(Announcements: later)

Recap:

- Defined concatenation, power, star for languages, and closed closure.

See next page

$L_1 \circ L_2 = \{w \mid w = xy,\ x \in L_1,\ y \in L_2\}$

$L^k = \underbrace{L \circ L \circ \dots \circ L}_{k \text{ times}}$

$(L^0 = \{\varepsilon\}$ for all $L)$

$L^* = \bigcup_{k=0}^{\infty} L^k = \{w \mid \exists k \geq 0, \text{strings } x_1 \dots x_k, \text{ s.t. } w = x_1 \dots x_k, \text{ every } x_i \in L\}$

Q: Can L^* be a finite language?

A: yes! eg $L = \{\varepsilon\}$, $L^k = \{\varepsilon\}$ $\forall k \geq 0$, $L^* = \{\varepsilon\}$

↙ finite language (contains one string)

What about $L = \{\varepsilon, 1\}$? Can show $L^* = \{1^k \mid k \geq 0\}$

↙ infinite!

Is there any other L s.t. L^* is infinite? → exercise

→ <u>solution:</u> take $L = \{\}$, $L^* = \{\varepsilon\}$ - finite. No others!

Can prove L^* is infinite $\Leftrightarrow \exists w \in L$ s.t. $|w| \geq 1$

L^* is finite $\Leftrightarrow L = \{\}$ or $L = \{\varepsilon\}$

(Note that $\emptyset \neq \{\varepsilon\}$ but $\emptyset^* = \{\varepsilon\}^* = \{\varepsilon\}$)

We proved:

<u>Theorem:</u> The class of regular languages is closed under concatenation, power, star.

i.e: — L_1, L_2 regular $\Rightarrow$ $L_1 \circ L_2$ regular

— L regular $\Rightarrow \forall k \geq 0$ L^k is regular

— L regular $\Rightarrow$ L^* is regular.

At this point, we've seen that the class of regular languages is closed under:

union $\cup$, intersection $\cap$, Complement — } proved earlier

Concatenation $\circ$, power, Kleene star * } proved last time

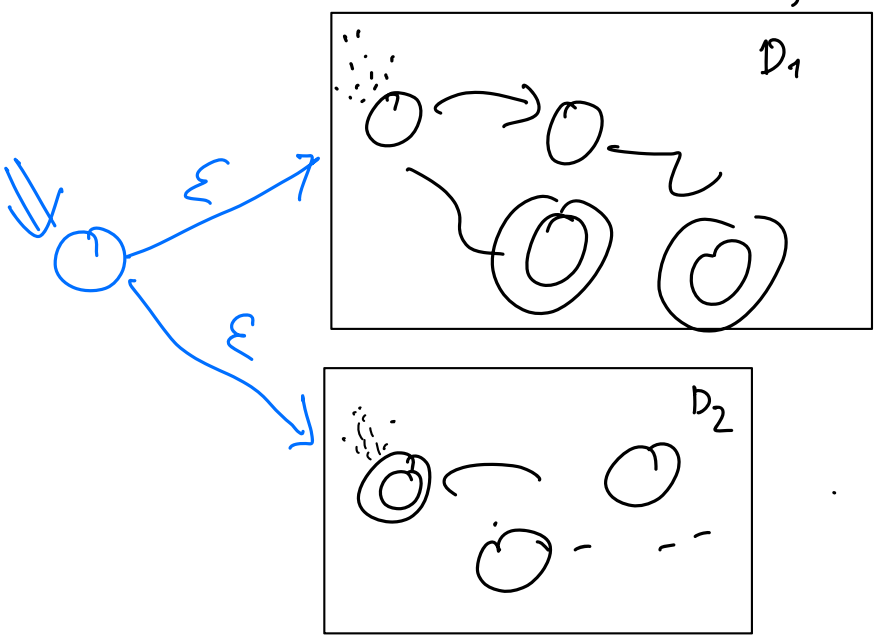
· Another proof for closure under union:

L_1, L_2 regular $\Rightarrow$ $L_1 \cup L_2$ regular

(we saw a proof by De-Morgan $L_1 \cup L_2 = \overline{\overline{L_1} \cap \overline{L_2}}$ and a construction of DFA for $L_1 \cup L_2$ given DFAs for L_1, L_2. Now we'll see a simpler construction of NFA for $L_1 \cup L_2$:)

Suppose we have DFAs D_1, D_2 s.t. $L(D_1) = L_1$
$L(D_2) = L_2$

Can construct an NFA for $\min$:
add a new start state with ϵ transition to each of the start states of D_1, D_2.



(Formal description: exercise)

w is accepted
$\Leftrightarrow$ $w \in L_1$ or $w \in L_2$
$\Leftrightarrow$ $w \in L_1 \cup L_2$

Def: The operations $\cup, \circ, *$ are called the regular operations.

We will see: L is regular $\iff$ L can be written as a finite combination of characters in Σ, $\emptyset$, and regular operations. (and also (,)) → "regular expressions"

Def: R is a regular expression over alphabet Σ if:

base cases:
- $R = \emptyset$
- $R = \varepsilon$
- $R = a$ for some $a \in \Sigma$

inductive cases:
- $R = (R_1 \cup R_2)$ for R_1, R_2 that are regular expressions
- $R = (R_1 \circ R_2)$ for R_1, R_2 that are regular expressions
- $R = (R)^*$ for R that's a regular expression

Just like we defined DFA, NFA syntax (5-tuple), and each such M recognized the language $L(M)$, here a regular expression R is a string of characters over alphabet $\{(,), \emptyset, \cup, *, \varepsilon\} \cup \Sigma$ and has a language associated with it $L(R)$.

(set of strings)

Def: For a regular expression R we define $L(R)$ the language associated with R (aka generated by R), as follows:

- $L(\emptyset) = \{\}$ $(= \emptyset)$
- $L(\varepsilon) = \{\varepsilon\}$
- $L(a) = \{a\}$ $\forall a \in \Sigma$
- $L((R_1 \cup R_2)) = L(R_1) \cup L(R_2)$
- $L((R_1 \circ R_2)) = L(R_1) \circ L(R_2)$ ← (original notes had a typo $\cup$ instead of $\circ$)
- $L((R)^*) = L(R)^*$

Note:
- parentheses may be omitted
- order of operations is $*$, $\circ$, $\cup$
- concatenation symbol $\circ$ is sometimes omitted

- For reg exp R_1, R_2 we say $R_1 \equiv R_2$ if $L(R_1) = L(R_2)$

(sometimes we just write $R_1 = R_2$)

Examples:

- $\emptyset = \{\}$ = the empty set (set with no strings)

$\emptyset$: a string "$\emptyset$"

the language associated with $\emptyset$ is $\{\}$

- $ua b^*)$ → not a valid regexp

- $R = ab \cup a b^* a$ (valid reg exp)

$\equiv ((a \circ b) \cup ((a \circ (b^*)) \circ a))$

$L(R) = \{ab, aa, aba, abba, abbba, \ldots\}$

$= \{w \in \{a,b\}^* \mid w = ab \text{ or } w = ab^i a \text{ for } i \geq 0\}$

(We've been denoting regexp in blue, but we'll drop this and just figure out from context if needed.)

what is the lang generated by:

- $R = (a \cup b)^*$ $\rightsquigarrow$ $L(R) = \{a,b\}^* = \{$all strings of a's and b's$\}$

  ($\{a\}$ $\{b\}$ $\{a,b\}^*$)

- $R = ((0 \cup 1)(0 \cup 1)(0 \cup 1))^*$ $\rightsquigarrow$ $L(R) = ?$

  $\{$strings of length 3$\}^*$ i.e. strings of length divisible by 3

i.e. $L(R) = \{w \in \{0,1\}^* \mid |w| \text{ is } 0 \bmod 3\}$

Note: this is not the same as $R' = (0 \cup 1)(0 \cup 1)(0 \cup 1)^*$

(what is $L(R')$? -exercise)

- Suppose R is some regexp generating lang. $L(R)$. What's the language generated by:

- $R \cdot \varepsilon$? ⇝

$$L(R \cdot \varepsilon) = L(R) \circ L(\varepsilon) = L(R) \circ \{\varepsilon\} =$$

$$= \{xy \mid x \in L(R), y \in \{\varepsilon\}\} = \{x \mid x \in L(R)\} = L(R)$$

- $R \cdot \emptyset$? ⇝

$$L(R \cdot \emptyset) = L(R) \circ \{\} =$$

~~$L(R)$?~~ ~~$\{\varepsilon\}$?~~

$$= \{xy \mid x \in L(R), y \in \{\}\} = \{\}$$

no such y!

- ε^*? ⇝ $L(\varepsilon^*) = (L(\varepsilon))^* = \{\varepsilon\}^* = \{\varepsilon\}$
- $\emptyset^*$? ⇝ $L(\emptyset^*) = (L(\emptyset))^* = \{\}^* = \{\varepsilon\}$
- Note: $R \cdot \emptyset^* \equiv R \cdot \varepsilon \equiv R$, but $R \cdot \emptyset \equiv \emptyset$

Theorem: A language L is generated by a reg exp $\iff$ L is regular

proof: Need to show both directions:

(1) ($\Rightarrow$): if $L = L(R)$ for some regexp R, then there's an NFA recognizing L.

(2) ($\Leftarrow$): If L is recognized by some NFA then $\exists$ reg exp R that generates it.

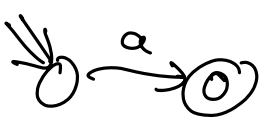
pf of (1): given reg exp R, $\exists$ a NFA for $L(R)$

proof by induction on # of operations in R

Base case:

- $R = \emptyset$: $L(R) = \{\}$ NFA: 

- $R = \varepsilon$: $L(\varepsilon) = \{\varepsilon\}$ NFA: 
- $R = a$ for $a \in \Sigma$: $L(a) = \{a\}$

NFA: 

Inductive step:

Assume true for all reg exp with $< k$ operations
prove for R with k operations:

- $R = (R_1 \cup R_2)$: R_1, R_2 each have $< k$ operations

(since together they have $k-1$:
$R_1 \cup R_2$
one
k total)

Thus, by inductive hypothesis ("IH")
$L(R_1)$, $L(R_2)$ regular.

By closure under union (previously proved)
$L(R_1) \cup L(R_2) = L(R)$ is regular

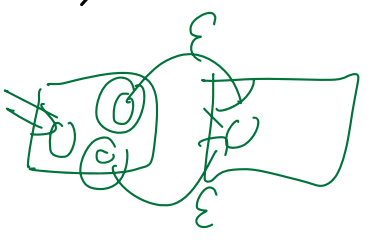
↳ given NFAs for $L(R_1), L(R_2)$
can make NFA for $L(R)$:



- $R = (R_1 \circ R_2)$: R_1, R_2 each have $< k$ operations

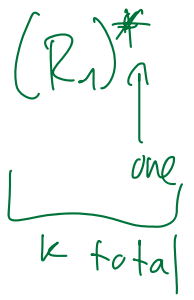
Thus, by IH, $L(R_1), L(R_2)$ regular.

By closure under concatenation,
$L(R_1) \circ L(R_2) = L(R)$ regular.



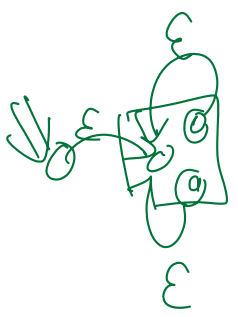
- $R = (R_1)^*$: R_1 has $k-1$ operations.

Thus, by IH, $L(R_1)$ regular.



By closure under Kleene star,

$(L(R_1))^* = L(R)$ regular



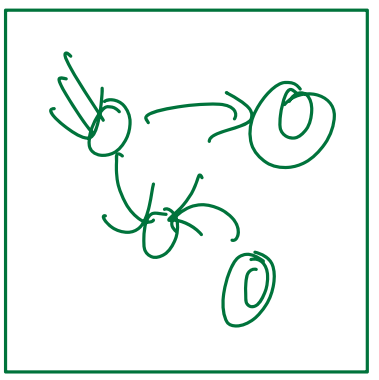
✓

proof of (2): We'll skip: see book if interested.

↘ (or see example at end of lecture)

Idea: To go from NFA to regexp:

start from NFA N:

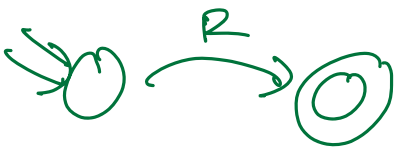


⇒ "generalized NFA" which has regexp labels on all transitions

⇒ remove one state at a time while maintaining same language

⇒

end up with



R is the desired regexp

$L(R) = L(N)$

☒

So we proved that regexp is another characterization of regular languages

L is regular $\iff$ $\exists$ DFA recognizing L

$\iff$ $\sim_L$ has finitely many equivalence classes

$\iff$ $\exists$ NFA recognizing L

$\iff$ $\exists$ regexp generating L

[Aside: the way we defined regexp is basically same (upto notation) as regexp that are used in various other contexts like grep. However for us $L(R)$ is the set of strings that match R - not just those that have a substring matching R]

Some more examples:

- whats the lang for this reg exp:

$R = (0 \cup 1)^* 00 (0 \cup 1)^*$

Answer: $L(R) = \{w \in \{0,1\}^* \mid w \text{ contains } 00\}$

- Whats the reg. exp for

$\{w \in \{0,1\}^* \mid w$ <u>does not</u> contain $00\}$?

Note: we know there is a regexp, since Reg. lang. closed under complement.
But constructing it can be tricky.

Answer:

$1^* (011^*)^* \cup 1^*(011^*)^*0$

- 1^*: any # of 1's
- $(011^*)^*$: any # of times
- 011^*: 0 followed by at least one 1
- $1^*(011^*)^*$: captures all strings in L that end in 1, as well as ε → Forgot to write this in class!
- $1^*(011^*)^*0$: captures all strings in L that end in 0

[that's all the examples we got to,
but see William's lecture notes for solutions
to the following exercises:

↓
try to solve for yourself first!

(1) give regexp for $\{w \mid |w| \text{ is not divisible by } 3\}$

(2) Same or not?

- $(aa)^* \stackrel{?}{\equiv} aa^*$
- $(0 \cup 1)^* \stackrel{?}{\equiv} (0^* \cup 1^*)^*$]

Announcements:

- This is the end of the technical material for Quiz 2, HW2, Handout 2

  ↙ next week

- Tue is quiz 1!
- HW1 solutions out shortly (hopefully also graded HW1 + feedback)
- fun challenge problem!

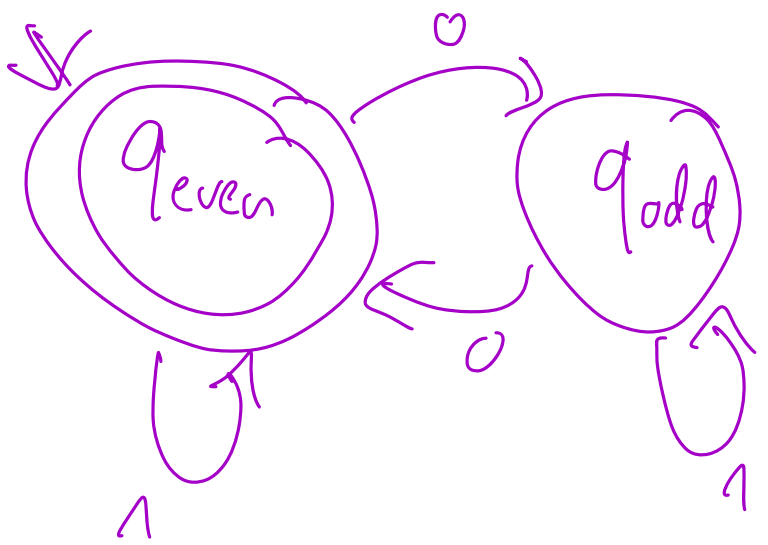
- optional example below

Example of converting NFA to reg exp: → added after class

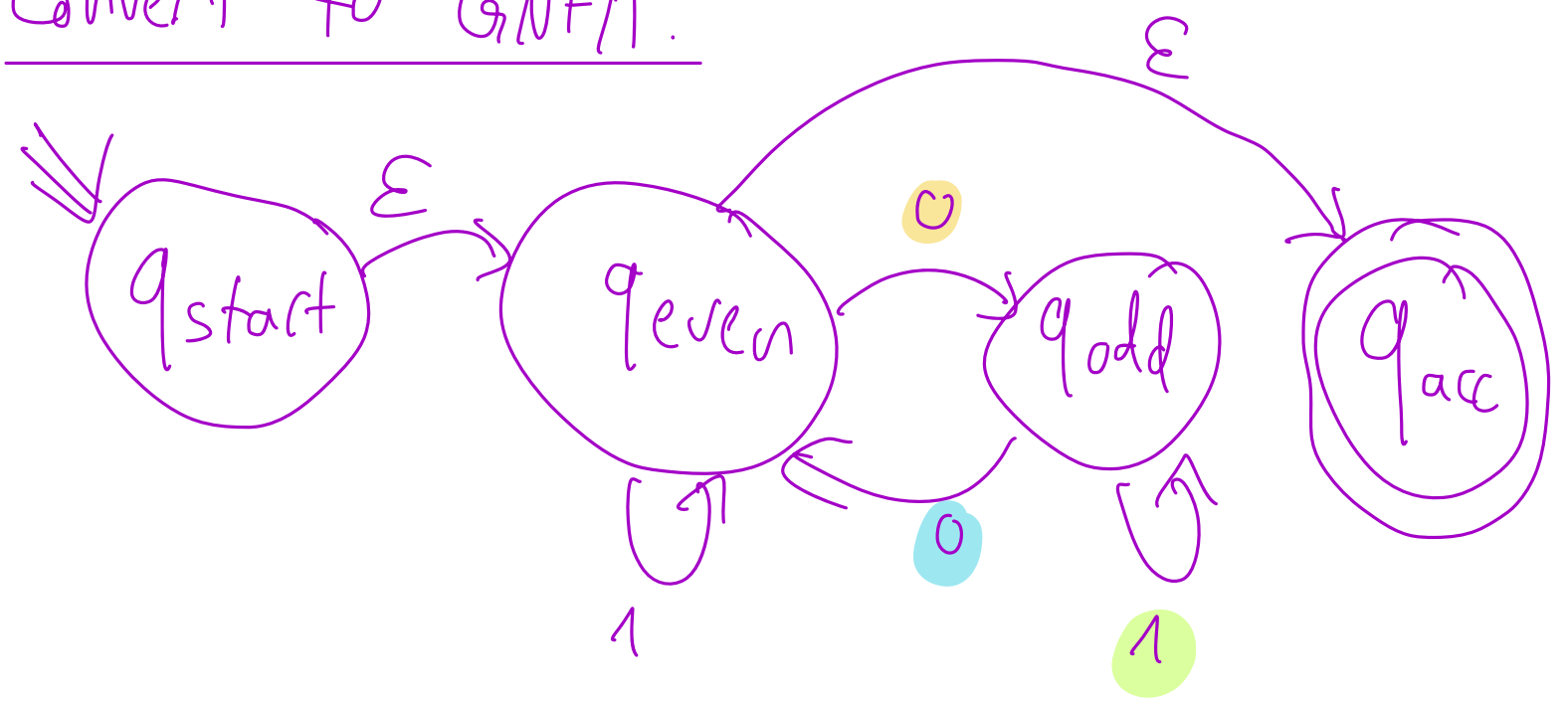
(was not covered in class and is not part of required material).

Let $L = \{w \in \{0,1\}^* \mid w \text{ has an even \# of 0's}\}$

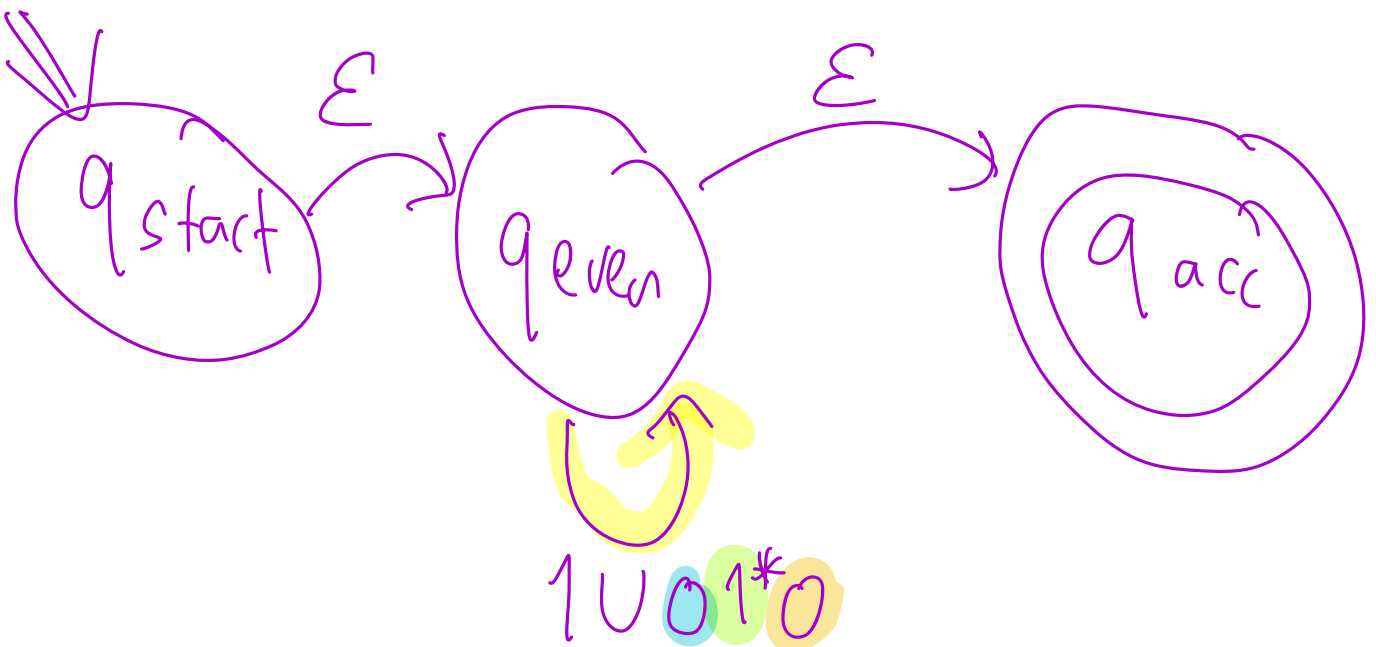
NFA for L:



Convert to GNFA:



Remove q_{odd}:



Remove q_{even}:



output $\boxed{R = (1 \cup 01^*0)^*}$

Indeed $L(R) = L = \{w \mid w \text{ has even \# of 0's}\}$