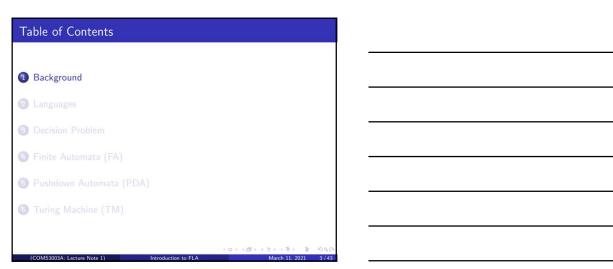
Introduction to Formal Languages and Automata	
COMS3003A: Lecture Note 1	
March 11, 2021	
(COMS 3003A: Lecture Note 1) Introduction to FLA March 11, 2021 1.743	
(COMS3003A: Lecture Note 1) Introduction to FLA March 11, 2021 1/43	
Outline	
Outilile	
1 Background	
2 Languages	
3 Decision Problem	

Finite Automata (FA)

6 Turing Machine (TM)

6 Pushdown Automata (PDA)



Λ	hi	+	$^{-1}$	- h	1	\sim 1	100	\sim	IIn		
Α	υı	L	UI.	L	ıa	ur	12 1	U	411	ıu	P

Theory of Computation is a branch of computer science (theoretical computer science) that deals with the fundamental mathematical properties of computer hardware, software, and certain applications thereof 1

The theory of computation has three branches, namely:

- Automata Theory
- Computability Theory
- Complexity Theory

Our focus in this course is Automata Theory.

¹Michael Sipser (2013). Introduction to the Theory of Computation 3rd. Cengage

What is Automata Theory?

Automata theory

- is the study of abstract machines
- \bullet deals with the definitions and properties of mathematical models of

These mathematical models of computation or abstract machines are called Automata.

In this course, we will examine different mathematical models of computation including:

- Finite Automata
- Push Down Automata
- Turing Machines

5

Formal Languages and Automata

Automata (the plural of automaton) are usually described by the class of formal language they recognize.

Formal languages in simple terms refer to mathematically represented

We will look into formal languages later on in this lecture.

Why Study Formal Languages and Automata? To understand the theoretical foundations of computer science. To understand the underlying principles and theory of how a computer works. The topics we will cover have practical use in other aspects of computer science, such as compilers and programming languages.

Table of Contents			
Background			
2 Languages			
3 Decision Problem			
Finite Automata (FA)			
Pushdown Automata (F			
Turing Machine (TM)			
(COMS3003A: Lecture Note 1)	Introduction to FLA	March 11, 2021	8/43

Languages
Informal definition: a language is a set of strings involving symbols from some alphabet.
Examples: Natural languages, e.g., English High-level programming languages, e.g., Java A random assortment of unrelated strings
(COMS3003A: Lecture Note 1) Introduction to FLA March 11, 2021 9/43

	1
Components of a language	
A language comprises of three components: • The alphabet	
Strings	
• Rules	
(0) (2) (2) 2 990	
(COMS3003A: Lecture Note 1) Introduction to FLA March 11, 2021 10/43	
English Language	1
Let's examine the English language based on the three components:	
The Alphabet	
The English alphabet contains 26 letters (upper and lowercase), blanks, punctuation symbols.	
(COMS3003A: Lecture Note 1) Introduction to FLA March 11, 2021 11/43	
11	
English Language	
Let's examine the English language based on the three components:	
The Alphabet	
The English alphabet contains 26 letters (upper and lowercase), blanks, punctuation symbols.	

Strings

Strings in English can be words or sentences e.g. Hat (word); The cat in the hat. (a sentence)

English Language
Let's examine the English language based on the three components:
The Alphabet
The English alphabet contains 26 letters (upper and lowercase), blanks, punctuation symbols.
Strings
Strings in English can be words or sentences e.g. Hat (word); The cat in the hat. (a sentence)
The Rules: Is the English language simply a 'set of strings'?
No: We accept The cat is in the hat , but not The the cat hat is in . Not all combinations of the alphabets are valid strings. There are rules that govern what set of strings are acceptable.
(COMS3003A: Lecture Note 1) Introduction to FLA March 11, 2021 11/43
13

Recap: Sets

To understand formal languages, we need to understand the notations used to describe them.

A set is a group of objects represented as a unit.

The objects in a set are referred to as elements or members.

While sets can be described in different ways, in this course, we will list the elements of a set in a pair of curly brackets. $S=\{a,b,c\}$

The membership of a set is denoted by the $\in \mathsf{symbol}$

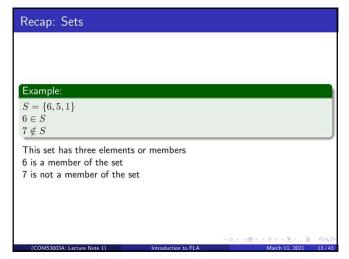
Non-membership of a set is denoted by the \notin symbol

(COMS3003A: Lecture Note 1)

Introduction to FLA

March 11, 2021 12 / 43

14



Towards a formal definition of a language
A language is a set of strings over an alphabet of symbols.
An alphabet is a finite set of symbols. First specify the alphabet of all legal symbols that can be used to form
strings in the language.
Examples: English: the alphabet contains 26 letters (upper and lowercase), blanks, punctuation symbols.
Programming languages: the alphabet usually contains alphabetic letters (upper and lowercase, by definition), numeric digits, blanks, underscores and other punctuation symbols.
An alphabet can even have only one symbol, as we'll see in the definition of regular languages.
(D) (B) (\$) (\$) \$ 990
(COMS3003A: Lecture Note 1) Introduction to FLA March 11, 2021 14/43
10
Charles are
Strings
The alphabet is represented with the $\boldsymbol{\Sigma}$ symbol.
Let Σ be an alphabet. A string over Σ is a number (perhaps zero) of
elements of Σ placed in order.
Example: $\Sigma = \{a,b\}$
Some strings over Σ are a,aaa,baa,aba and $aabba$.
(COMS3003A: Lecture Note 1) Introduction to FLA March 11, 2021 15 / 43
17
The Empty String
The null (empty) string is the string of no symbols.
We'll denote it with λ .
In other publications, you might find the null string represented with other symbols e.g., ε , \wedge .
Symbols e.g., E, /\.

Note that we'll never represent a symbol of Σ with $\lambda.$

Length of Strings
If x is a string over Σ , the length of x is the number of symbols in x .
We'll denote it by $\left x\right $.
Examples: $ aaa = aba = 3$
a = 1
$ \lambda = 0.$
Note that the length of an empty string is 0.
(D)(B)(E)(E) & O(C)
(COMS3003A: Lecture Note 1) Introduction to FLA March 11, 2021 17/43
19

For a string x over Σ and an element $a \in \Sigma$, $n_a(\boldsymbol{x}) =$ the number of occurrences of the symbol a in the string \boldsymbol{x} Example: $\Sigma = \{a,b\}$ x = abbab $n_a(x) = 2$ $n_b(x) = 3$

20

Rules of the Language Let Σ be an alphabet. We denote the set of all strings over Σ by $\Sigma^*.$ Any language over Σ is therefore a subset of $\Sigma^*.$ $\Sigma^* = \{\lambda, a, b, aa, ab, ba, aaa, aba, \ldots\}$ Some languages over $\boldsymbol{\Sigma}$ are: $\bullet \ \{\lambda,a,b,aa\}$ $\bullet \ \{x \in \{a,b\}^* | \ |x| \leq 8\}$ $\bullet \ \{x \in \{a,b\}^* | \ |x| \ \mathsf{is} \ \mathsf{odd}\}$ $\bullet \ \{x \in \{a,b\}^* | \ x \ \text{begins and ends with} \ b\}$

Set Operations on La	nguages	
Languages are sets of strin	gs, therefore new la	nguages can be constructed
using set operations.		
Given any two languages L $L_1 \cup L_2$; intersection $L_1 \cap$ over Σ .		Iphabet Σ , their union, L_1-L_2 ; are also languages
(COMS3003A: Lecture Note 1)	Introduction to FLA	March 11, 2021 20 / 43

Complement of a Language

The complement of a language L over Σ is

 $L' = \Sigma^* - L.$

This means the set of strings over Σ^{\ast} but not in L.

Note that $\Sigma^* - L$ can also be represented as $\Sigma^* \backslash L$

23

String Concatenation

We can construct new languages using the concatenation operation on strings.

Let $x,y\in \Sigma^*.$ The concatenation of x and y is xy.

 ${\sf Example:}\ x=abb, y=ba$

Then xy = abbba, yx = baabb.

For any string $x,x\lambda=\lambda x=x$

Concatenation is associative: for any strings x,y,z,(xy)z=x(yz) .

Thus we don't have to worry about the order in which the operations are performed.

Concatenation of Languages

We can apply concatenation to languages as well as strings.

For $L_1, L_2 \subseteq \Sigma^*$,

$$L_1L_2 = \{xy | x \in L_1 \text{ and } y \in L_2\}$$

 $\{hope, fear\}\{less, fully\} = \{hopeless, hopefully, fearless, fearfully\}$

We use exponential notation to indicate the number of items being concatenated. Note that these items can be symbols, strings or languages.

25

The Exponential Notation

The exponential notation can be used to represent the concatenation of \boldsymbol{k} copies of a symbol, string, or language.

Let Σ be an alphabet, $a \in \Sigma$, $x \in \Sigma^*$, and $L \subseteq \Sigma^*$. Then

$$\begin{aligned} a^k &= aa \dots a \\ x^k &= xx \dots x \\ \Sigma^k &= \Sigma\Sigma \dots \Sigma = \{x \in \Sigma^* | \, |x| = k\} \\ L^k &= LL \dots L \end{aligned}$$

where on each right hand side there are \boldsymbol{k} items altogether.

Consider k=0:

$$a^{0} = \lambda; \ x^{0} = \lambda; \ \Sigma^{0} = \{\lambda\}; \ L^{0} = \{\lambda\}$$

26

By definition,

$$L^k = LL \dots L = \{x_1 x_2 \dots x_k \mid x_i \in L\}$$

in other words, \mathcal{L}^k is the set of all strings that can be obtained by concatenating k elements of L.

We want to define the set of all strings that can be obtained by concatenating any number of elements of L, denoted by L^{\ast} :

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

Note:

- ullet There is a mistake in Edition 2 of the prescribed text. The superscript of L should be i, not k.
- The operation * is often called the Kleene star or the Kleene closure.
- Since $L^0=\{\lambda\}$, λ is always in L^* , regardless of what L is.

Introduction to ELA

02 4 2 2 4 2 4 9 Q C

March 11, 2021 26

28

We want to define the set of all strings that can be obtained by concatenating one or more elements of L, denoted by $L^+\colon$

$$L^+ = \bigcup_{i=1}^{\infty} L^i$$

Note:

- \bullet Here is another mistake in Edition 2 of the prescribed text. The superscript of L should be i, not k.
- $\bullet \ L^+ = L^*L = LL^*$
- $\bullet \ L^+ \ {\rm and} \ L^* \ {\rm may \ be \ equal}.$

(COMS3003A: Lecture Note 1

ntroduction to FLA

) (E) (E) E -090

29

Substrings

Definition: Let $x,\,y$ be strings. We say x is a substring of y if there are strings w and z, not necessarily empty, so that y=wxz.

Example: car is a substring of each of $descartes, \, vicar, \, carthage, \, car, \, {\it but}$ not of charity.

Definition: A prefix of a string is an initial substring, e.g., the prefixes of abaa are $\lambda,a,ab,aba,abaa$.

Definition: A suffix of a string is a final substring, e.g., the suffixes of abaa are $\lambda, a, aa, baa, abaa$.

(COMS3003A: Lecture Note 1)

ntroduction to FLA

B > (2) (2) 2 00

ř					
ı	ntı	nite	Lan	OII:	ages

Most interesting languages have an infinite number of strings. If we want to work with an infinite language, we must be able to specify it in a way that is finite. There are various approaches to that. Consider

$$L_1 = \{ab, bab\}^* \cup \{b\}\{bb\}^*$$

and

$$L_2 = \{x \in \{a, b\}^* \mid n_a(x) \ge n_b(x)\}$$

where $n_a(x)$ represents the number of $a\mbox{'s}$ in the string $x\mbox{,}$ similarly for $n_b(x).$

OMS3003A: Lecture Note 1) Introduction to FLA March 11, 2021 2

31

 $L_1 = \{ab, bab\}^* \cup \{b\}\{bb\}^*$:

We describe ${\cal L}_1$ by saying how an arbitrary string in it can be ${\it constructed}$:

- \bullet by concatenating an arbitrary number of strings, each of which is either ab or bab
- \bullet by concatenating the string b with an arbitrary number of copies of the string bb

(D) (B) (E) (E) 2 - O((

32

 $L_2 = \{x \in \{a, b\}^* \mid n_a(x) \ge n_b(x)\}$

We describe L_2 by specifying a property that characterizes the strings in it, in other words, we specify how to recognize a string in it – count the number of a's and the number of b's and compare the two.

COMS3003A: Lecture Note 1) Introduction to FLA March 11, 2021 3

Whether a description concentrates on how to construct or how to recognize a string, is not always clear (there is no clear line separating them). Consider

$$L_3 = \{byb|y \in \{a,b\}^*\}$$

The definition can be interpreted as giving a method of

- \bullet generating strings: start with an arbitrary string y and add b to each end.
- recognizing strings: assuming that the total length of a string is at least 2, compare the symbols at the ends.

Even if a definition clearly belongs to one category, it might suggest a definition of the other category.

4

March 11, 2021 32 / 43

34

We'll consider

- more and more general ways of generating languages, and
- \bullet increasingly sophisticated methods for recognizing strings in these languages.

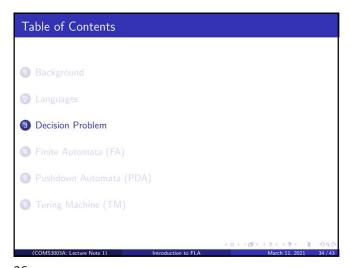
One can think of the algorithm for recognizing the language as an abstract machine. A precise decription of the machine effectively gives us a precise way of specifying which strings are in the language.

(COMS3003A: Lecture Note 1)

Introduction to FLA

March 11, 2021 33 / 43

35



$\overline{}$					
	ecis	lon	Pro	h	Δm

- In simple terms, we think of the computer as a device that receives an input, performs some form of computation and returns the output.
- There are different types of computational problems which have different levels of complexity.
- A computational problem that requires the computer to return a 'yes' or 'no' as output, i.e., a problem for which every specific instance can be answered 'yes' or 'no', is a Decision problem.
- Example: Given a positive integer n, is it prime?
- ullet We encode n as a string of digits; a computation that solves the problem starts with this string and ends with 'yes' or 'no'.
- Different computational problems can be formulated as decision problems, in this course, we will look into several of them.

OMS3003A: Lecture Note 1)	Introduction to ELA		March 11, 202
		(0) (6)	451451

Language Recognition Problem as a Decision Problem

Solving any decision problem can be thought of as recognizing a certain language.

Language recognition problem: Take an arbitrary string of digits and determine whether it is one of the strings in the given language.

In our example on prime numbers, given a string, we want to determine if it is one of the strings in the languages of all strings representing primes.

The computation accepts a string and returns yes, if the string is a prime number, or no otherwise.

38

Computing machines of different types recognize languages of different complexity.

As mentioned earlier, we'll be concerned with

- Finite automata
- Pushdown automata
- Turing machines

In this lecture, we'll touch on these computing machines briefly. In subsequent lectures, we'll examine them in more details.

Table of Contents	
Background	
2 Languages	
3 Decision Problem	
(FA) Finite Automata (FA)	
(5) Pushdown Automata (PDA)	
(6) Turing Machine (TM)	
(COMS3003A: Lecture Note 1) Introduction to FLA	March 11, 2021 38/43

Any system that is at each moment in one of a finite number of discrete states, and moves among these states in a predictable way in response to individual input signals, can be modeled by a finite automaton. The languages that FAs recognize are called regular. Regular languages are the languages obtained from one-element languages by repeated applications of certain basic operations. FAs are used in compiler design, text editing, etc.



Push	down /	Automata ((PDA)

The most obvious limitation of an FA is that it has no memory, over and above being able to keep track of its current state. Therefore an FA can recognize only simple languages.

A pushdown automaton is an FA with auxiliary memory in the form of a stack.

The languages that PDAs recognize are called context-free.

Context-free languages allow a more complicated syntax than regular languages.

PDAs can be used to parse statements in high-level programming languages.

43

Table of Contents Background Languages Decision Problem Finite Automata (FA) Pushdown Automata (PDA) **6** Turing Machine (TM)

44

Turing Machine

A PDA is more powerful than an FA. However, the fact that the memory is organized as a stack imposes constraints.

A $\it Turing\ machine$ is an FA with the ability to change symbols on the input.

The languages that Turing machines recognize are called recursively enumerable.

Recursively enumerable languages are more general than context-free $\,$ languages.

Turing machines can carry out any algorithm whatsoever.