-L$

$$\overline{\{a,ba\}}=\{\lambda,b,aa,ab,bb,aaa,\ldots\}$$

- **Reverse**

Definition: $L^R=\{w^R : w\in L\}$

Examples: $\{ab,aab,baba\}^R=\{ba,baa,abab\}$

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

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

- **Concatenation**

Definition: $L_1L_2=\{xy : x\in L_1, y\in L_2\}$

Example:

$$\{a,ab,ba\}\{b,aa\}$$

$$=\{ab,aaa,abb,abaa,bab,baaa\}$$

- **Another Operation**

Definition: $L^n=\underbrace{LL\cdots L}_{n}$

$$\{a,b\}^3=\{a,b\}\{a,b\}\{a,b\}=$$
$$\{aaa,aab,aba,abb,baa,bab,bba,bbb\}$$

Special case: $L^0=\{\lambda\}$

$$\{a,bba,aaa\}^0=\{\lambda\}$$

- **Palindrome :** is a language that={ Λ, all strings x such that reverse(x)=x}

Example aba, aabaa, bab, bbb…

– More Examples

$$L = \{a^n b^n : n \geq 0\}$$
$$L^2 = \{a^n b^n a^m b^m : n, m \geq 0\}$$
$$aabbaaabbb \in L^2$$

– Star-Closure (Kleene *)

Definition: $L^* = L^0 \cup L^1 \cup L^2 \cdots$

Example:

$$\{a, bb\}^* = \begin{Bmatrix} \lambda, \\ a, bb, \\ aa, abb, bba, bbbb, \\ aaa, aabb, abba, abbbb, \ldots \end{Bmatrix}$$

Definition Kleene Closure: Let $\sum$ be an alphabet. The Kleene closure of $\sum$, denoted $\sum^*$, is the set of all finite strings whose characters all belong to $\sum$. Formally, $\Sigma^* = \cup_{n \in \mathbb{N}} \sum^n$. The set $\sum^0 = \{\varepsilon\}$, where ε is the empty string.

Note: ε = Λ = λ is empty string.

Definition Kleene star * (Kleene Closure): Given an alphabet Σ we wish to define a language in which any string of letters from Σ is a word, even the null string, this language we shall call the closure of the alphabet.

Example if Σ= {a,b,c}

Then Σ*= {Λ a b c aa ab ac ba bb bc ca cb cc aaa aab aac bba bbb bbc cca ccb ccc aaaa aaab …}

Let's S= alphabet of language S*= closure of the alphabet

Example S= {x}

S*= {Λ, x^n |n>=1}

To prove a certain word in the closure language S* we must show how it can be written as a concatenation of words from the set S.

Example Let S= {a,ab}

To find if the word abaab is in S* or not, we can factor it as follows: (ab)(a)(ab)

Every factor in this word is a word in S* so as the whole word abaab. In the above example, there is no other way to factor the word that we called unique.

While, sometimes the word can be factored in different ways.

- **Positive Closure**

Definition: $L^{+} = L^{1} \cup L^{2} \cup \cdots$

$$= L^{*} - \{\lambda\}$$

$$\{a, bb\}^{+} = \begin{Bmatrix} a, bb, \\ aa, abb, bba, bbbb, \\ aaa, aabb, abba, abbbb, \ldots \end{Bmatrix}$$