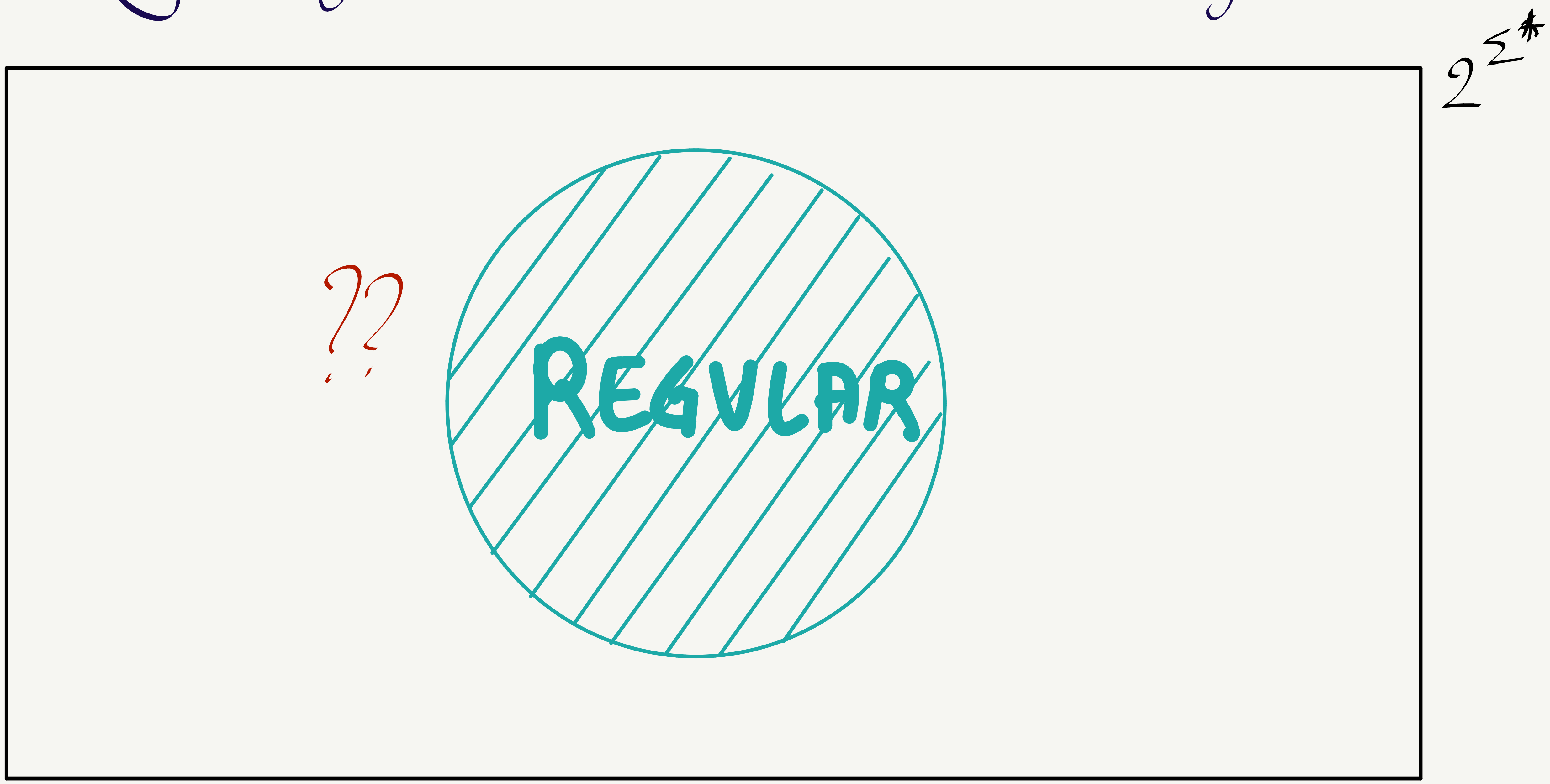
Non-Regular Languages

Recall: We showed how to formally show a language not regular.
Use the Myhill-Nerode theorem (can also show regularity!)
or the Pumping Lemma (can only show non-regularity).

Today: Try to fill in more details into this picture.



Consider the language over $\Sigma = \{a, b\}$

$\mathcal{L} = \{\omega \mid \omega$ has an equal number of 'a's and 'b's$\}$

Instead of directly jumping to Myhill-Nerode or the pumping lemma, try to see what closure properties for regularity can give you.

Suppose, towards a contradiction, that $\mathcal{L}$ is regular.

What is $\mathcal{L} \cap a^*b^*$? $a^*b^* \subseteq \Sigma^*$ is regular, obviously

$\mathcal{L} \cap a^*b^* = \{a^n b^n \mid n \geq 0\}$ is not regular!

Contradiction.

Why are we even bothered about whether a language is regular or not?

Perhaps I want to find all strings which can match each 'a' with a 'b'

Would be handy to be able to write a regex; but not possible!

So now that we know that the language is not regular,

what is possible? Representations? Machine model?

Not regex, but what? Automata?

Grammars

What is a grammar? A set of rules which specifies how to construct well-formed objects in a language.

What does a grammar for $\mathcal{L} = \{a, b\}^*$ look like?

- ε is included in the language
- If a string is included in the language, so is
  - its extension with 'a'
  - its extension with 'b'

So we can write it as follows, in what is called Backus-Naur form (BNF)

$$S ::= \varepsilon \mid Sa \mid Sb$$

$$S ::= \varepsilon \mid aS \mid Sb$$

* What language is coded up by $S ::= \varepsilon \mid aS \mid bS$?

What does a grammar for a mobile number look like?

10 digits preceded by ε, 0, +91-

$$S ::= \langle fd\rangle\langle d\rangle\langle d\rangle\langle d\rangle\langle d\rangle\langle d\rangle\langle d\rangle\langle d\rangle\langle d\rangle\langle d\rangle \mid$$

$$0\langle fd\rangle \qquad '' \qquad \mid$$

$$+91\text{-}\langle fd\rangle \qquad ''$$

$$fd ::= 6 \mid 7 \mid 8 \mid 9$$

$$d ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

A major advantage of grammars is the recursive specification

What language does the following code up?

$$\text{exp} \to \text{exp} + \text{term} \mid \text{exp} - \text{term} \mid \text{term}$$

$$\text{term} \to \text{term} * \text{factor} \mid \text{term} / \text{factor} \mid \text{factor}$$

$$\text{factor} \to \text{int} \mid (\text{exp})$$

$$\text{int} \to \text{digit} \mid \text{digit int}$$

$$\text{digit} \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

The symbols in red are called the terminals of the grammar

exp, term, factor, int, digit are non-terminals

We start at exp and expand non-terminals as we see them.

Expansion of a non-terminal works according to the same rule(s) no matter what symbols surround it. Only one n-t on the left!

So this grammar is called a context-free grammar (CFG).

A CFG is formally described by a 4-tuple: $G = (\underbrace{NT, T, R}_{\text{all finite sets}}, S)$

- the set of non-terminals NT
- the set of terminals T
- the set of production rules R each of the form $X ::= y$ or $X \to y$ where $X \in NT$, and $y \in (NT \cup T)^*$
- the start symbol $S \in NT$

$X \to y_1 \mid y_2 \mid \cdots \mid y_n$

What is the language of G?

$$\mathcal{L}(G) = \{ \omega \mid S \to^* \omega \} \subseteq T^*$$

some finite sequence of rules in R takes S to ω

Going back to our example of

$\mathcal{L} = \{ \omega \mid \omega \text{ contains as many `a's as `b's} \} \subseteq \{a, b\}^*$

Can we write a grammar G s.t. $\mathcal{L} = \mathcal{L}(G)$?

$G = (NT, T, R, S)$

$T = \{a, b\}$

$S ::= \varepsilon \mid aSb \mid bSa \mid SS$, or

$S ::= \varepsilon \mid aSbS \mid bSaS$