BU CS 332 – Theory of Computation

Link to polls:
https://forms.gle/XkxqNuX8EGJenf7h7

Lecture 2:

- Parts of a Theory of Computation

- Sets, Strings, and Languages

Reading:

Sipser Ch 0

Reminders:
- HW0 due + HW1 out tomorrow night (Thu, 11:59PM)

Mark Bun

January 24, 2024

What makes a good theory?

- General ideas that apply to many different systems
- Expressed simply, abstractly, and precisely
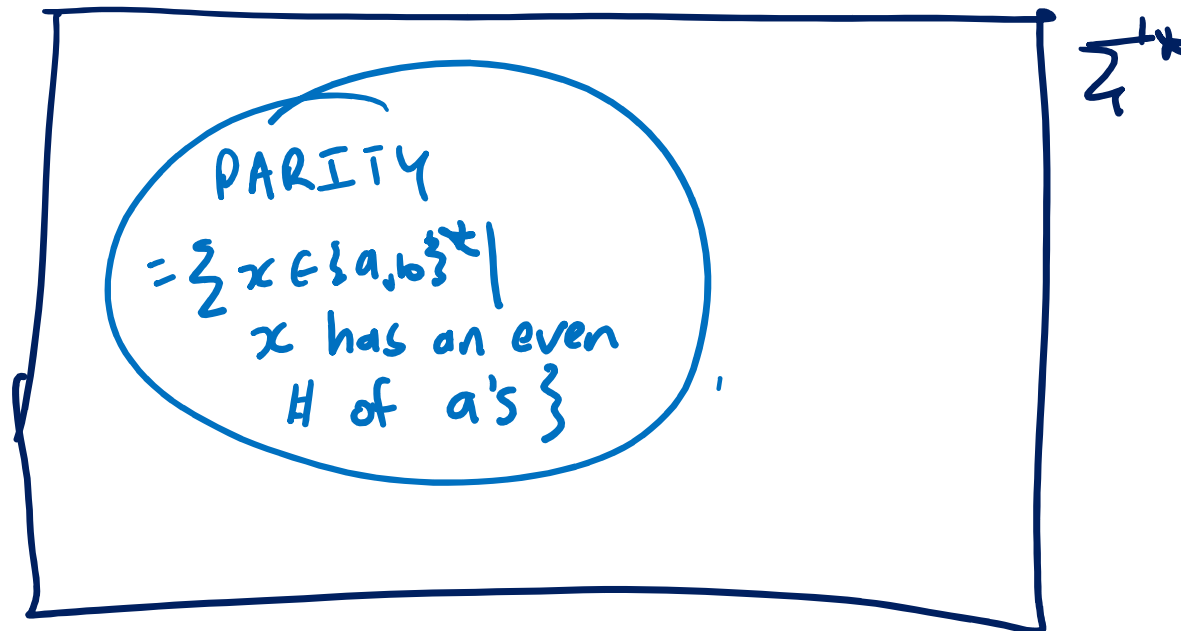

Parts of a Theory of Computation

- Models for machines (computational devices)
- Models for the problems machines can be used to solve
- Theorems about what kinds of machines can solve what kinds of problems, and at what cost

What is a (Computational) Problem?

For us: A problem will be the task of determining whether a *string* is in a *language*

E.g.   Parity: Given a string of a's and b's, does it contain an even number of a's?

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

PARITY
$= \{x \in \{a, b\}^* \mid x$ has an even # of a's $\}$

Σ^*

What is a (Computational) Problem?

For us: A problem will be the task of determining whether a *string* is in a *language*

- **Alphabet**: A finite set Σ Ex. $\Sigma = \{a, b\}$
- **String:** A finite concatenation of alphabet symbols
  Ex. $bba, ababb$

  ε denotes empty string, length 0

  Σ^* = set of all strings using symbols from Σ

  Ex. $\{a, b\}^* = \{\varepsilon, a, b, aa, ab, ba, bb, \dots\}$

- **Language:** A set $L \subseteq \Sigma^*$ of strings

Examples of Languages

Parity: Given a string consisting of a's and b's, does it contain an even number of a's?

$\Sigma = \{a, b\}$ $\qquad$ $L = \{x \in \{a, b\}^* \mid x \text{ has an even \# of a's}\}$

Primality: Given a natural number x (represented in binary), is x prime?

$\Sigma = \{0, 1\}$ $\qquad$ $L = \{x \in \{0, 1\}^* \mid x \text{ is the binary rep. of a prime}\}$

Halting Problem: Given a C program, can it ever get stuck in an infinite loop?

$\Sigma = $ Extended ASCII $\qquad$ $L = \{P \in \Sigma^* \mid P \text{ describes a C program}$
$\text{that loops forever on some input}\}$

Machine Models

Computation is the processing of information by the **unlimited application** of a **finite set** of operations or rules

Input
a	b	a	a		

...
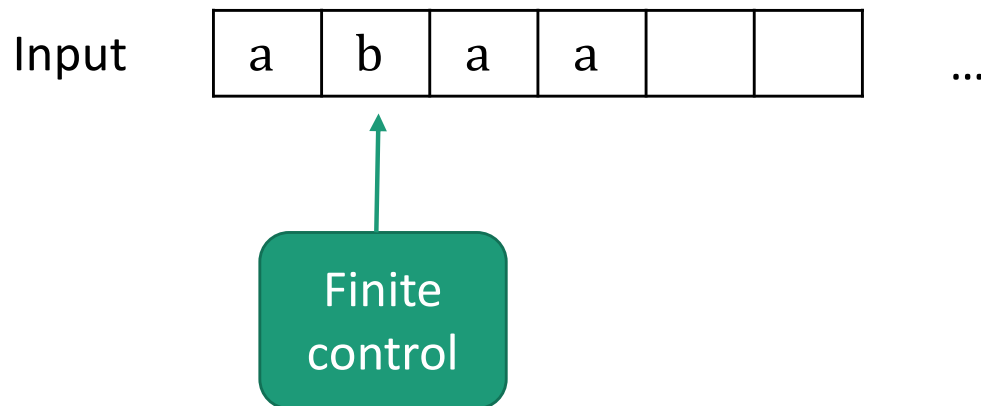
Finite control

Abstraction: We don't care how the control is implemented. We just require it to have a finite number of states, and to transition between states using fixed rules.

Machine Models

- <u>Finite Automata (FAs):</u> Machine with a finite amount of unstructured memory

Input

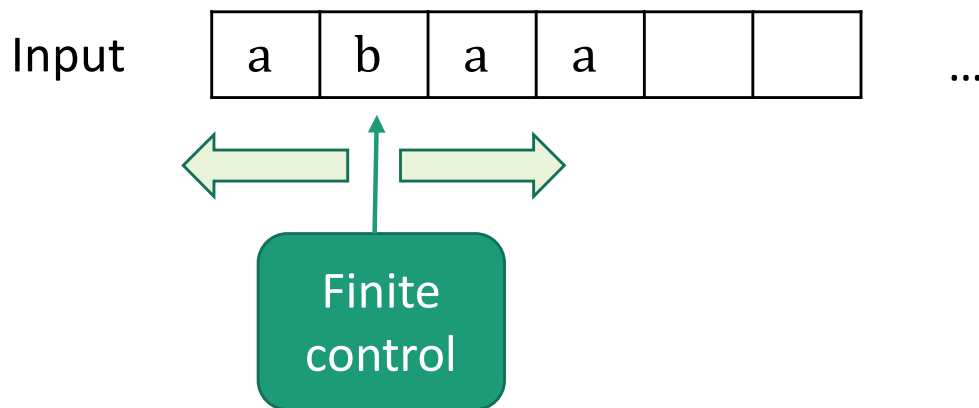| a | b | a | a | | |

...

Finite control
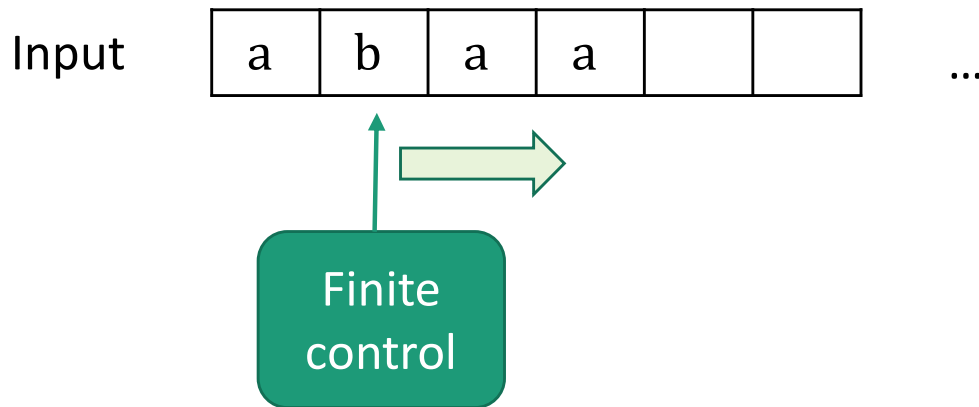
Control scans left-to-right
Can check simple patterns
Can't perform unlimited counting

Useful for modeling chips, simple control systems, choose-your-own adventure games…

Machine Models

- <u>Turing Machines (TMs):</u> Machine with unbounded, unstructured memory

Input

| a | b | a | a | | |

...

Finite control

Control can scan in both directions
Control can both read and <u>write</u>

Model for general sequential computation
Church-Turing Thesis: Everything we intuitively think of as "computable" is computable by a Turing Machine

What theorems would we like to prove?

We will define <u>classes</u> of languages based on which machines can solve the associated computational problems

Inclusion: Every language recognizable by a FA is also recognizable by a TM

Non-inclusion: There exist languages recognizable by TMs which are not recognizable by FAs

Completeness: Identify a "hardest" language in a class

Robustness: Alternative definitions of the same class

Ex. Languages recognizable by FAs = regular expressions

Why study theory of computation?

- You'll learn how to formally reason about computation
- You'll learn the technology-independent foundations of CS

Philosophically interesting questions:

- Are there well-defined problems which cannot be solved by computers?
- Can we always find the solution to a puzzle faster than trying all possibilities?
- Can we say what it means for one problem to be "harder" or "no harder" than another?

Why study theory of computation?

- You'll learn how to formally reason about computation
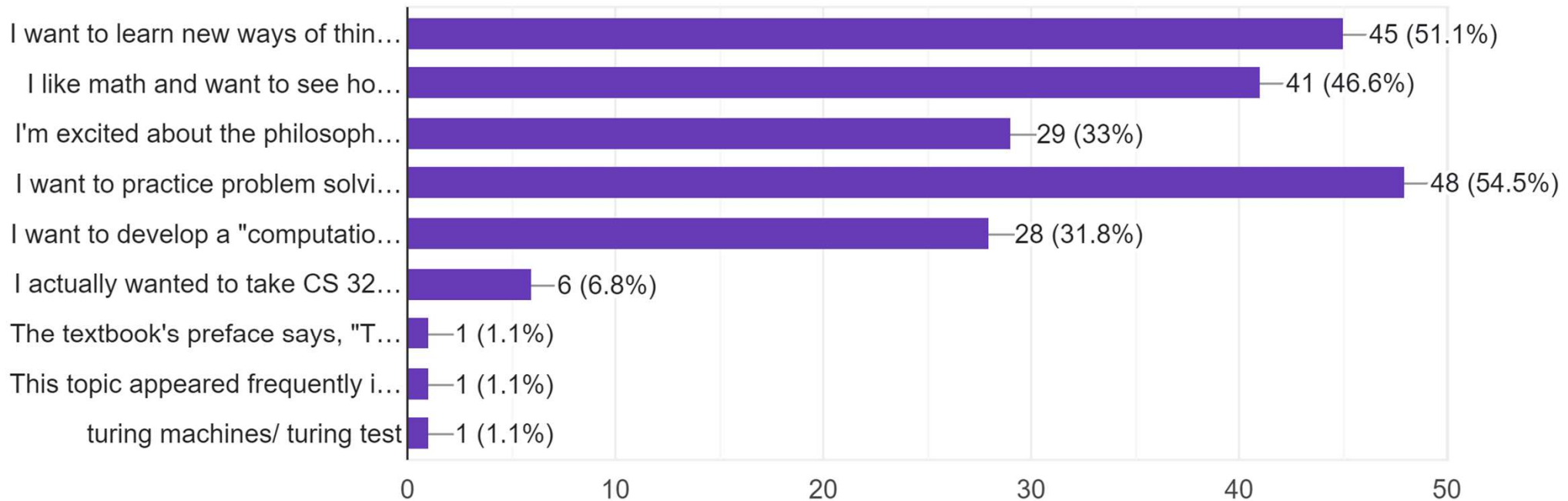- You'll learn the technology-independent foundations of CS

Connections to other parts of science:

- Finite automata arise in compilers, AI, coding, chemistry
  https://cstheory.stackexchange.com/a/14818
- Hard problems are essential to cryptography
- Computation occurs in cells/DNA, the brain, economic systems, physical systems, social networks, etc.

What appeals to you about the theory of computation?

Why do you want to study the theory of computation?

88 responses

Response	Count
I want to learn new ways of thin...	45 (51.1%)
I like math and want to see ho...	41 (46.6%)
I'm excited about the philosoph...	29 (33%)
I want to practice problem solvi...	48 (54.5%)
I want to develop a "computatio...	28 (31.8%)
I actually wanted to take CS 32...	6 (6.8%)
The textbook's preface says, "T...	1 (1.1%)
This topic appeared frequently i...	1 (1.1%)
turing machines/ turing test	1 (1.1%)

Why study theory of computation?

Practical knowledge for developers

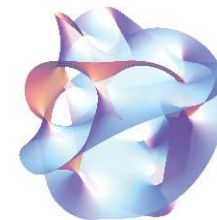"Boss, I can't find an efficient algorithm. I guess I'm just too dumb."

"Boss, I can't find an efficient algorithm because no such algorithm exists."

Will you be asked about this material on job interviews?
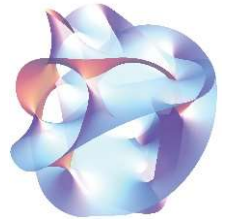
No promises, but a true story...

More about strings and languages

String Theory

- **Symbol:** Ex. $a, b, 0, 1$

- **Alphabet**: A finite set Σ of symbols $\qquad$ Ex. $\Sigma = \{a, b\}$

- **String:** A finite concatenation of alphabet symbols
  Ex. $bba, ababb$

  ε denotes empty string, length 0

  Σ^* = set of all strings using symbols from Σ

  $\qquad$ Ex. $\{a, b\}^* = \{\varepsilon, a, b, aa, ab, ba, bb, \dots\}$

- **Language:** A set $L \subseteq \Sigma^*$ of strings

String Theory

- **Length** of a string, written $|x|$, is the number of symbols

Ex. $|abba| =$ 4 $\qquad |\varepsilon| =$ 0

$x \circ y$

- **Concatenation** of strings x and y, written xy, is the symbols from x followed by the symbols from y

Ex. $x = ab, y = ba \qquad \Rightarrow \qquad xy =$ abba

$\quad x = ab, y = \varepsilon \qquad \Rightarrow \qquad xy =$ ab

- **Reversal** of string x, written x^R, consists of the symbols of x written backwards

Ex. $x = aab \qquad \Rightarrow \qquad x^R =$ baa

Fun with String Operations

What is $(xy)^R$?

Ex. $x = \text{aba}, y = \text{bba} \quad \Rightarrow \quad xy = \text{ababba}$

$\Rightarrow \quad (xy)^R = \underbrace{\text{abb}}_{y^R}\underbrace{\text{aba}}_{x^R}$

a) $x^R y^R$

b) $y^R x^R$

c) $(yx)^R$

d) xy^R

Fun [Proofs] with String Operations

Claim: $(xy)^R = y^R x^R$

Proof: Let $x = x_1 x_2 \ldots x_n$ and $y = y_1 y_2 \ldots y_m$

Then $(xy)^R = (x_1 x_2 \ldots x_n \, y_1 y_2 \ldots y_m)^R$

$= y_m y_{m-1} \ldots y_2 y_1 \, x_n x_{n-1} \ldots x_2 x_1$

$= y^R x^R$

Not even the most formal way to do this:
1. Define string length recursively
2. Prove by induction on $|y|$

More than what's needed for us

Languages

A language L is a set of strings over an alphabet Σ

   i.e., $L \subseteq \Sigma^*$

Languages = computational (decision) problems

<u>Input:</u> String $x \in \Sigma^*$

<u>Output:</u> Is $x \in L$? (Yes or No?)

                  Accept or Reject

Some Simple Languages

$$\Sigma = \{0, 1\} \qquad\qquad \Sigma = \{a, b, c\}$$

$\emptyset$ (Empty set)

$\phi = \{\ \}$ $\qquad\qquad$ $\phi = \{\ \}$

Σ^* (All strings)

$\{0,1\}^* = \{\epsilon, 0, 1, 00, 01, ..\}$

$\{a,b,c\}^k =$
$\{\epsilon, a, b, c, aa, ab, ac, ..\}$

$\Sigma^n = \{x \in \Sigma^* \mid |x| = n\}$
(All strings of length n)

$n = 2$
$\{0,1\}^2 = \{00, 01, 10, 11\}$

$n = 2$
$\{a,b,c\}^2 = \{aa, ab, ... cc\}$

$n = 3$
$\{0,1\}^3 = \{000, 001, 010, 011,$
$\qquad\qquad 100, 101, 110, 111\}$

$n = 3$
$\{a,b,c\}^3 = \{aaa, aab,...$
$\qquad\qquad ccc\}$

Some More Interesting Languages

- L_1 = The set of strings $x \in \{a, b\}^*$ that have an equal number of a's and b's $a\,bba \in L_1$ $baa \notin L_1$

$$L_1 = \{\, x \in \{a,b\}^* \mid \# \text{ of a's in } x \text{ is equal to \# of b's} \,\}$$
$$\text{in } x$$

- L_2 = The set of strings $x \in \{a, b\}^*$ that start with (0 or more) a's and are followed by an equal number of b's

$$L_2 = \{\, x \in \{a,b\}^* \mid \cdots \,\} = \{\, a^n b^n \mid n \geq 0 \,\}$$

- L_3 = The set of strings $x \in \{0, 1\}^*$ that contain the substring "0100" $101001 \in L_3$ $001010 \notin L_3$

NOT THE SAME: $\{\, y\,0100\,y \mid y \in \{0,1\}^* \,\}$

$$L_3 = \{\, x \in \{0,1\}^* \mid x = y\,0100\,z, \; y \in \{0,1\}^*, \; z \in \{0,1\}^* \,\}$$
$$= \{\, x \in \{0,1\}^* \mid 0100 \text{ is a substring of } x \,\} = \{\, y\,0100\,z \mid y, z \in \{0,1\}^* \,\}$$

Some More Interesting Languages

- $L_4 = $ The set of strings $x \in \{a, b\}^*$ of length at most 4

$$L_4 = \{ x \in \{a, b\}^* \mid |x| \leq 4 \}$$

- $L_5 = $ The set of strings $x \in \{a, b\}^*$ that contain at least two a's

$$L_5 = \{ x \in \{a, b\}^* \mid x = yazal \mid y, z, l \in \{a, b\}^* \}$$
$$= \{ yazal \mid y, z, l \in \{a, b\}^* \}$$

$$L_{42} = \{ x_0 \, a \, x_1 \, a \, x_2 \, a \ldots a \, x_{42} \mid x_0, \ldots, x_{42} \in \{a, b\}^* \}$$
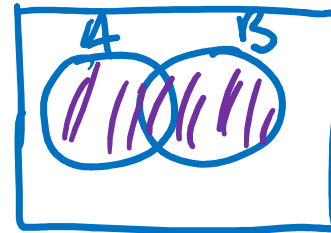
New Languages from Old

L_6 = The set of strings $x \in \{a, b\}^*$ that have an equal number of a's and b's and length greater than 4

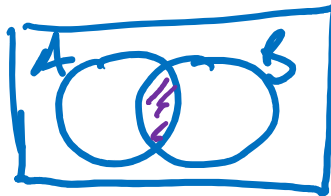Since languages are just sets of strings, can build them using set operations:

$A \cup B$ "union"

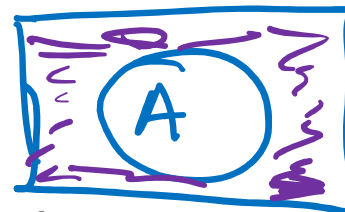$$A \cup B = \{x \mid x \in A \text{ or } x \in B\}$$

$A \cap B$ "intersection"

$$A \cap B = \{x \mid x \in A \text{ and } x \in B\}$$

$\bar{A}$ "complement"

$$\bar{A} = \{x \mid x \notin A\}$$

New Languages from Old

$L_6 = $ The set of strings $x \in \{a, b\}^*$ that have an equal number of a's and b's and have length greater than 4

- $L_1 = $ The set of strings $x \in \{a, b\}^*$ that have an equal number of a's and b's

- $L_4 = $ The set of strings $x \in \{a, b\}^*$ of length at most 4

$$\Rightarrow L_6 = \{ x \in \{a,b\}^* \mid x \in L_1 \text{ and } x \notin L_4 \}$$
$$= L_1 \cap \overline{L_4}$$

Operations Specific to Languages

- **Reverse**: $L^R = \{x^R \mid x \in L\}$

  Ex. $L = \{\varepsilon, a, ab, aab\}$ $\qquad \Rightarrow L^R = \{baa, ba, a, \epsilon\}$
  $\qquad = \{aab, ab, a, \epsilon\}$ $\qquad\qquad = \{\epsilon, a, ba, baa\}$

- **Concatenation:** $L_1 \circ L_2 = \{xy \mid x \in L_1, y \in L_2\}$

  Ex. $L_1 = \{ab, aab\} \qquad L_2 = \{\varepsilon, b, bb\}$
  $\qquad \Rightarrow L_1 \circ L_2 = \{ab, abb, abbb, aab, aabb, aabbb\}$

L_1, L_2 finite: $\quad |L_1 \circ L_2| \leq |L_1 \times L_2| = |L_1| \cdot |L_2|.$

A Few "Traps"

String, language, or something else?

ε	String	Empty String consisting of no characters
$\emptyset$	Could be a Language	Empty set (language) consisting of no strings
$\{\varepsilon\}$	Language	Lang. consisting of only the empty string
$\{\emptyset\}$	Something else:	Set of (possibly) languages

Languages

Languages = computational (decision) problems

<u>Input:</u>   String $x \in \Sigma^*$

<u>Output:</u> Is $x \in L$? (Yes or No? I.e., Accept or Reject?)

The language **recognized** by a program is the set of strings $x \in \Sigma^*$ that it *accepts*

What Language Does This Program Recognize?

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

On input $x = x_1 x_2 \dots x_n$:

count = 0

  For $i = 1, \dots, n$:

    If $x_i = a$:

      $\text{count} = \text{count} + 1$

  If count ≤ 4: accept

  Else: reject

a) $\{x \in \Sigma^* \mid |x| > 4\}$
b) $\{x \in \Sigma^* \mid |x| \leq 4\}$
c) $\{x \in \Sigma^* \mid |x| = 4\}$
d) $\{x \in \Sigma^* \mid x \text{ has more than } 4 \text{ a's}\}$
e) $\{x \in \Sigma^* \mid x \text{ has at most } 4 \text{ a's}\}$
f) $\{x \in \Sigma^* \mid x \text{ has exactly } 4 \text{ a's}\}$