Lecture 10

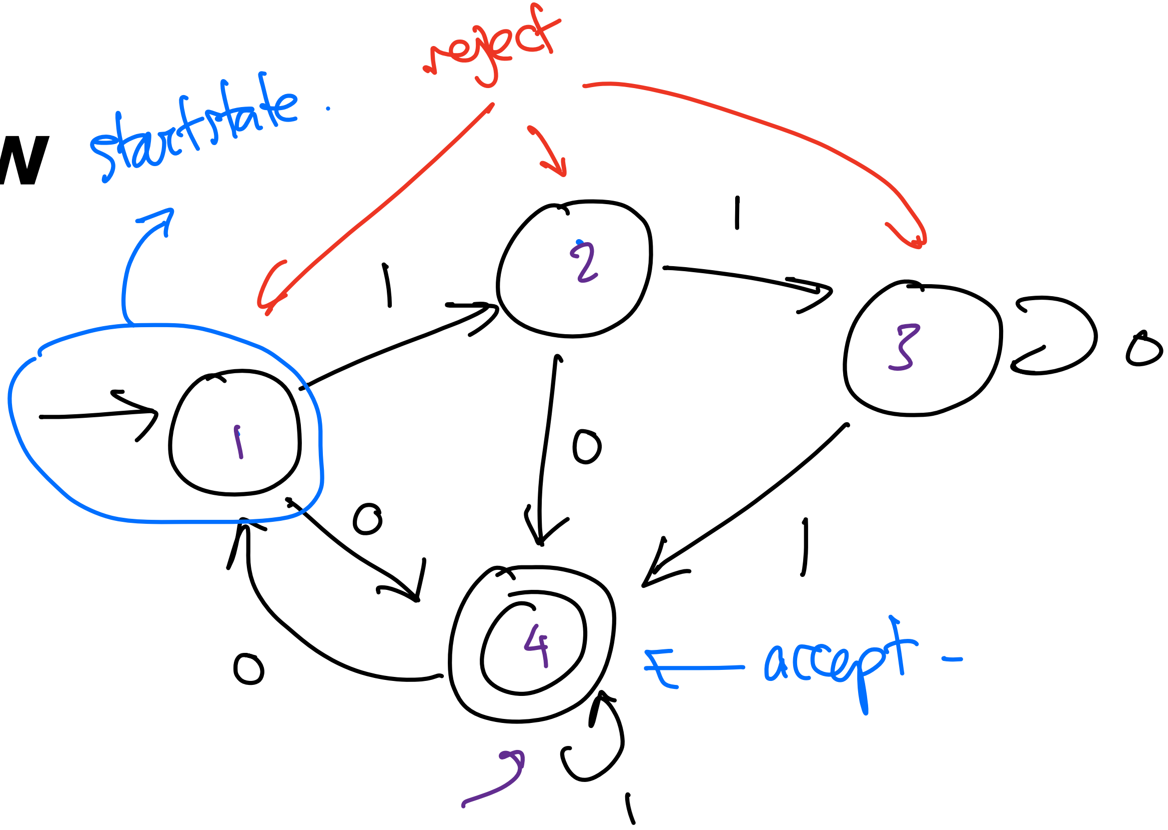
Regular Languages

2025-07-23

Review

DFAs

deterministic

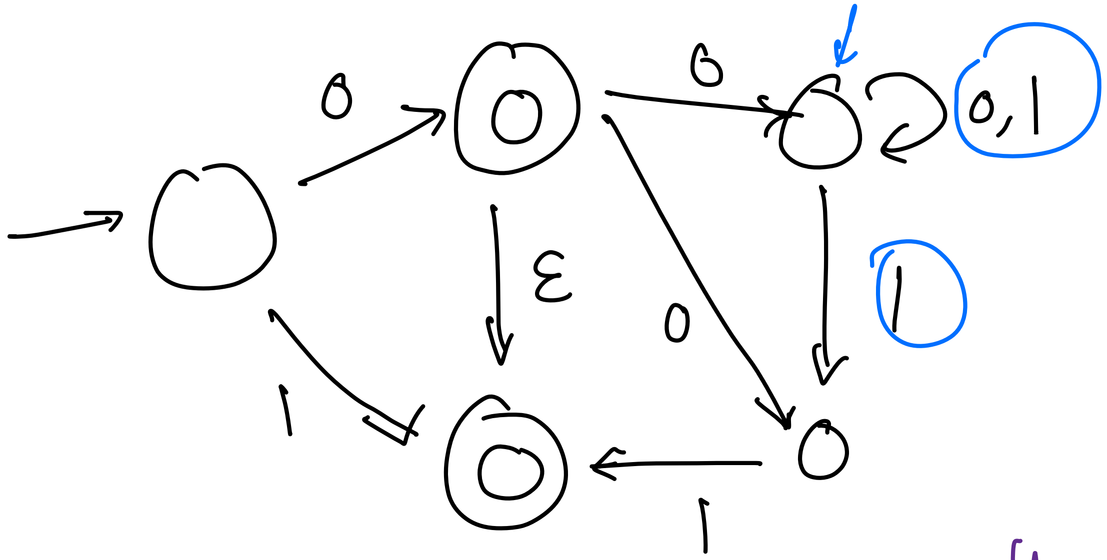


1010011

For every state & every symbol, there is one arrow coming out of that state labelled w/ the symbol

Review

NFAs



Computation: - similar to DFA, except, when there are multiple choices, just pick one,

- if any sequence of choices leads the NFA to accept, then that string is in the language of the NFA.

Regular Languages

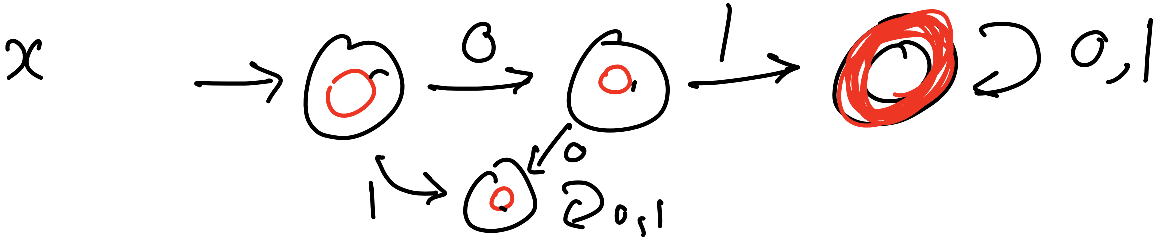
- A language $A \subseteq \Sigma^*$ is regular if and only if there exists some DFA M, such that $L(M) = A$

Complement

- **Claim.** If A is regular, then so is $\overline{A}$

WTS if $A = L(M)$, then $\overline{A} = L(M')$

$A = \{w : w \text{ starts w/ } 01\}$. $\overline{A} = \{w : \text{doesn't start w/ } 01\}$.



"swap acept and reject states."

Union

$\exists$ DFAs M_A, M_B, s.t. $L(M_A) = A$,
$L(M_B) = B$.

- **Claim.** If A, B are regular, then so is $A \cup B$

want. $\exists$ a DFA, M' s.t.
$L(M') = A \cup B$.

on input x.

Idea I: simulate M_A on x,
simulate M_B on x,
if either accepts, accept,
else: reject.

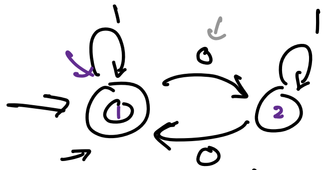
doesn't work b/c can only read the input once.

The fix: parallelization: run both M_A, M_B @ the same time.

$A = \{w : w$ has an even number of 0s $\}$.

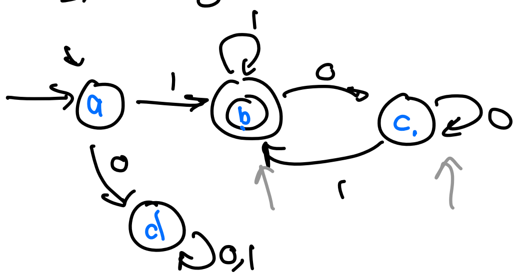
$B = \{w : w$ starts and ends with $1\}$.

$M_A =$

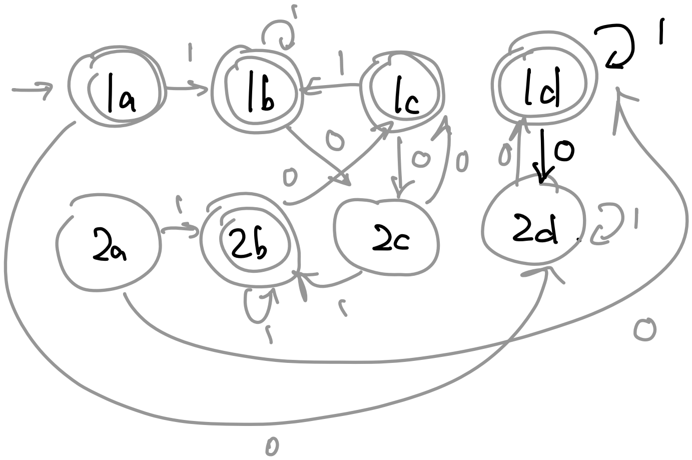


Goal: a DFA for $A \cup B$ = $\{w$: w has even # of 0, OR starts and ends w/ 1$\}$.

$M_B =$



M'



p

Closure

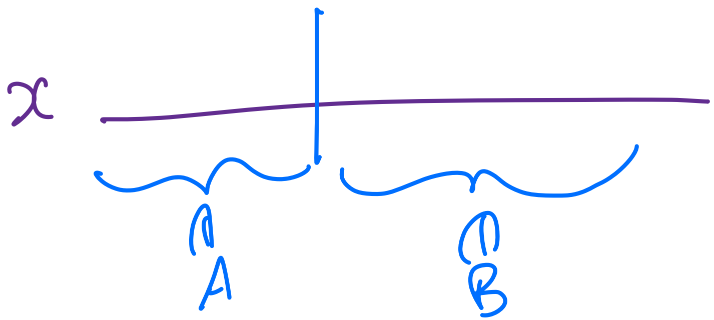
way to get more regular languages from regular languages.

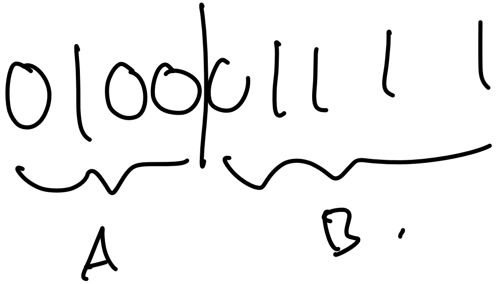
If A, B are regular, then so are…

- $\overline{A}$
- $A \cup B$
- $A \cap B$ HW
- AB = $\{ab : a \in A, b \in B\}$
- A^n
- A^*

Today.

Concatenation: AB





Equivalence of NFAs and DFAs

$\forall A \subseteq \Sigma^*$, $\quad \exists$ a DFA, M, s.t. $L(M) = A$

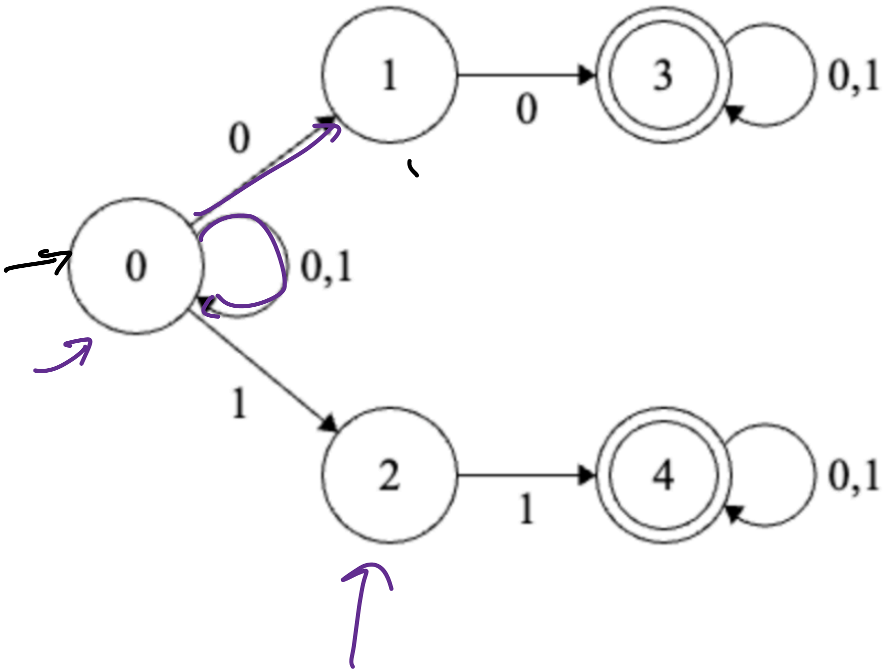
$\iff$

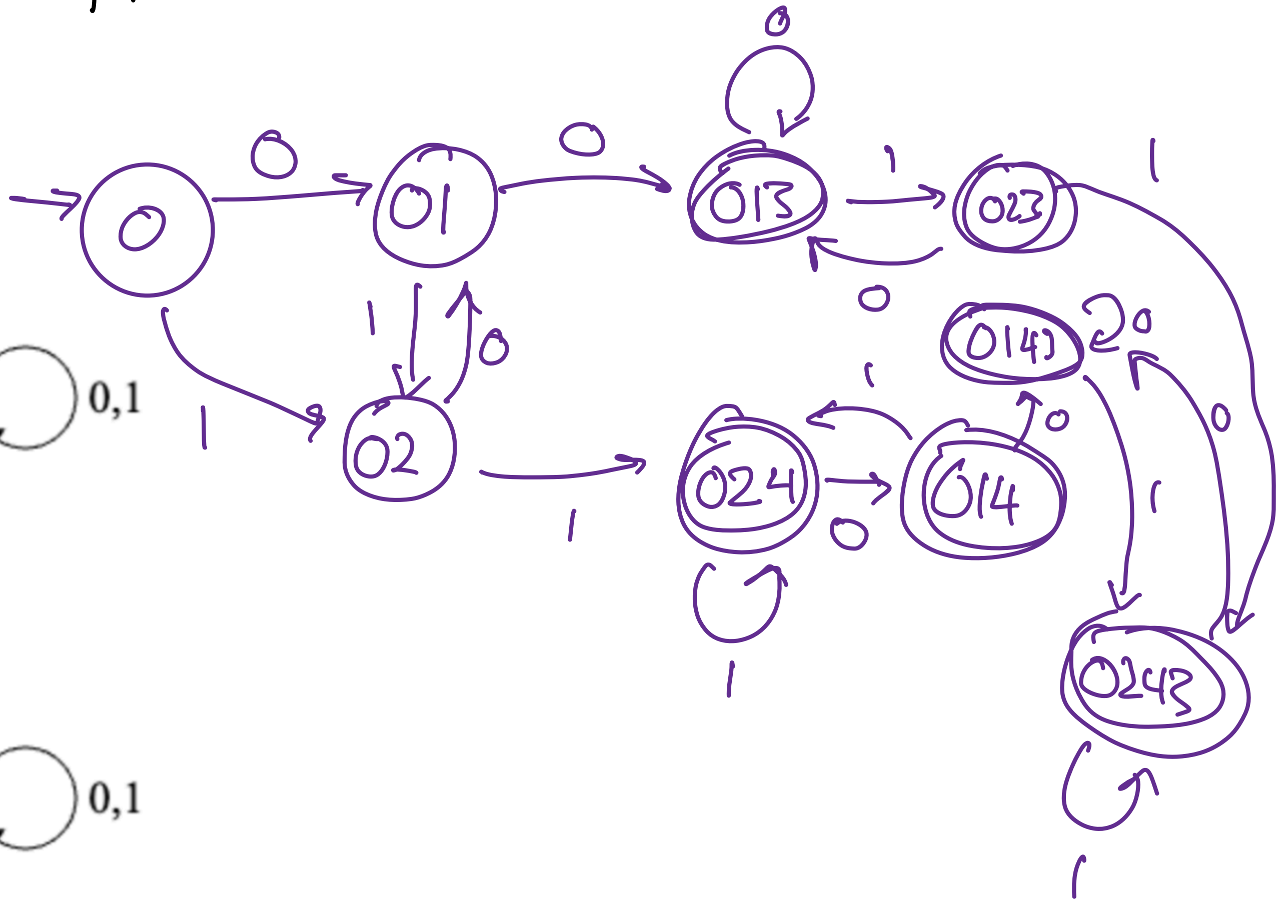
$\exists$ a NFA, N, s.t. $L(N) = A$.

$\Rightarrow$ is easy.

$\Leftarrow$ requires work.

Goal: simulate NFA w/ a DFA.



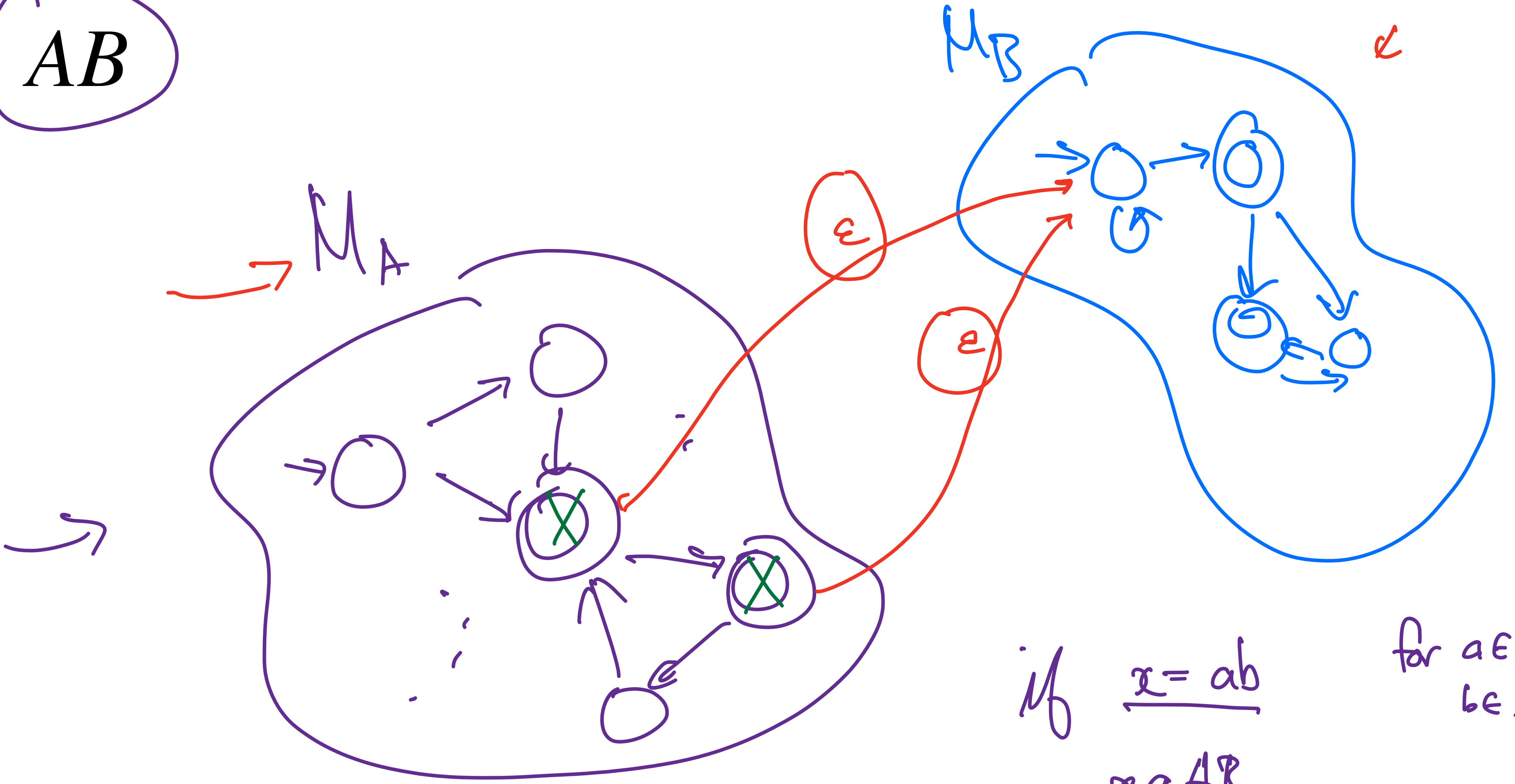


Regular Languages

$\hookrightarrow$ A is regular if $\exists$ a DFA, M s.t. $L(M) = A$,

$\iff \exists$ a NFA N, s.t. $L(N) = A$.

AB



if $\underline{x = ab}$

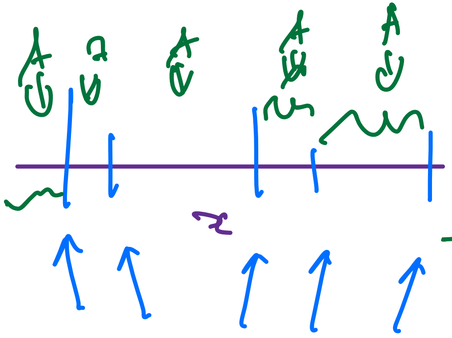
$x \in AB$.

for $a \in A$, $b \in B$.

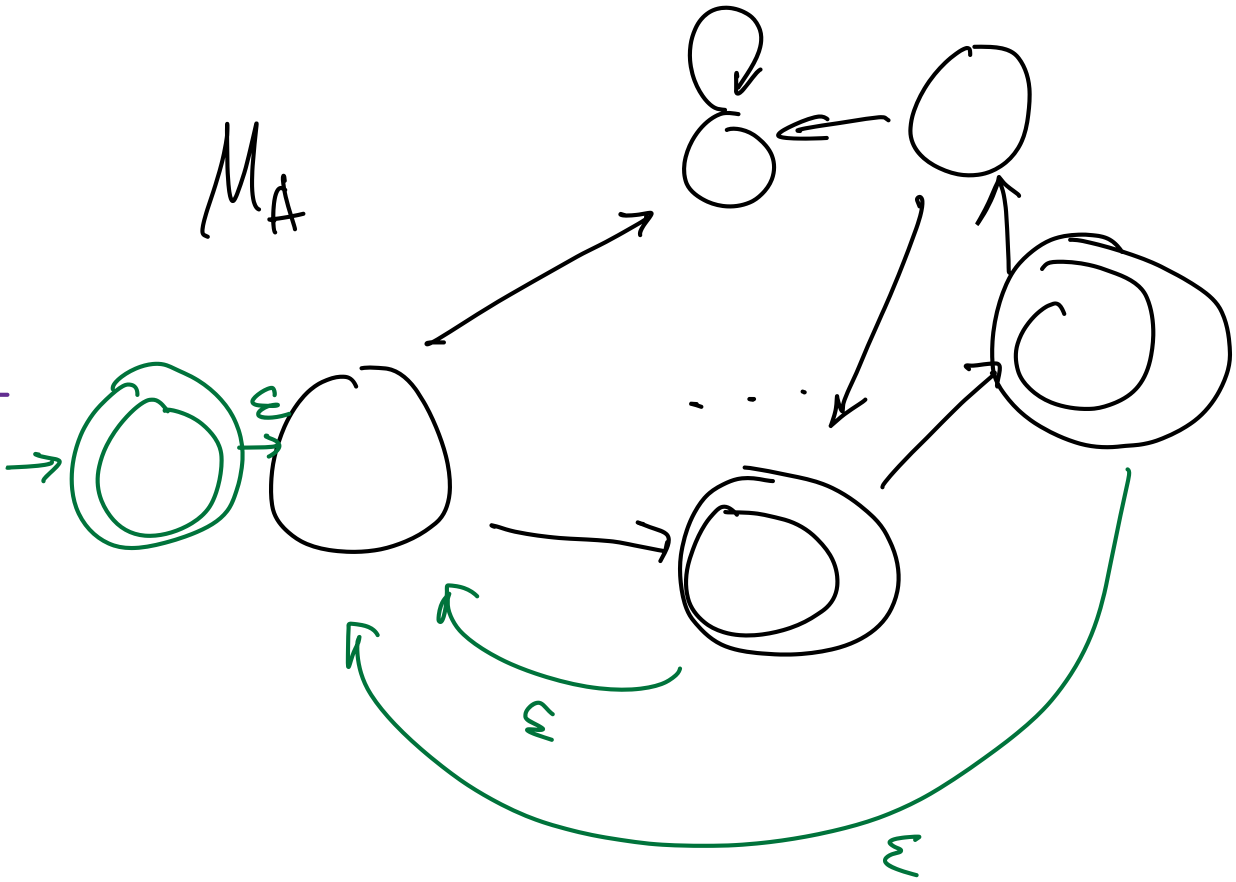
$$A^n = \{ a_1 a_2 \ldots a_n : a_1, \ldots, a_n \in A \}.$$

$\hookrightarrow$ By induction ✓ .

$$A^* = \bigcup_{i=0}^{\infty} A^i \qquad A^0 = \{\varepsilon\}.$$



M_A



Regular Expressions

- Fix an alphabet Σ. Then R_Σ, the set of regular expressions over the alphabet Σ is defined recursively/inductively as follows.

$\Sigma = \{0, 1\}.$

- Base cases: $B = \Sigma \cup \{\emptyset, \epsilon\}$

$B = \{0, 1, \emptyset, \varepsilon\}.$

- Functions:
  - $f_1(R) = (R^*)$
  - $f_2(R, S) = (RS)$
  - $f_3(R, S) = (R \mid S)$

$((((0^*)|1)\varepsilon)|1)^*$

- R_Σ is the set generated by $\{f_1, f_2, f_3\}$ from the base cases B.

Precedence

OR -

- * then concatenation then |

$$((((0^*)|1)\varepsilon)|)^* \Rightarrow ((0^*|1)\varepsilon|)^*$$

Language of a Regular Expression

Base cases:

$a \in \Sigma : L(a) = \{a\}.$

$L(\emptyset) = \{\ \}.$

$L(\varepsilon) = \{\varepsilon\}.$

---

if $R, S \in \mathcal{R}_\Sigma$, the $L(RS) = L(R) \cdot L(S)$

$L(R^*) = L(R)^*$

$L(R|S) = L(R) \cup L(S).$

Examples

$L(0|1) = L(0) \cup L(1)$.
$\{0\} \cup \{1\} = \{0,1\}$.

- $((0|1)(0|1)(0|1))^*$
- $(1^*01^*0)^*$
- $(0|1)^*1(0|1)(0|1)$
- $(0|1)^*010(0|1)^*$ $=$ any string containing 010.
- $(0|\epsilon)1^*$

$L((0|1)(0|1)(0|1)) = \{w : |w| = 3\}$.

$L(\quad) = \{w : |w| = 3k \text{ for some } k \in \mathbb{N}\}$.

Shorthand

$(0|1)(0|1)(0|1)$.

$(0|1)^3$

$\{0,1\}^3$.

If R is a regex, and $m \in \mathbb{N}$, then

- R^m means m copies of R.
- $R^+ = RR^*$, i.e. at least one copy of R.
- $R^{m+} = R^m R^*$ means at least m copies of R.

If $S = \{s_1, s_2, \ldots, s_n\}$ is a finite set of strings, then S is shorthand for $s_1|s_2|\ldots|s_n$. A common one is to use the alphabet, Σ.

More Examples

$\{0,1\}^*$ Σ^*.

Write regular expressions for the following languages

- Starts with 010. $010\{0,1\}^*$
- The second and the second last letter of w are the same. $((0|1)0(0|1)^*0(0|1) \mid (0|1)1(0|1)^*1(0|1))$
- Contains 110. $\{0,1\}^*110\{0,1\}^*$
- 1s always occur in pairs. $(0^*(11)^*0^*)^*$
- Doesn't contain more than four 1s in a row.

Equivalence of NFAs and Regex

Regex $\Longrightarrow$ NFA

Regular Languages

- The following are equivalent
  - A is regular
  - There exists a DFA M such that $L(M) = A$
  - There exists a NFA N such that $L(N) = A$
  - There exists a regular expression R such that $L(R) = A$

How to choose

- I typically use regular expressions for languages that seem to require some form of ‘matching’. For example *contains 121* as a substring, or *ends with 11*. Regular expressions are typically faster to find and write out in an exam setting.

- I’ll use NFAs when I can’t easily figure out a regular expression for something. These are usually languages for which memory seems to be useful like the $\mathrm{Dogwalk}$ example from hw.

- Stuff involving negations also seems easier to do with NFAs than with regular expressions. For example, *contains the substring* 011 is easy with regular expression, but *doesn’t contain the substring* 011 is a bit more complicated.