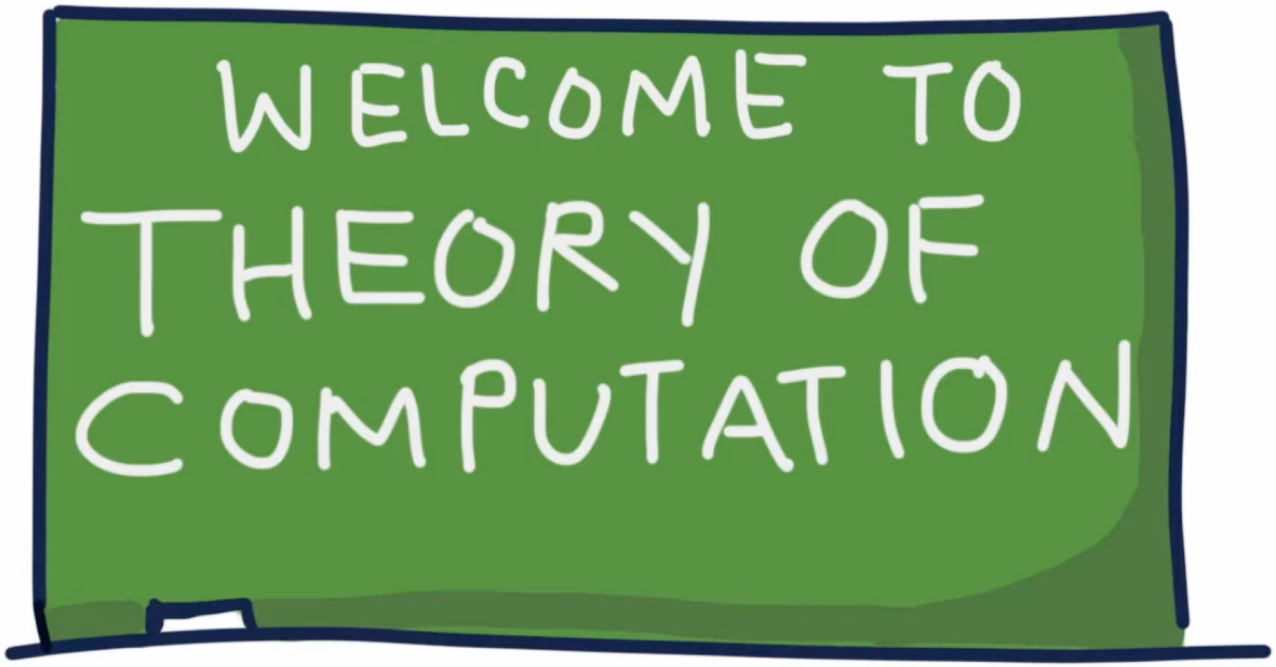
CS F351: Theory of Computation

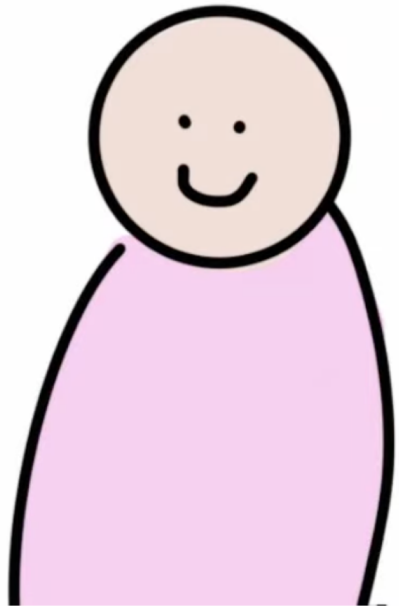
01 – Alphabets, Strings & Languages

BITS Pilani
Dubai Campus

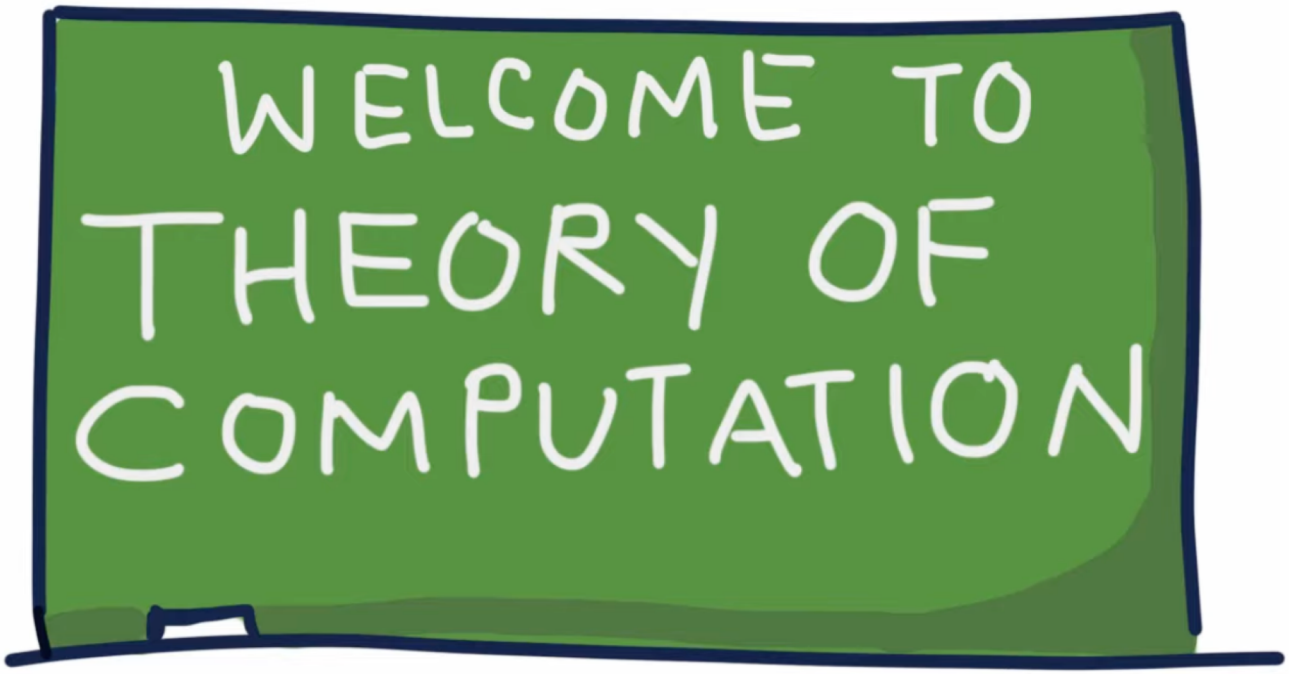
Dr Elakkiya R, AP/CS

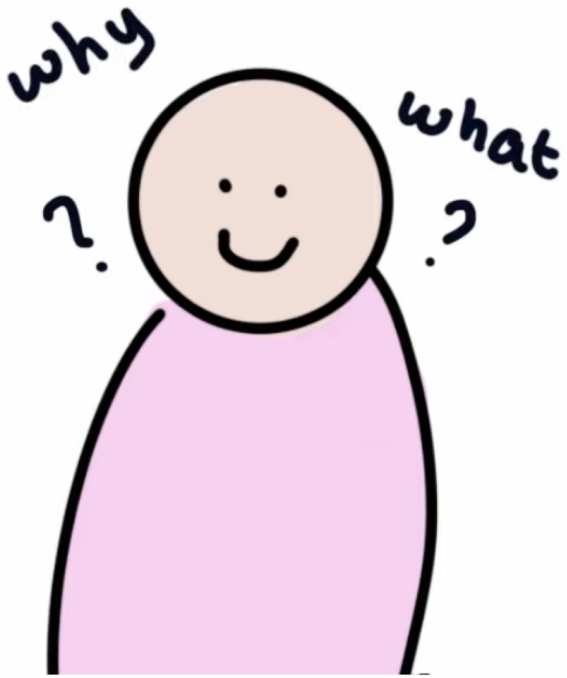
Introduction



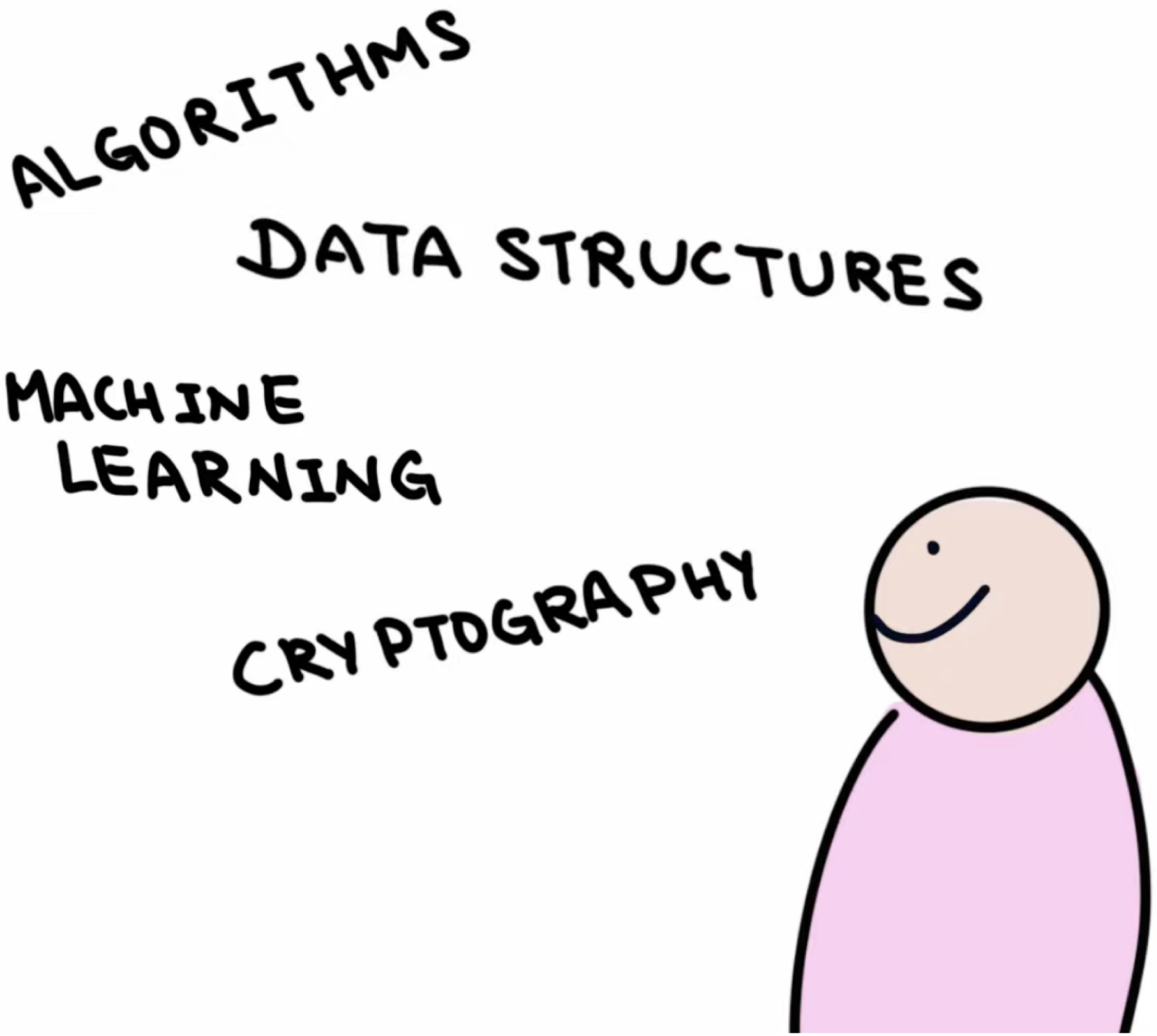


Introduction

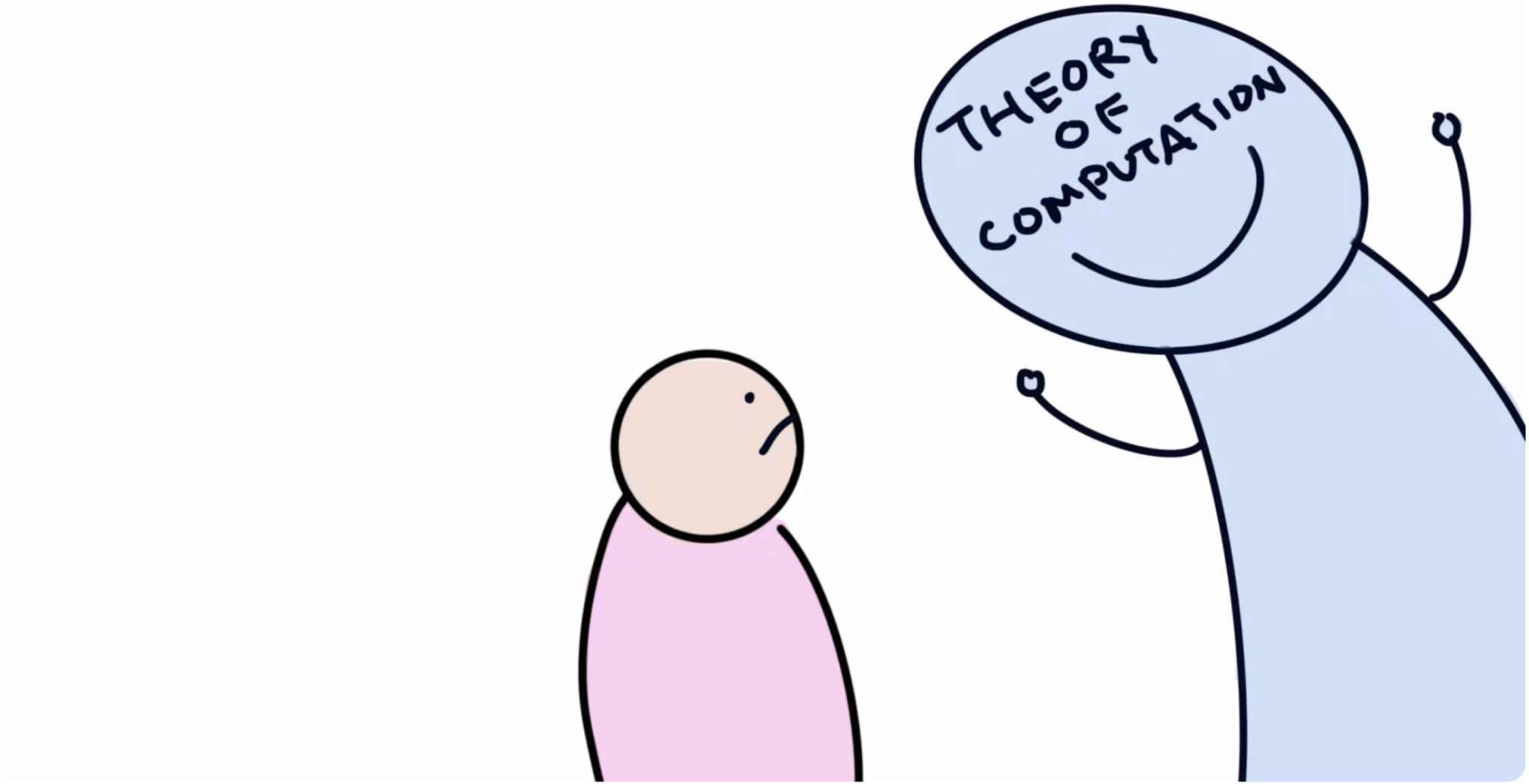




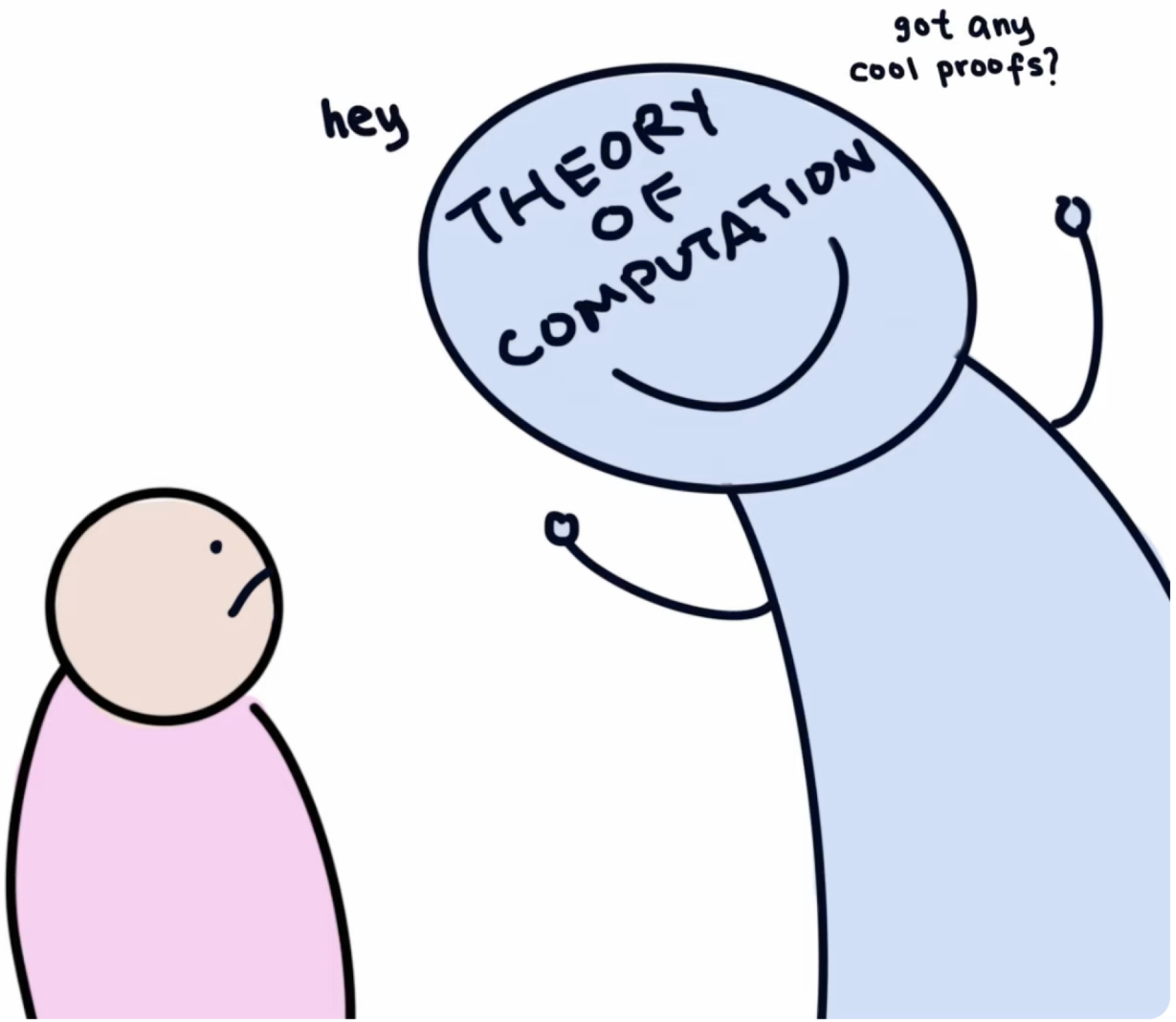
Introduction



Introduction

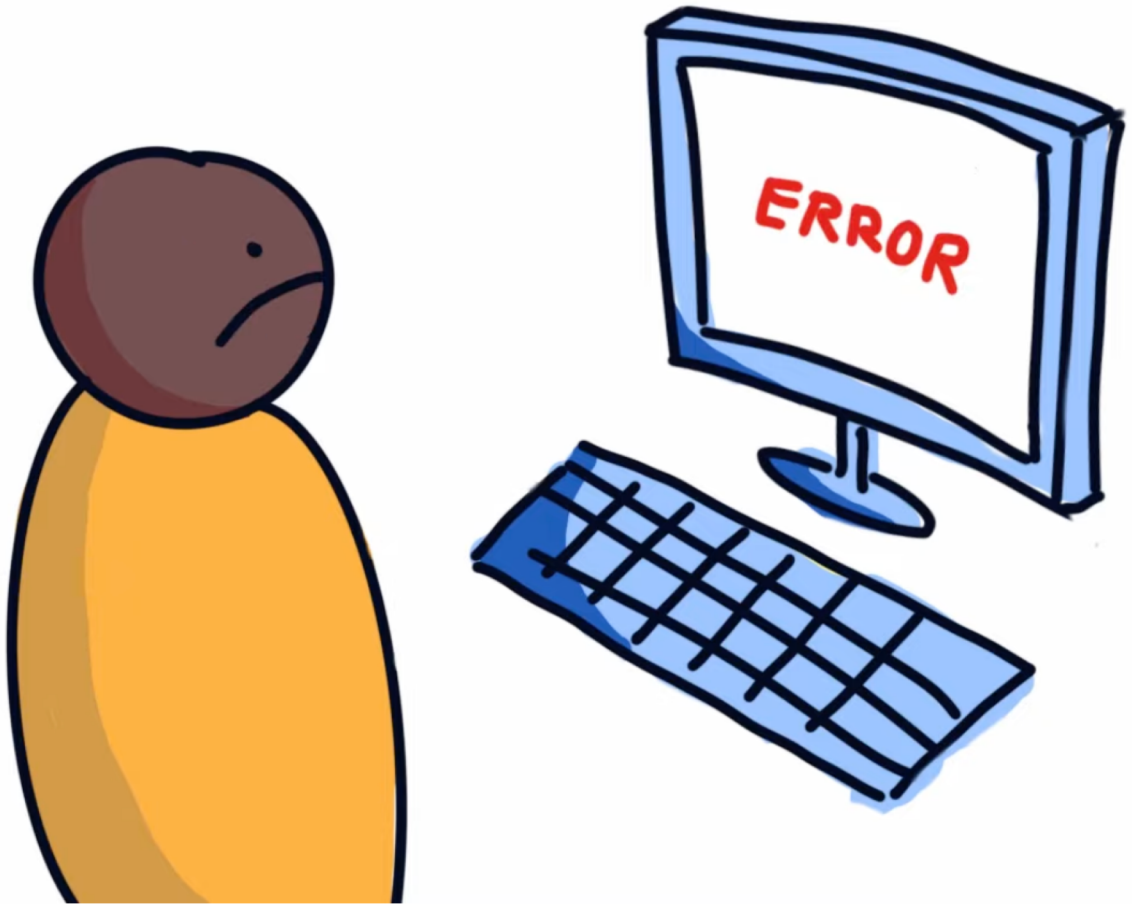


Introduction



Introduction

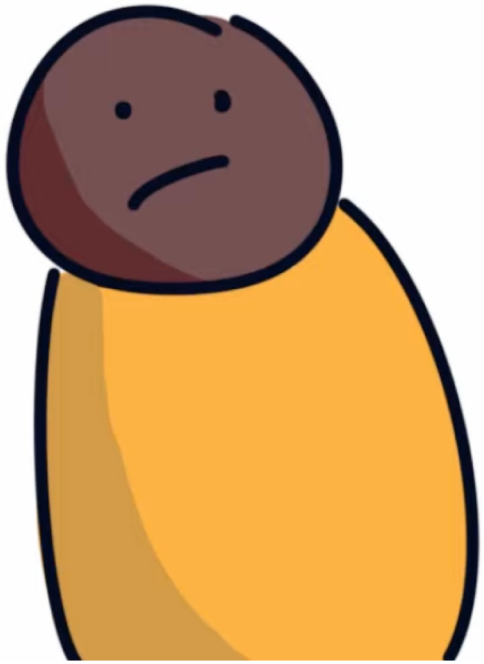
Introduction



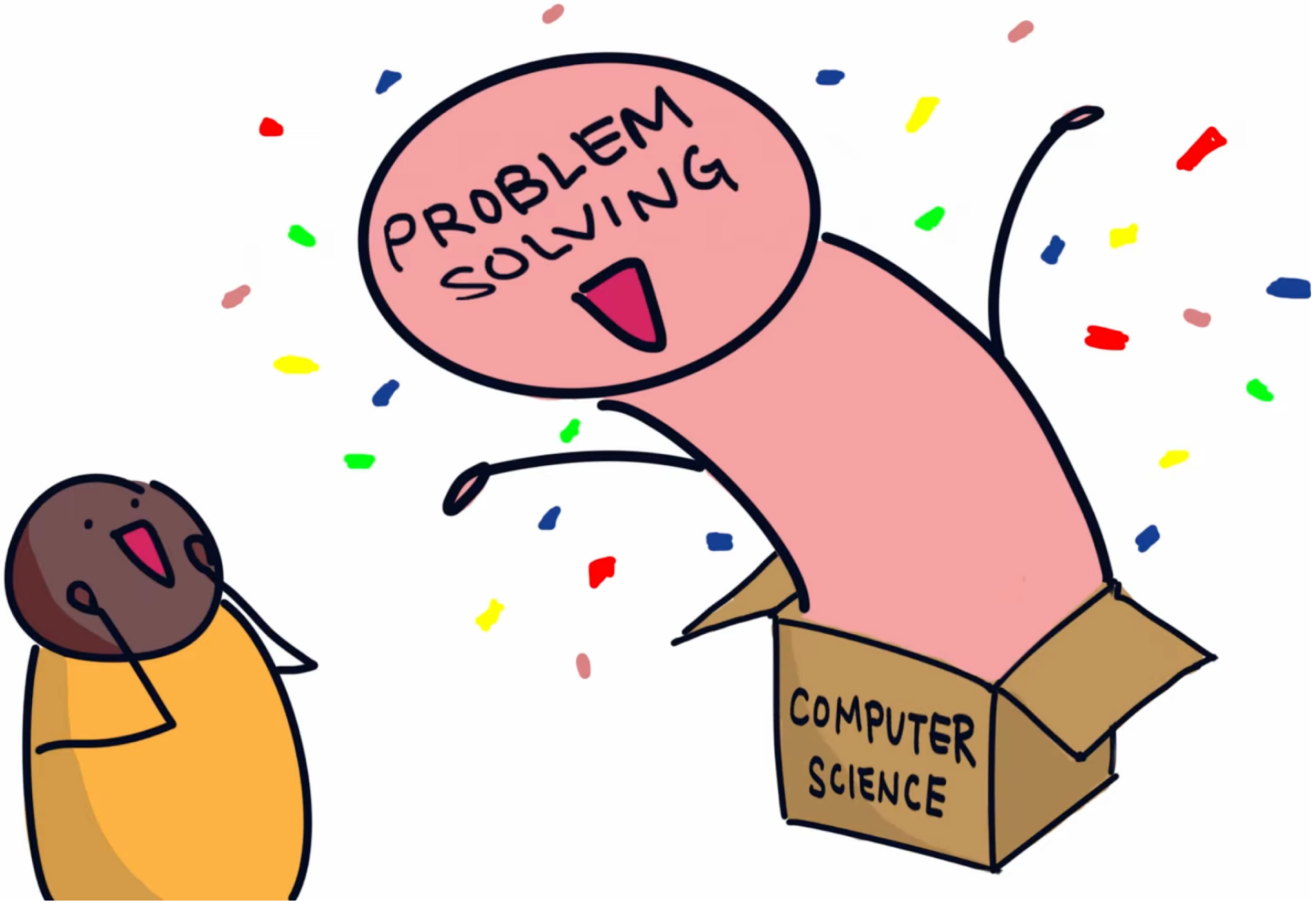
Introduction

Introduction

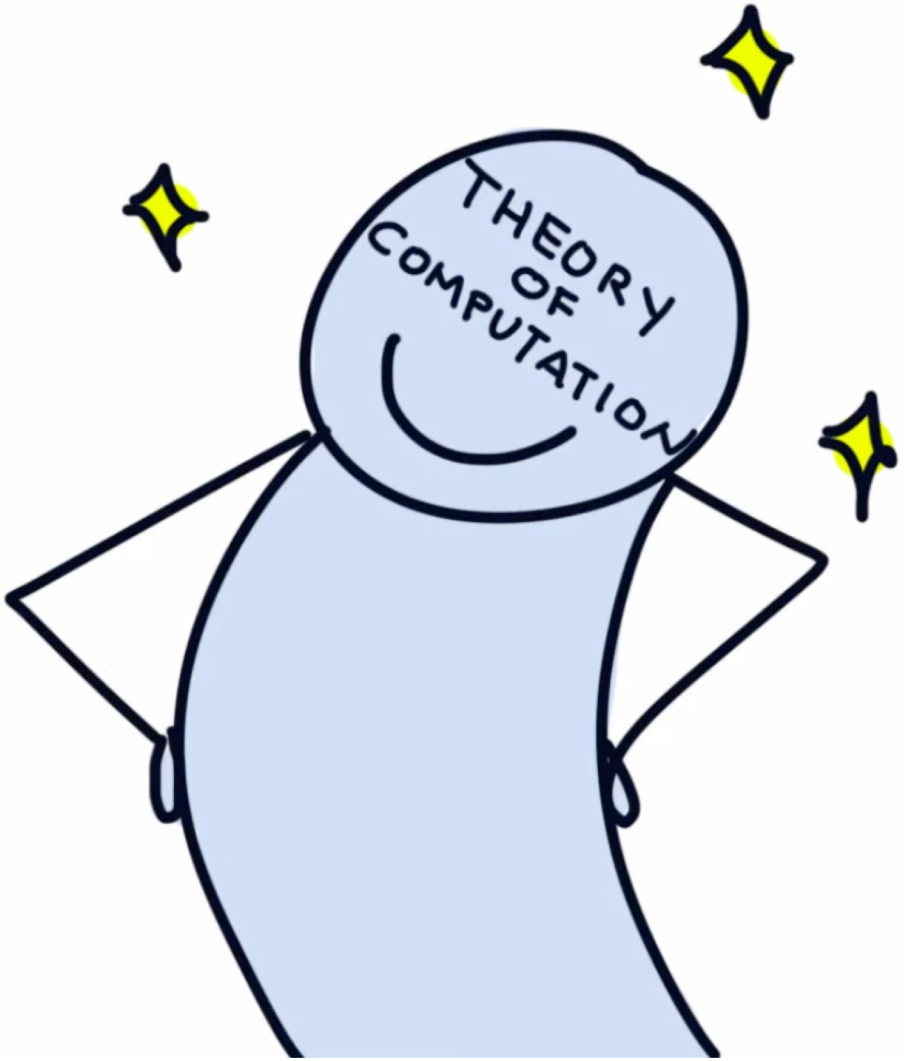
Introduction



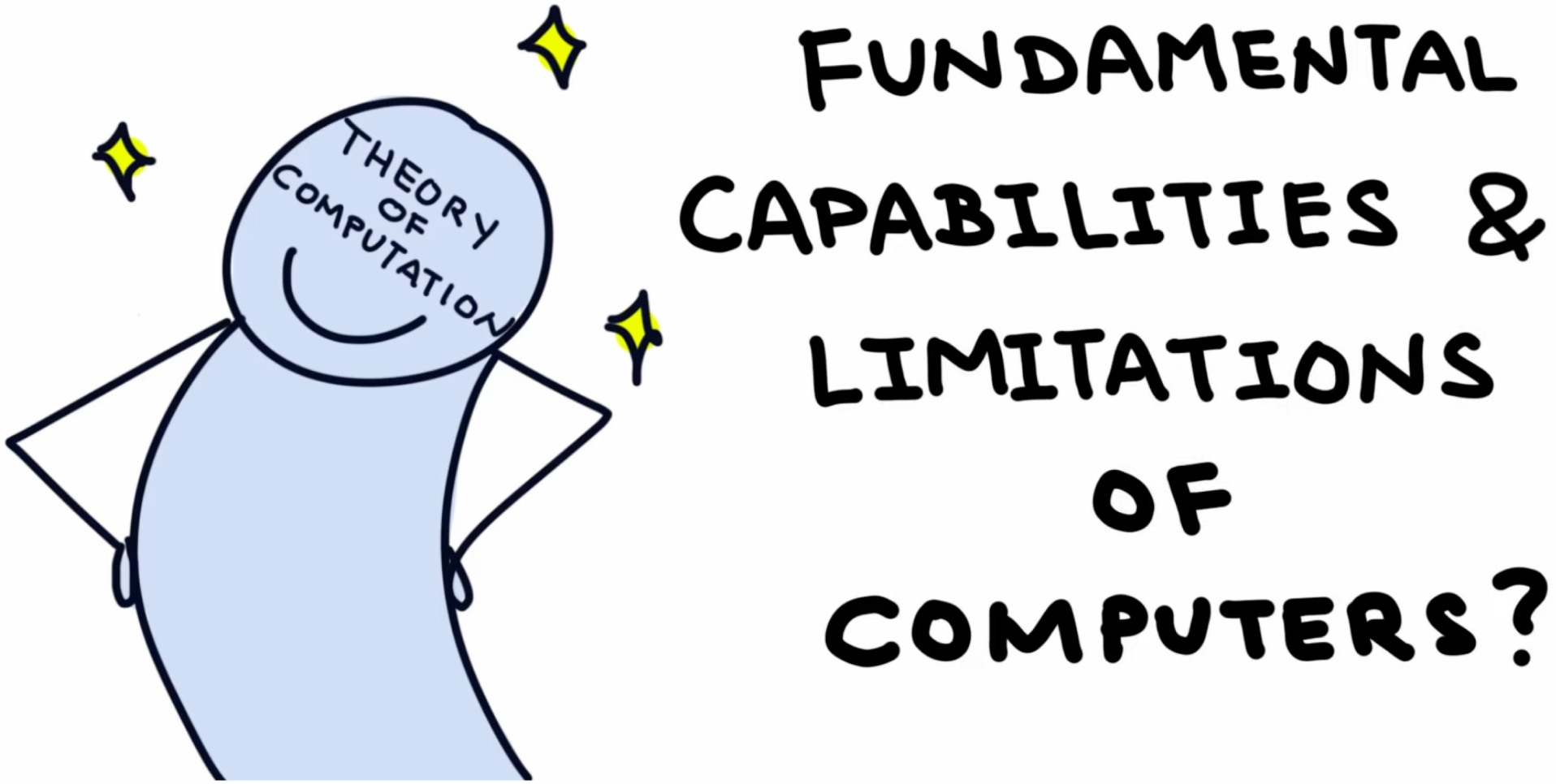
Introduction



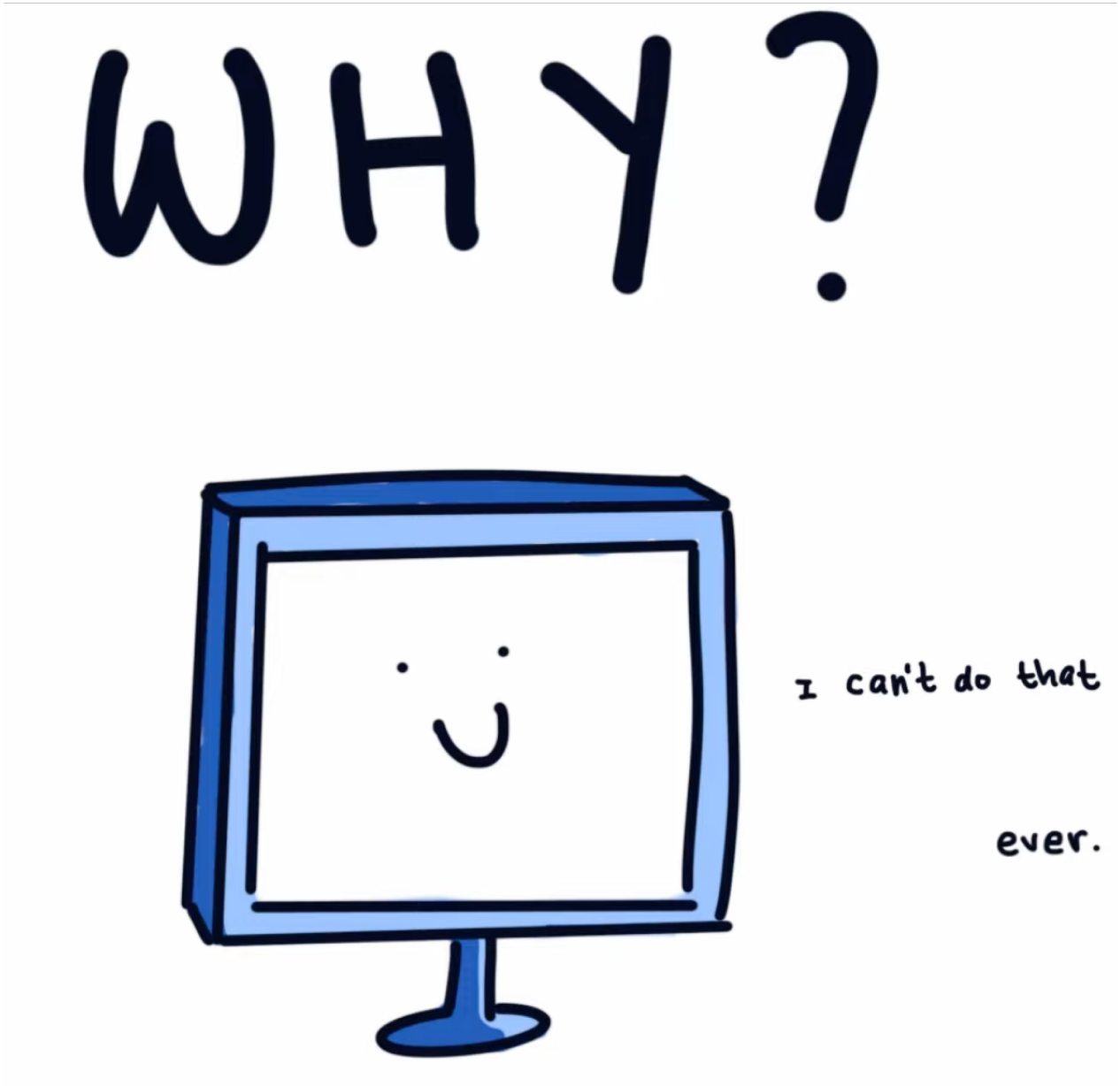
Introduction



Introduction

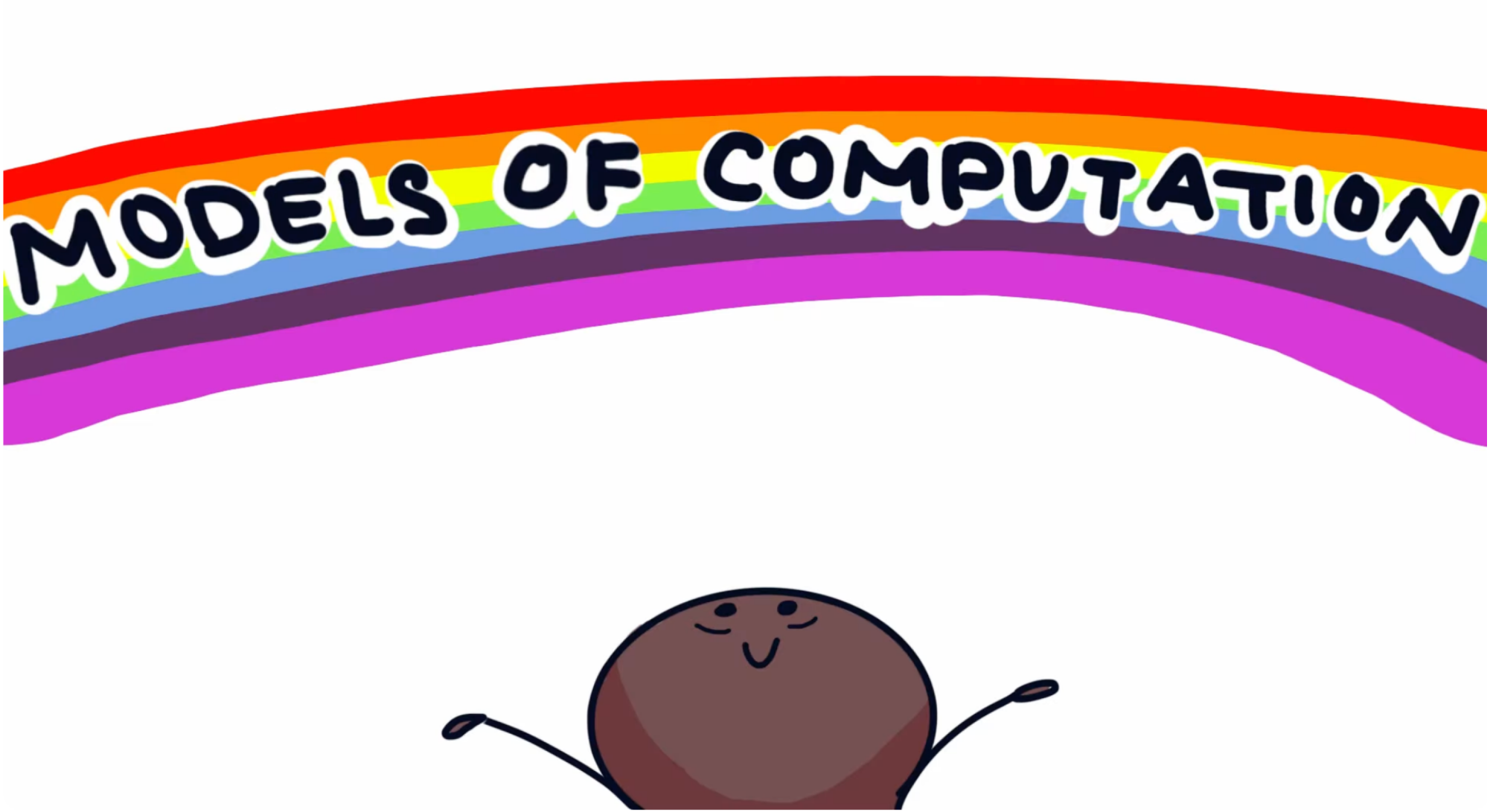


Introduction

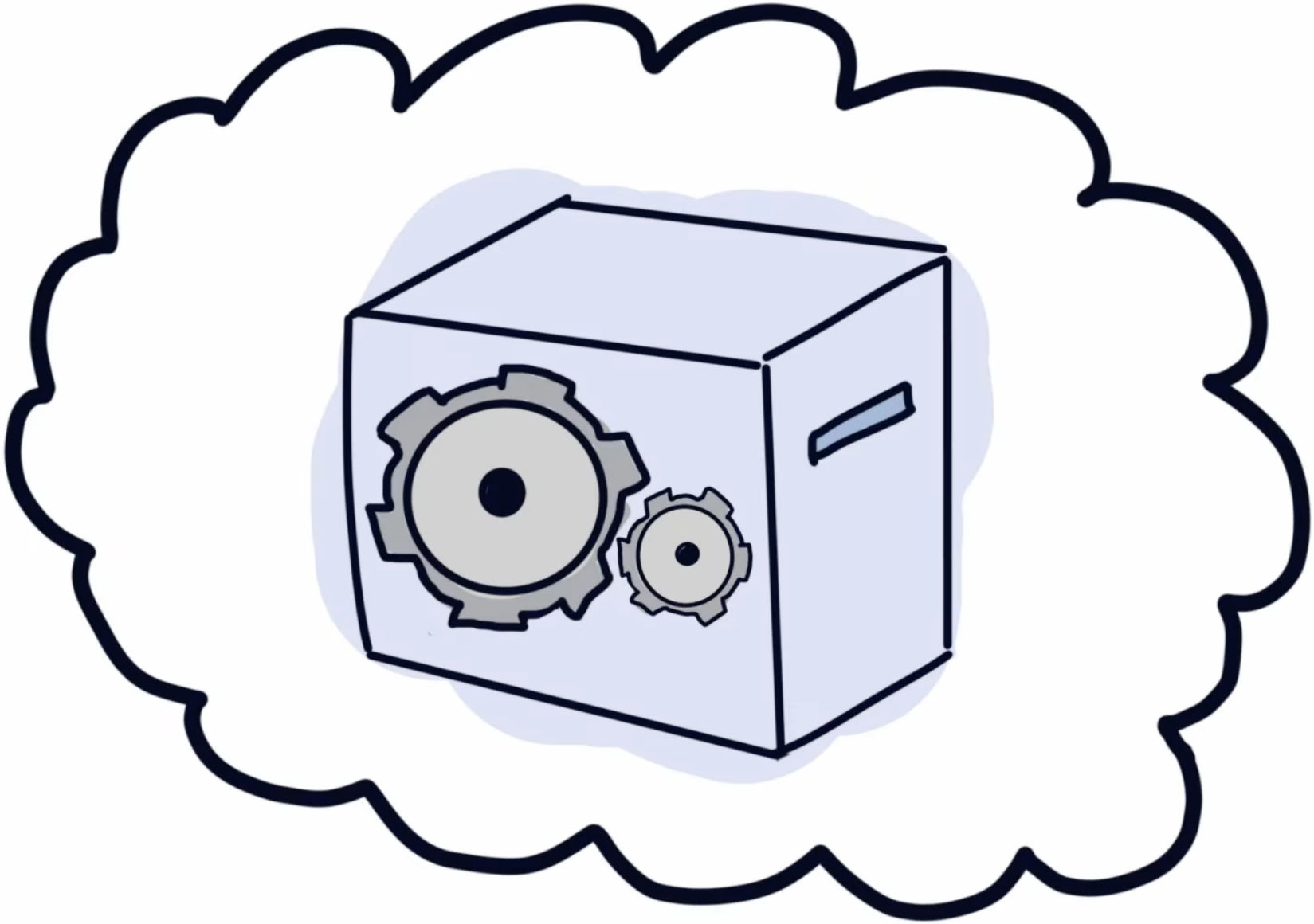


Introduction

Introduction



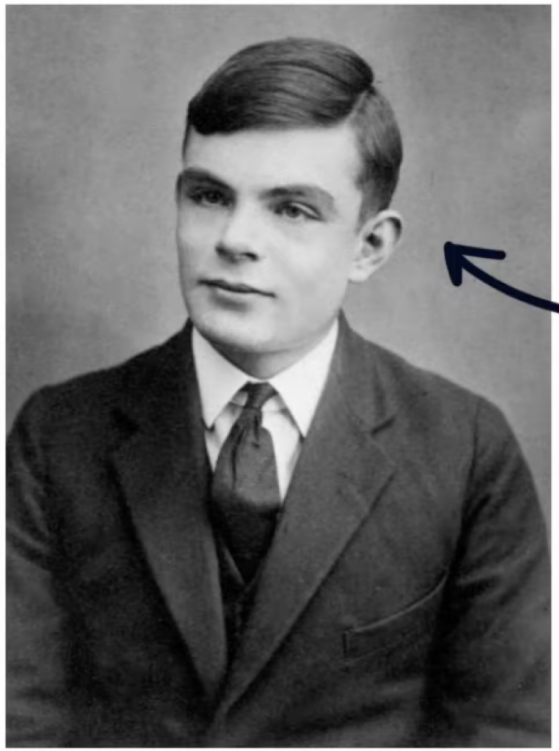
Introduction



Introduction

innovate achieve lead

TURING MACHINE



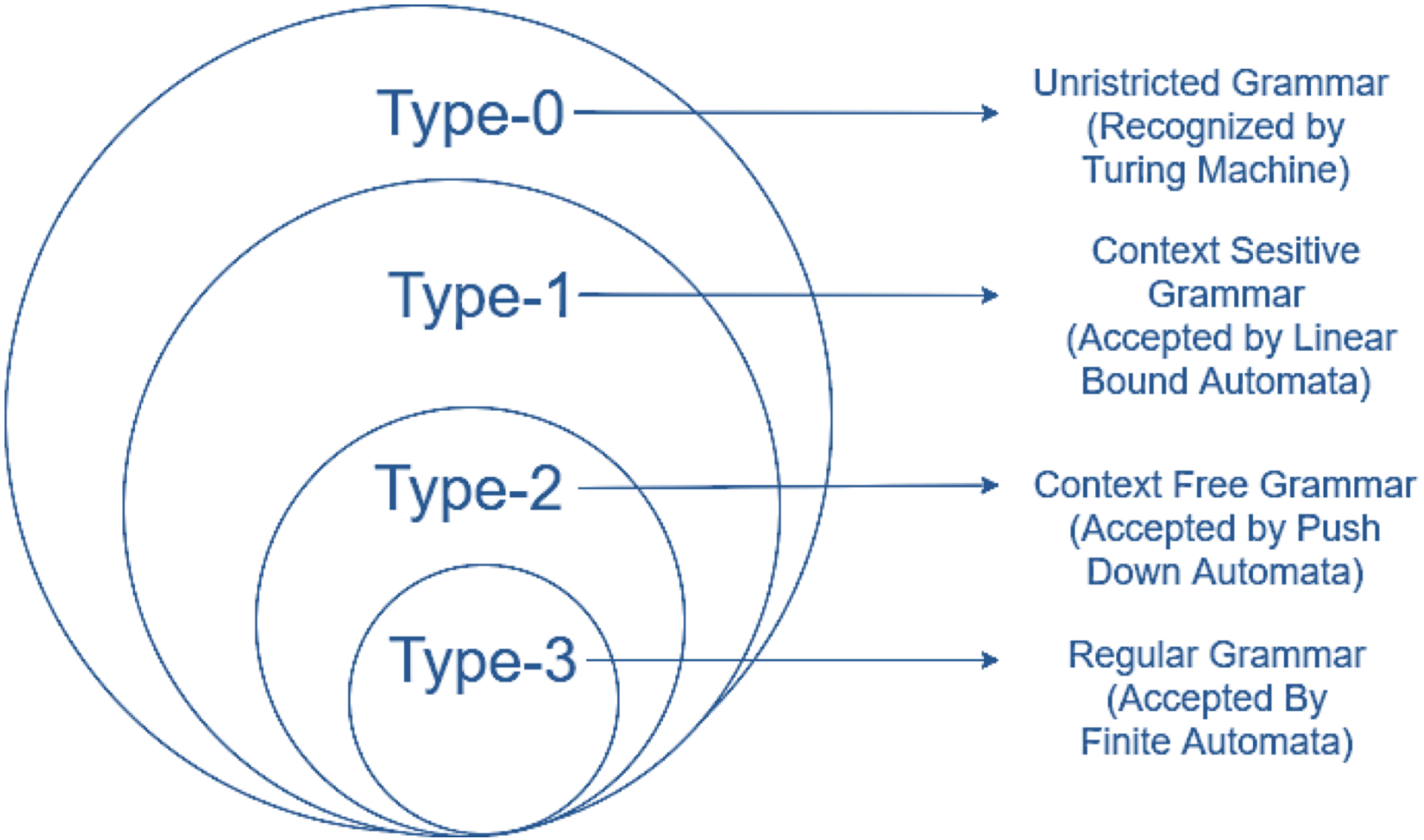
What is Automata Theory?

- **Automata** theory (ToC) is a theoretical branch of Computer Science and Mathematics
  - deals with the logic of computation with respect to simple machines, referred to as automata.

- Automata* enables scientists to understand how machines compute the functions and solve problems.

- Automata originated from the word “Automaton” which is closely related to “Automation”.

What is Automata Theory?



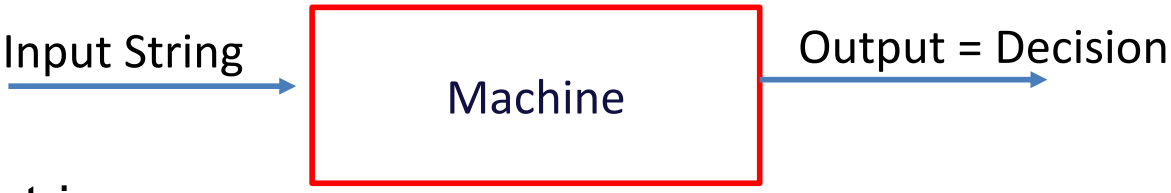
Why Study Formal Language?

- Connected...
  - ...to many other branches of knowledge
- Rigorous...
  - ...mathematics with many open questions at the frontiers
- Useful...
  - ...with many applications in computer systems, particularly in programming languages and compilers
- Accessible...
  - ...no advanced mathematics required
- Stable...
  - ...the basics have not changed much in the last thirty years

Why Study Formal Language?

- Languages (used by humans)
  - Alphabet, Words, Sentences -> Convey some information

- Formal Language
  - Language is a **set** of strings with some property
  - Alphabets, Strings, Sets of Strings (called Languages)



  - Applications
    - Recognize strings
    - Compilers
  - What can be computed? Is there some computation that cannot be performed by a computer ?

Languages (Formal)

- 1.1 Alphabets
- 1.2 Strings
- 1.3 Languages

Formal Language, chapter 1, slide 5

Alphabets

- An *alphabet* is any finite (non-empty) set of symbols
  - {0,1}: binary alphabet
  - {0,1,2,3,4,5,6,7,8,9}: decimal alphabet
  - ASCII, Unicode: machine-text alphabets
  - Or just {a,b}: enough for many examples
  - {}: a legal but not usually interesting alphabet

- We will usually use Σ as the name of the alphabet we're considering, as in Σ = {a,b}

Outline

- 1.1 Alphabets
- 1.2 Strings
- 1.3 Languages

Strings

- A *string* is a finite sequence of zero or more symbols
- Length of a string: |*abbb*| = 4
- *A string over the alphabet* Σ means *a string, all of whose symbols are in* Σ
  - if Σ = {a,b} the set of all strings of length 2 over Σ is {*aa*, *ab*, *ba*, *bb*}
- Empty String:
  - String with 0 occurrences of symbols
  - The empty string is denoted as ε
  - Like "" in some programming languages
  - | ε | = 0

Distinction between ε, Ø

- $\phi, \{\}$ are empty sets (sets containing no elements) $|\phi| = 0$
- ε is empty string (like "" in programming langs) Len
- $\phi \neq \varepsilon$, $\varepsilon \notin \phi$
- $\phi \neq \{\varepsilon\}$
- $S = \{\varepsilon\}$ is set containing an empty string; $|S| = 1$

Concatenation

- The *concatenation* of two strings *x* and *y* is the string containing all the symbols of *x* in order, followed by all the symbols of *y* in order
- We show concatenation just by writing the strings next to each other
- If $x = abc$ and $y = def$, then $xy = abcdef$
- For any string x, $\varepsilon x = x\varepsilon = x$

Exponents/Power of a string

- Exponent/Power *n* concatenates
  - a string x with itself *n* times
  - $x^0 = \varepsilon$ $(n = 0)$
  - $x^n = x^{n-1}x$ $(n >= 1)$

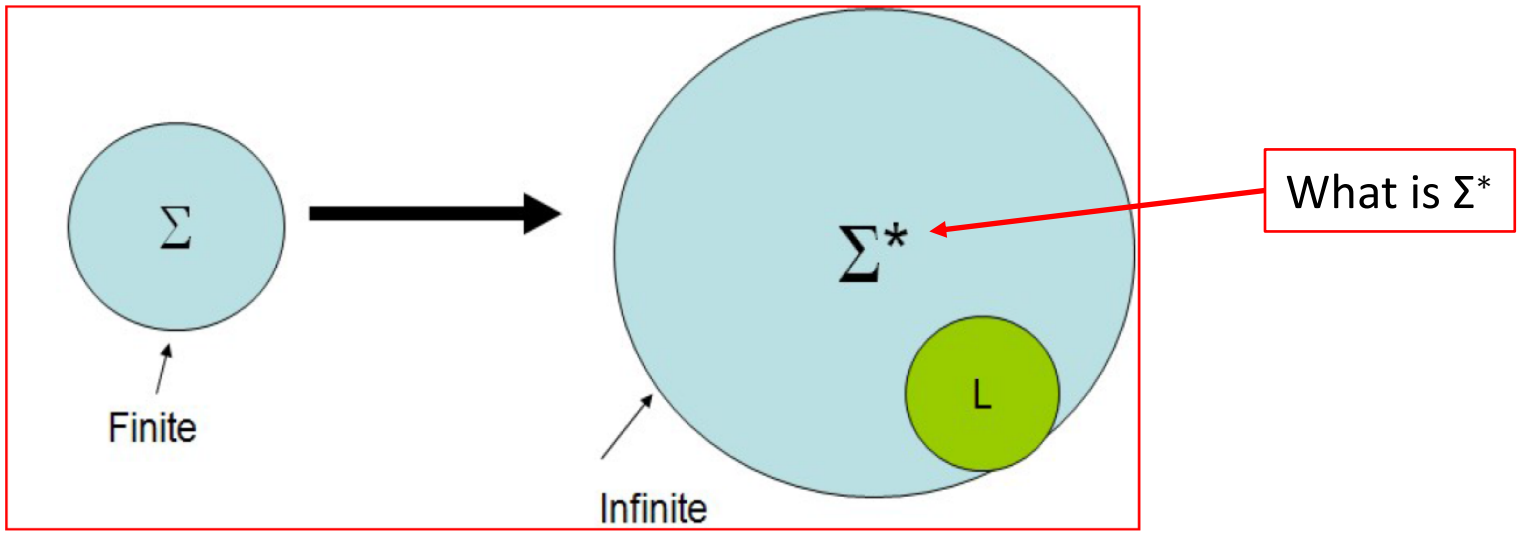
- Eg.
  - If $x = ab$, then
    - $x^0 = \varepsilon$
    - $x^1 = x^0x = \varepsilon x = x = ab$
    - $x^2 = x^1x = xx = abab$, etc.
  - We use parentheses for grouping exponentiations (assuming Σ does not contain the parentheses)
    - $(ab)^7 = abababababababab$

Outline

- 1.1 Alphabets
- 1.2 Strings
- 1.3 Languages

Languages (Formal)

- A *language L* is a **set of strings** over some finite alphabet (Σ)
  - *Not* restricted to finite sets: in fact, finite sets are not usually interesting languages
  - All our alphabets are finite, and all our strings are finite, but most of the languages we're interested in are infinite



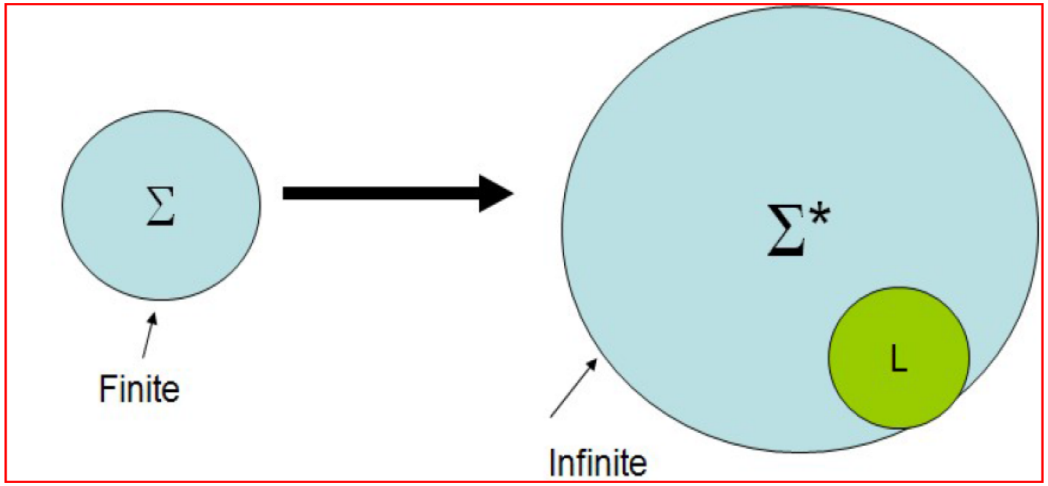
- *L* can be finite or (countably) infinite

Powers of an alphabet (Σ^k)

- If Σ is an alphabet, set of all strings of a certain length *k* from that alphabet is denoted by using the exponential notation:
  - Σ^k: the set of strings of length *k*, each of whose is in Σ
  - Σ^0 : {ε}, regardless of what alphabet Σ is. ε is the only string of length 0
  - $\Sigma^1 = \Sigma^0 \cdot \Sigma = \{\varepsilon\}.\Sigma$ Strings of length 1 (Concatenation Operation)
  - $\Sigma^2 = \Sigma^1 \cdot \Sigma = \Sigma.\Sigma$ Strings of length 2
  - $\Sigma^n = \Sigma^{n-1} \cdot \Sigma = \{\omega | \omega = xy \text{ and } x \in \Sigma^{n-1} \text{ and } y \in \Sigma\}$
  - Eg: If Σ = {0, 1}, then:
    - $\Sigma^0 = \{\varepsilon\}$
    - $\Sigma^1 = \{0, 1\}$
    - $\Sigma^2 = \{00, 01, 10, 11\}$
    - $\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$
  - Note: Distinction between Σ and Σ^1:
    - Σ is an alphabet; its members 0 and 1 are symbols
    - Σ^1 is a set of strings; its members are strings (each one of length 1)

Kleene Closure/Star (Σ^*)

- The set of all possible strings (including ε) over Σ is denoted by Σ^*. (mother of all languages)
- $\Sigma^* = \{x_1, \ldots, x_n | n \geq 0 \text{ and } x_i \in \Sigma \text{ for all } i\}$
- $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots \Sigma^n \cup \ldots = \bigcup_0^\infty \Sigma^i$
- $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \ldots$
- Thus: $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$
- $\{0, 1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$
- if Σ = Φ (ie. {}, empty alphabet), then $\Sigma^* = \{\varepsilon\}$



- What is Σ^*
  - set of all possible strings over Σ
- What is Language *L*
  - $\{w \in \Sigma^* | w \text{ has some property}\}$
  - **Set** of strings

Language Egs

- Σ={*a*}. So {*a*}* is the set of all strings of zero or more a's
  = {ε, *a*, *aa*, *aaa*, …} = { $a^n | n >= 0$ }

- {*a*,*b*}* is the set of all strings of zero or more symbols, each of which is either *a* or *b* = {ε, *a*, *b*, *aa*, *bb*, *ab*, *ba*, *aaa*, …}

- $x \in \Sigma^*$ means *x* is a string over Σ

- $L = \Sigma^*$ – The mother of all languages!

- *L* = {*a*, *ab*, *aab*} – A finite language.

Language Egs

- $L = \{a^n b^n : n \geq 0\} = \{\varepsilon, ab, aabb, aaabbb, \ldots\}$
  (equal no. of *a*'s & *b*'s)

- $L = \{w | n_a(w) \text{ is even}\}$;i.e. the no. of occurrences of *a* in *w* is even all strings with even number of a's.

- $L = \{w \mid w = w^R\}$; All strings which are the same as their reverses (palindromes).

- $L = \{w \mid w = xx\}$; All strings formed by duplicating some string. once.

- $L = \{w \mid w$ is a syntactically correct Java program $\}$

- if $\Sigma=\{a\}$, $L = \{ a^{2n+1} \mid n \geq 0\} = \{ a, aaa, aaaaa, \ldots\}$

Operations on Languages

- Since languages are sets, all usual set operations such as intersection and union, etc. are defined.
  - Union
  - Intersection
  - Concatenation
  - Complement
  - Closure

Union of Languages

Definition: The union of two languages L_1 and L_2 is

$$L_1 \cup L_2 = \{ x \mid x \in L_1 \text{ or } x \in L_2 \}$$

- $L_1 \cup L_2$ consists of all elements in L_1 or in L_2 (or in both).

Examples:

- If $L_1 = \{ ab, bb \}$ and $L_2 = \{ aa, bb, a \}$,

  then $L_1 \cup L_2 = \{ ab, bb, aa, a \}$.

- If $L_1 = \{ a, ba \}$ and $L_2 = \emptyset$, then $L_1 \cup L_2 = L_1$.

- If $L_1 = \{ a, ba \}$ and $L_2 = \{ \varepsilon \}$, then $L_1 \cup L_2 = \{ \varepsilon, a, ba \}$.

Intersection of Languages

Definition: The intersection of two languages L_1 and L_2 is

$$L_1 \cap L_2 = \{ x \mid x \in L_1 \text{ and } x \in L_2 \},$$

- $L_1 \cap L_2$ consists of elements that are in both L_1 and L_2 .

Definition: Languages L_1 and L_2 are disjoint if $L_1 \cap L_2 = \emptyset$.

Examples:

- Let $L_1 = \{ ab, bb \}$ and $L_2 = \{ aa, bb, a \}$.

  Then $L_1 \cap L_2 = \{ bb \}$.

- Let $L_1 = \{ ab, bb \}$ and $L_2 = \{ aa, ba, a \}$.

  Then $L_1 \cap L_2 = \emptyset$, so L_1 and L_2 are disjoint.

Concatenation of Languages

Definition: The concatenation (or product) of sets L_1 and L_2 is $L_1 \circ L_2 = \{ xy \mid x \in L_1, y \in L_2 \}$.

Remarks:

- $L_1 \circ L_2$ is the set of strings that can be split into 2 parts first part of string is in L_1, and second part is in L_2 .
- Also written as $L_1 L_2$ rather than $L_1 \circ L_2$ to denote concatenation.

Examples:

- If $L_1 = \{ a, aa \}$ and $L_2 = \{ \varepsilon, a, ba \}$, then

$$L_1 \circ L_2 = \{ a, aa, aba, aaa, aaba \},$$
$$L_2 \circ L_1 = \{ a, aa, aaa, baa, baaa \}.$$

$aba \in L_1 \circ L_2$, but $aba \notin L_2 \circ L_1$. Thus, $L_1 \circ L_2 \neq L_2 \circ L_1$.

- If $L_1 = \{ ab, ba \}$ and $L_2 = \emptyset$, then $L_1 \circ L_2 = L_2 \circ L_1 = \emptyset$.
- If $L_1 = \{ ab, ba \}$ and $L_2 = \{\varepsilon\}$, then $L_1 \circ L_2 = L_2 \circ L_1 = L_1$.

Complement of a Language

Definition: The complement of a Language L is $\bar{L} = \{ x \mid x \notin L \}$.

- $\bar{L}$ is the set of all elements under consideration that are *not* in L.

Example:

- Let L be set of strings over alphabet $\Sigma = \{a, b\}$ that begin with symbol b.
  - Then $\bar{L}$ is set of strings over Σ that do not begin with symbol b,
  - i.e.,$\bar{L} = \Sigma^* - L$.

- $\bar{L}$ is not the set of strings over Σ that begin with the symbol a
  - $\varepsilon \in \bar{L}$ (because ε does not begin with a).

Kleene Closure of Languages

Definition: The Kleene closure of L is

$$L^* = L^0 \cup L^1 \cup L^2 \cup \cdots L^n \cup \cdots = \bigcup_0^\infty L^i$$

- $L^0 = \varepsilon$ ($n = 0$)
- $L^n = L^{n-1}L$ ($n >= 1$) (i.e. concatenation of L^{n-1} and L)

Remarks:

- L^* is the set of all strings formed by concatenating zero or more strings from L. (same string from L can be used more than once).

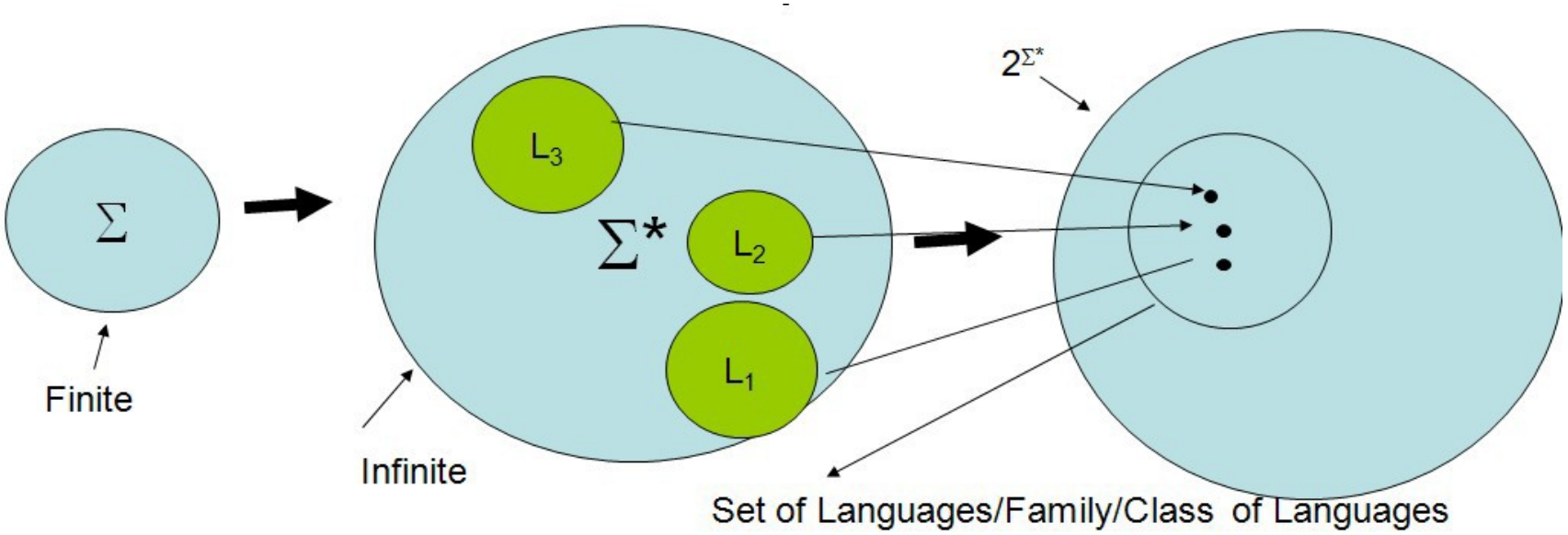
Examples:

- If $L = \{ ba, a \}$ then
  - $L^0 = \{ \varepsilon \}$
  - $L^1 = L^0L = \varepsilon L = \varepsilon \{ ba, a \} = \{ ba, a \}$
  - $L^2 = L^1L = LL = \{ ba, a \} \{ ba, a \} = \{ baba, baa, aba, aa \}$
  - $L^* = \{ \varepsilon, a, aa, ba, aaa, aba, baa, aaaa, aaba, \ldots \}$.

- if $L = \Phi$ (empty language) then $L^* = \{ \varepsilon \}$

Sets of Languages

- The power set of Σ^*, the set of all its subsets, is denoted as 2^{Σ^*}



The power set $P(S)$ of a set S is $P(S) = 2^S = \{ A \mid A \subseteq S \}$.

- $P(S)$ is the set of all possible subsets of S.

Example: If $S = \{ a, bb \}$, then $P(S) = \{ \emptyset, \{a\}, \{bb\}, \{a, bb\} \}$.

If $|S| < \infty$, then $|P(S)| = 2^{|S|}$

Since Σ^* is (countably) ∞, the no. of Languages is uncountably infinite.

Thank you!

Tutorial - I

Answer the questions below for the sets of strings:

- C = {ε, aab, baa},
- D = {bb, aab},
- E = {ε},
- F = ∅.

Except for parts (c) and (h), write out any sets with the elements in string order, i.e., shorter strings appear before longer strings, and strings of the same length are in alphabetical order. For any infinite sets, list only the first 8 elements in string order, but be sure to also indicate that the set is infinite by including 3 dots after listing the first 8 elements. For any set S of strings, we define

$S^* = \{ x_1x_2 \cdot \cdot \cdot x_k \mid k \geq 0 \text{ and each } x_i \in S \}$,

$S+ = \{ x_1x_2 \cdot \cdot \cdot x_k \mid k \geq 1 \text{ and each } x_i \in S \}$,

where the concatenation of k = 0 strings is the empty string ε.

Tutorial - I

(a) What is $D \cup C$?

(b) What is $C \cup F$?

(c) What is $C \cap D$?

(d) What is $D \circ C$?

(e) What is $C \circ E$?

(f) What is $D \circ D \circ D$?

(g) What is P(C), the power set of C ?

(h) What is $D - C$?

(i) What is C+ ?

(j) What is F^* ?

(k) Is $E \subseteq C$?

(l) Is $D \subseteq C$?

Tutorial II

innovate achieve lead

2. For each of the following parts, give an example satisfying the given conditions. Give a brief explanation for each of your examples.

(a) Give an example of a set S of strings such that $S^* = S^+$.

(b) Give an example of a set S of strings such that $S^* \neq S^+$.

(c) Give an example of a set S of strings such that $S = S^*$.

(d) Give an example of a set S of strings such that $S \neq S^*$.

(e) Give an example of a set S of strings such that S^* is finite