Languages → Sets of strings
↳ How described
↳ what is guaranteed

Finite → Regular → ∃ a dfa

Infinite → Depends on how the ∞ set is put together

Depends on the kind of patterns

→ If you build an infinite set from Σ using ∪, ∩, concat, complement, * then regular; (How shown?)

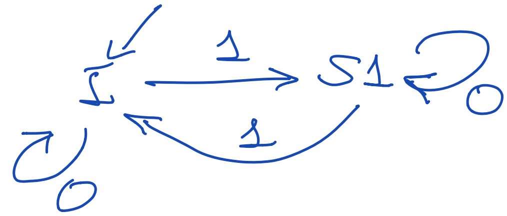
→ Show we can build reg expressions Then do re2nfa and nfa2dfa

→ These algos yet to be introduced.

→ When we define sets like "all strings with an odd # of 1's" we feel in our gut that ∃ a DFA Then we design one keeping "what needs to be kept" in the state name, and we are rewarded when we finish the DFA. Then we would have shown that we have a regular language

→ Summarize ∞ strings into a finite # of bits

Eg "odd 1s"



No need to count the # 1s; just remember the parity

Other ideas such as A0 B1 etc have been shown.

How else can ∞ sets of strings be defined?

By embedding a deep internal tally within that cannot be summarized.

Then prove $\exists$ no dfa with any loop size for an xyz string in L

Eg $L_{if} = \{a^i b^j c^k : \text{ if } i=3 \text{ then } j=k\}$

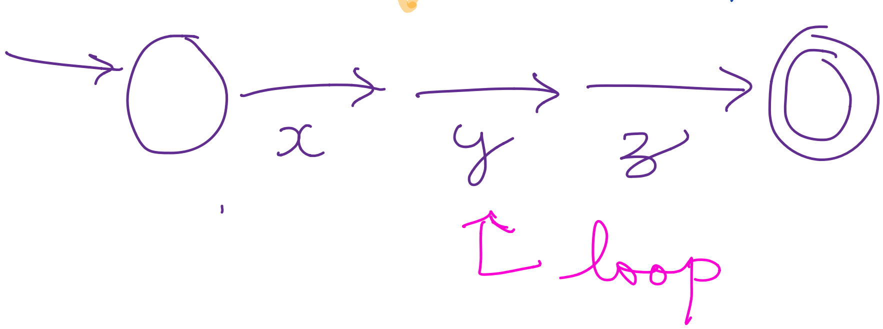
strings $a\,b\,c^2$

$a^3 b^{20} c^{20}$

To rule out all possible DFA, assume very little → Assume what any such DFA must have if it existed!

① Must have, for a long parametric string, this path.

$\Downarrow$



(2) If y repeats and $xy^iz \notin L$ it can't be the DFA but don't assume xy beyond:

- $xy \leq N$
- $y \neq \varepsilon$

So proof must work for all xyz.

NFA make designing DFA easier.

Usually we type-in an RE, get an NFA that almost reads

like the RE
then push a
button to
get a DFA or
min DFA.

$$\text{Write RE} \xrightarrow{\text{re2nfa}} \text{Get NFA} \xrightarrow{\text{nfa2dfa}} \text{Get DFA} \xrightarrow{\text{mindfa}} \text{Get min DFA}$$