MATH 2301

* The language of a regex

If r a regex, $L(r)$ = set of all words that match r.

** Example : Let $\Sigma = \{0,1\}$

If $L = \{w \in \Sigma^* \mid w$ begins with a 0 or w ends with a 1$\}$

Can we find r such that $L = L(r)$?

** Answer :

We know that

$$r = \Big(0 \underbrace{(0|1)^*}_{\text{"anything"}}\Big) \overset{\text{"or"}}{|} \Big(\underbrace{(0|1)^*}_{\text{"anything"}} 1\Big)$$

*** Remark : There can be many different r whose languages are exactly the same.

** Regex construction vs $L(r)$

1) If $r = \phi$: then $L(r) = \phi$

2) If $r = \varepsilon$: then $L(r) = \{\varepsilon\}$

3) If $r = a$ for $a \in \Sigma$: then $L(r) = \{a\}$

4) If $r = r_1 r_2$: then $L(r) = L(r_1) \circ L(r_2)$

5) If $r = r_1 | r_2$: then $L(r) = L(r_1) \cup L(r_2)$

6) If $r = r_1^*$: then $L(r) = L(r_1)^*$

** Examples

1) $r = ((0|1)(0|1)0)^*$

Want $L(r) = L(r_1)^*$, where $r_1 = (0|1)(0|1)0$.

$L(r_1)$ = all 3-letter words that end in a 0.

$L(r)$ = all words with $3k$ letters, for some $k \in \mathbb{N}$, such that every 3rd letter is a 0.

2) $r = \underbrace{(0(0|1)^*0)} \mid \underbrace{(1(0|1)^*1)} \mid \underbrace{0} \mid \underbrace{1}$.

$L(r)$ = all non-empty words that begin and end with the same letter.

3) $L = \{w \mid w \text{ contains exactly } 2k \text{ "1"s, for some } k \in \mathbb{N}\}$.

$r = 0^* (0^* 1 0^* 1 0^*)^* 0^*$

ensure that if there are no 1s, we still get all possible words.

$r = (0^* 1 0^* 1 0^*)^* \mid 0^*$ also works.

4) $L = \Sigma^*$

$r = (0|1)^*$ works.

$r = (0|1)^* \mid 1 \mid 00$ works.

** Non-example

Unfortunately, not every language is regular!

$L_1 = \{0^n 1^m \mid n, m \in \mathbb{N}\}$. is regular:

$r = 0^* 1^*$ has $L(r) = L_1$

$L_2 = \{0^n 1^n \mid n \in \mathbb{N}\}$

↑ there is no regex r such that $L(r) = L_2$!

(we'll eventually prove this ...)

** Bonus aside: Regex in programming

Typical alphabet Σ: all keyboard characters, as well as "escaped special characters" & more, eg. newline characters

Extended syntax, such as:

- `.` and `.*` ← any single char / any number of single characters
- `.*?` or `.*+` ← matches one or more characters
- `[^a]` ← matches any character except an a.
- `\d` ← matches a digit
- `\s` ⋮
- `\w`

short-cuts

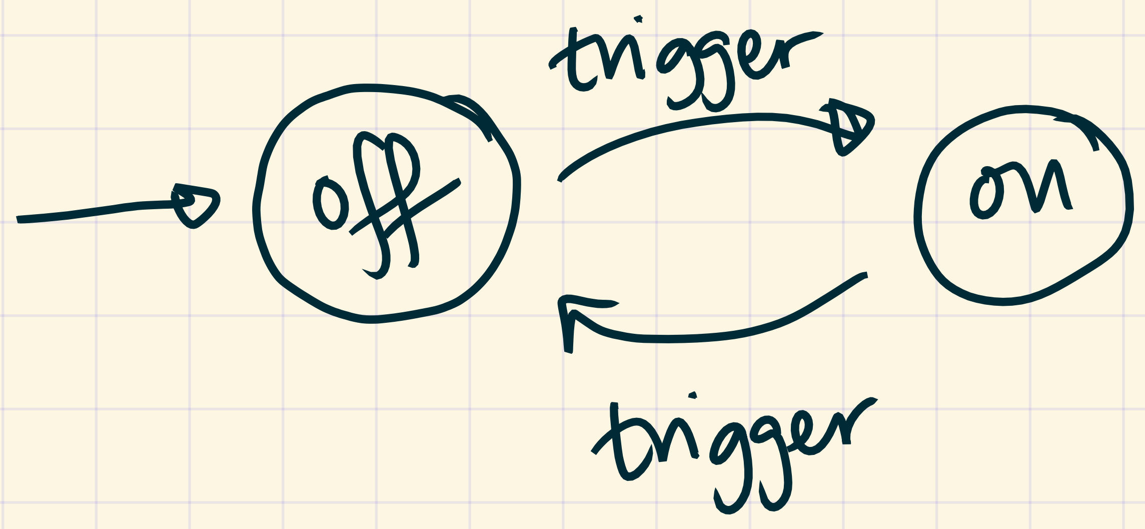
new

{ backreferences

* (Deterministic) finite automata.

Informally, a finite automaton is a decision tree flowchart, which represents a very simple "program".

** Informal example : Tap that turns on/off when you wave your hand in front of a sensor

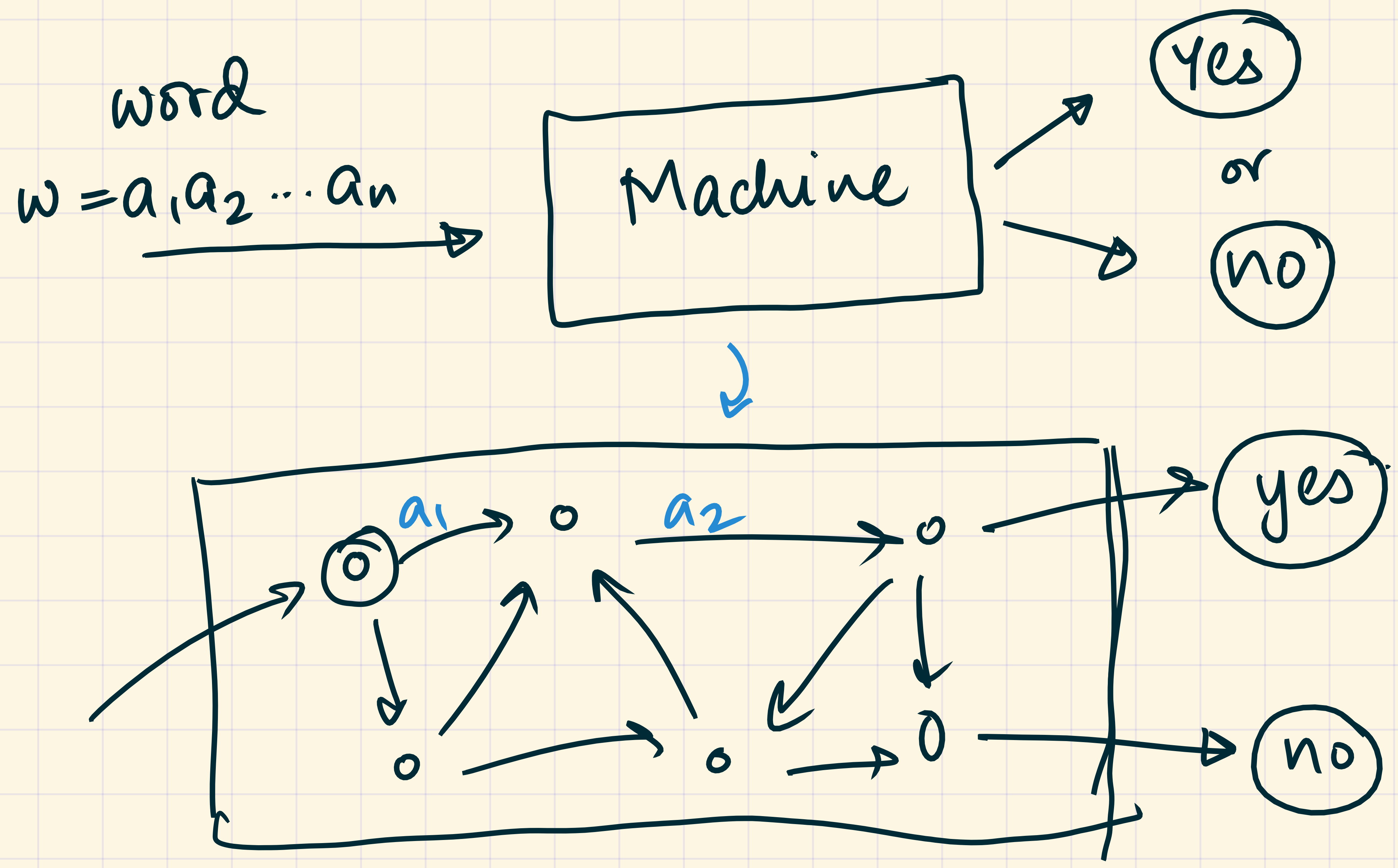


Let's try to formalise

We want some number of "internal states".

We will move between those states based on some external input.

In our case, we want each piece of the exrernal input to be a letter of some alphabet.

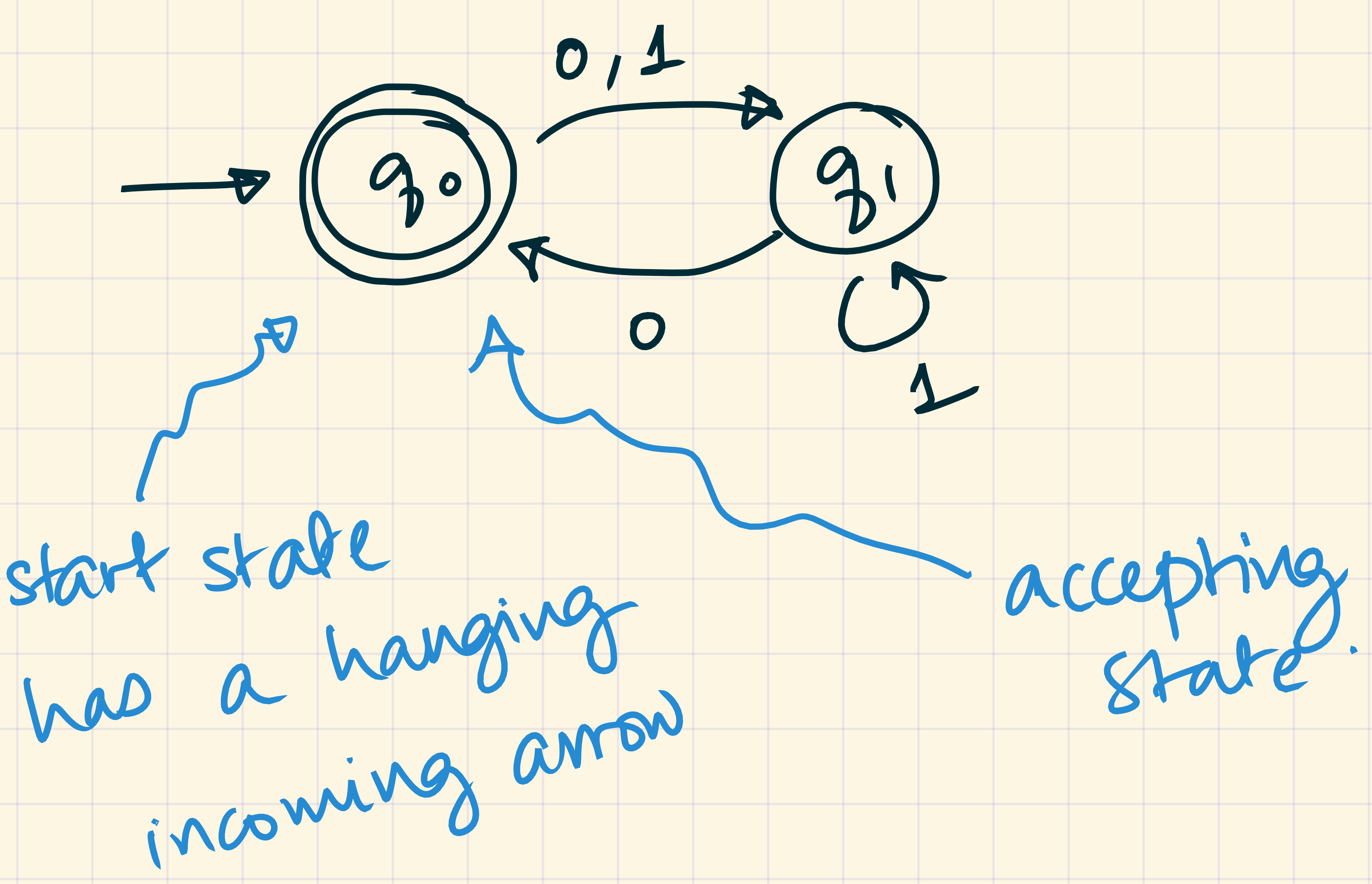


Fix an alphabet Σ.

** **Definition**: A deterministic finite automaton (DFA) consists of the following:

1) A directed graph, whose vertices are called "states"
2) A specified "starting state"
3) Each edge labelled by one or more letters of Σ
4) Some states are designated as "accept states" (circled) and others are reject states.
5) Every state has exactly one outgoing arrow labelled by each element of Σ.

*** **Example**



1) $w = 01$ ← rejected
   - Read letter-by letter.
   - Until the word ends.
   - "Accept" w if at the end, you land on an accepting state.
   - (Start at the start state.)

2) $w = 010$ ← accepted.