Lecture 4

Announcements:

- Office hours start this week! (Check calender)
- HW1 OUT (check webpage). Due in 2 weeks
- Review Sessions (optional) start soon (see calendar)
  handout from all review sessions will be posted on webpage
- Extra supplementary material at bottom of webpage

Closure Properties of Regular Languages

1. If $L \subseteq \Sigma^*$ is regular, then $\bar{L} = \{ w \in \Sigma^* \mid w \notin L \}$ is also regular

2. If L is regular, then $L^* = \{ w \mid w = v_1 \cdot v_2 \cdot \ldots \cdot v_k \mid v_1, \ldots, v_k \in L \}$ is regular

3. If L_1 and L_2 are regular, then $L_1 + L_2 = \{ w \mid w \in L_1 \text{ or } w \in L_2 \}$ is regular

4. If L_1 and L_2 are regular, then $L_1 \cdot L_2 = \{ w \mid w$ can be written as $u \cdot v$ where $u \in L_1$ and $v \in L_2 \}$

The closure properties are useful for showing that a language is regular.

<u>Example</u>

$$L = \{ w \mid \overbrace{w \text{ has an odd \# of 1's}}^{L_1} \text{ or } \underbrace{|w| = 7}_{L_2} \}$$

- so $L = L_1 + L_2$

- Can show L is regular in steps:

  First construct DFAs for L_1 and L_2 then since regular languages are closed under union (+), it follows that $L = L_1 + L_2$ is regular.

1. If L is regular then $\bar{L}$ is regular:

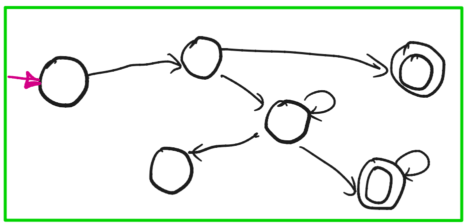
Proof sketch

Let M be a DFA for L, $M = (Q, q_0, \Sigma, F, \delta)$

Construct M' based on M $M' = (Q, q_0, \Sigma, F', \delta)$,

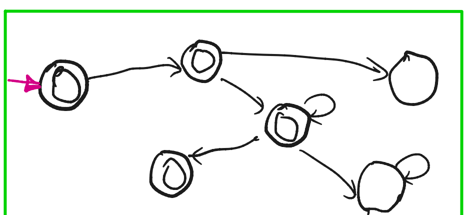
$F' = \{q \in Q \mid q \notin F\}$

M:



$\bar{L} = \{w \mid w \notin L\}$

M'



Proof of correctness sketch:

1. show $\forall w \in \Sigma^*$ $w \in L \rightarrow w \notin \bar{L}$

   if w accepted by $M \rightarrow w$ rejected by M'

2. $\forall w \in \Sigma^*$

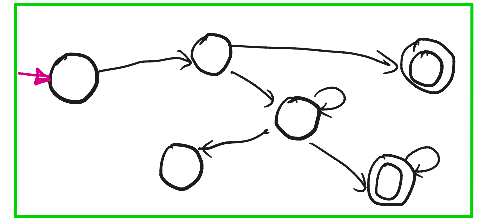
   if w is rejected by $M \rightarrow w$ accepted by M'

② If L is regular, then L^* is also regular.

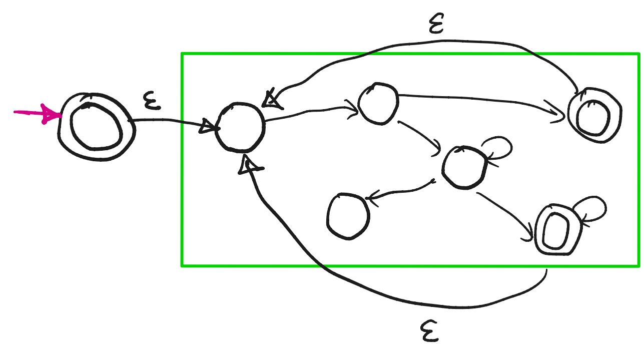
$L^* = \{w \mid w = w_1 \dots w_k, \text{ where each } w_i \in L, \text{ for some } k \geq 0\}$

Proof idea

Let M be a DFA for L:



NFA M' for L^*:



see textbook for proof of correctness of construction

Examples

$L = \{ w \mid w \text{ starts + ends in a } 1 \}$ over $\{0,1\}$

For example
$w_1 = 10111 \in L$ $\quad w_2 = 10001 \in L$

$w_1 w_2 \in L^*$, $w_1 w_1 \in L^*$, $w_1 w_2 w_1 \in L^*$

$L_1 = \{ w \mid w \text{ has length } 3 \}$ $\quad L_2 = \{ w \mid w \text{ has length } 7 \}$

$L_1 + L_2 = \{ w \mid w \in L_1 \text{ or } w \in L_2 \}$

For example: $1110000 \in L_1 + L_2$

$L_1 \cdot L_2 = \{ w \mid w = w_1 w_2 , w_1 \in L_1 \text{ and } w_2 \in L_2 \}$

$$L^* = \varepsilon \cup L \cup L \cdot L \cup L \cdot L \cdot L \cup \cdots$$

$$= \bigcup_{k \geq 0} \underbrace{L \cdot L \cdot \ldots \cdot L}_{k \text{ times}}$$

L^* is always infinite (unless $L = \phi$, or $L = \{\varepsilon\}$)

ex. $L = \{0\}$

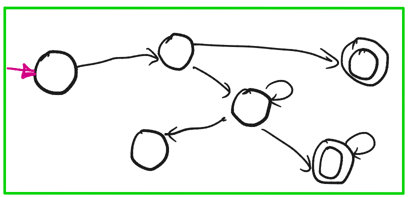
$$L^* = \{\varepsilon, 0, 00, 000, \ldots\}$$

③ If L_1, L_2 are regular, then $L_1 + L_2$ is regular

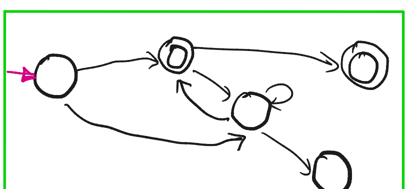
Proof sketch: Let $N_1 = (R = \{r_0, \dots, r_k\}, r_0, F_1, \Sigma, \delta_1)$ accept L_1

$N_2 = (S = \{s_0, \dots, s_\ell\}, s_0, F_2, \Sigma, \delta_2)$ accept L_2

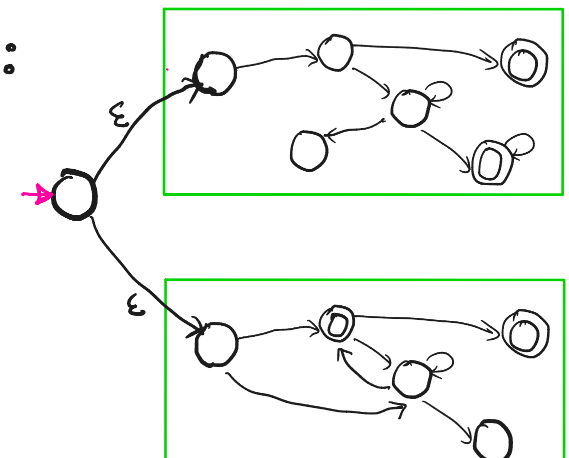
N_1:



N_2:



N':



Pf of correctness: show ① $\underbrace{\text{If } w \in L_1 \text{ or } w \in L_2}_{w \text{ accepted by either } N_1 \text{ or } N_2} \rightarrow w$ accepted by N

Pf of correctness

① Show: If $w \in L_1 + L_2$ then w accepted by N'

Since $w \in L_1 + L_2$ w is accepted by at least one of N_1 or N_2.

Case (i): w accepted by N_1

We need to show that in this case, w is also accepted by N'.

Since w accepted by N_1, there is a computation path of N_1 on w that ends in some accept state of N_1. So in N' take upper ε-transition and then follow accepting computation of N_1 on w.

Case (ii) w accepted by N_2 : same argument.

Pf of correctness continued.

(2) Show: if $w \notin L_1 + L_2$ then w is not accepted by N'.

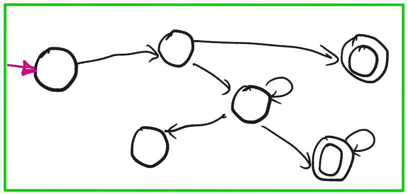
Since $w \notin L_1 + L_2$ this means all computation paths of N_1 on w lead to a non-accept state and similarly all computation paths of N_2 lead to a non accept state.

So when we run N' on w, whether we take the upper ε-transition or the lower ε-transition, in either case we can never reach an accepting state and therefore w is rejected by N'

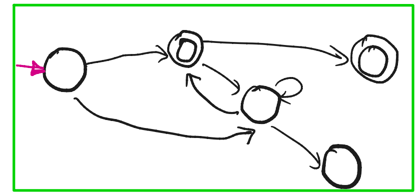
④ If L_1, L_2 are regular, then $L_1 \cdot L_2$ is regular. Let N_1, N_2 be NFAs for L_1 & L_2

Proof idea

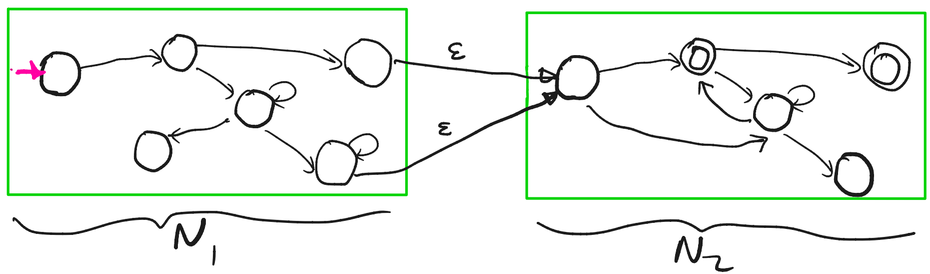
N_1:



N_2:



N':



Exercise: Work out the formal construction and proof of correctness for ① - ④

$w = w_1 w_2$

$L_1 = \{00, 000\}$

$L_2 = \{11\}$

$L_1 \circ L_2 = \{0011, 00011\}$

$L_1 \circ L_2 = \{w \mid w$ can be written as $w_1 w_2$ where $w_1 \in L_1$ and $w_2 \in L_2\}$

2 directions of proof of correctness

(that if L_1, L_2 are regular $\rightarrow L_1 \circ L_2$ regular

we constructed N' from NFAs for L_1 L_2

show $\forall w \in \Sigma^*$

$w \in L_1 \circ L_2 \rightarrow w$ accepted by N'

$w \notin L_1 \circ L_2 \rightarrow w$ not accepted by N'

Regular Expressions

Now we will give an equivalent characterization of regular languages.

A **regular expression** over Σ is another (different) way of describing a language.

DFAs : a "machine" characterization of a language

regular expression : an inductive definition of a language using operations $*, +, \cdot$

Some examples of regular expressions over $\Sigma = \{a, b, c\}$

1. ϕ $\qquad L = \phi$

2. ε $\qquad L = \{\varepsilon\}$

3. a $\qquad L = \{a\}$

4. $a + b$ $\qquad L = \{a, b\}$

5. $a \cdot b \cdot b \cdot c$ $\qquad L = \{abbc\}$

6. $(a+b)^* \cdot c$ $\qquad L = \{c, ac, bc, aac, abc, bac, bbc, aaac, aabc, \ldots, bbbc, \ldots\}$

$= \{w \in \Sigma^*$ that end in 'c' and contain no other c's $\}$

Formal Definition of a Regular Expression

Let Σ be a finite alphabet

R is a regular expression over Σ if :

$$\left.\begin{array}{ll} (1) & R = a \quad \text{for some } a \in \Sigma \\ (2) & R = \varepsilon \\ (3) & R = \phi \end{array}\right\} \text{base cases}$$

$$\left.\begin{array}{ll} (4) & R = R_1 + R_2 \quad \text{where } R_1, R_2 \text{ are regular expressions over } \Sigma \\ (5) & R = R_1 \cdot R_2 \quad \text{where } R_1, R_2 \text{ are regular expressions over } \Sigma \\ (6) & R = (R_1)^* \quad \text{where } R_1 \text{ is a regular expression over } \Sigma \end{array}\right\} \text{inductive cases}$$

* Note : in book $+$ is $\cup$ (union)

$\cdot$ is $\circ$ (concatenation)

More Examples

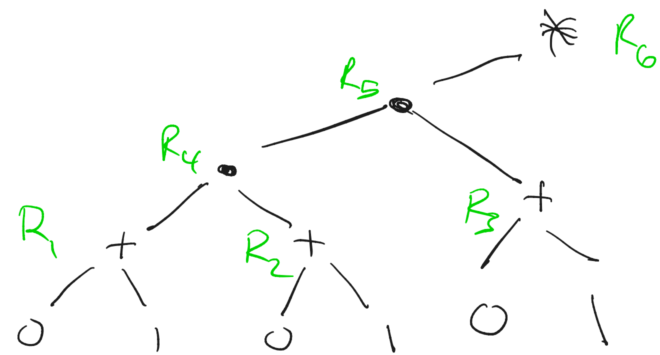
1. $((0+1)\cdot(0+1)\cdot(0+1))^*$

2. $(0+1)^*\cdot 1\cdot 1\cdot 1\cdot (0+1)^*$

3. $1^*\cdot 0\cdot 1^*$

$\{w\in\{0,1\}^* \mid \text{length of } w \text{ is divisible by } 3\}$

$$\overbrace{(\overbrace{\underbrace{(\underbrace{(0+1)}_{R_1}\cdot\underbrace{(0+1)}_{R_2})}_{R_4}\underbrace{(0+1)}_{R_3}}^{R_5})^*}^{R_6}$$



Theorem Let Σ be a finite alphabet.
The class of languages over Σ that are regular is equal to the class of languages that are described by regular expressions

Proof has 2 directions:

(i) L has a regular expression $\longrightarrow$ L has an NFA
(and therefore L has a DFA so L is regular)

(ii) L has a DFA (or NFA) $\longrightarrow$ L has a regular expression

An example of a language that doesnt have an associated regular expression:

$$L = \{ 0^n 1^n \mid n \geq 0 \}$$

$$= \{ \varepsilon, 01, 0011, 000111, \ldots \}$$

Theorem Let Σ be a finite alphabet.
The class of languages over Σ that are regular is equal to the class of languages that are described by regular expressions

Proof has 2 directions:

(i) L has a regular expression $\longrightarrow$ L has an NFA
(proof uses closure properties!)

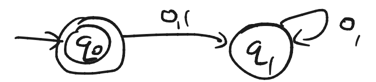
(ii) L has a DFA (or NFA) $\longrightarrow$ L has a regular expression
(harder)

(i) L has a regular expression $\rightarrow$ L has an NFA

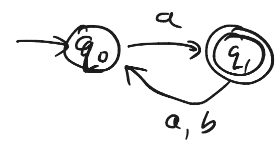
Proof by induction on length of regular expression for L.

Base cases: $L = \phi$

$L = \varepsilon$



$L = \{a\}$, $a \in \Sigma$,



(i) L has a regular expression $\rightarrow$ L has an NFA

Proof by induction on length of regular expression for L.

Inductive step:

IND hyp: For any L described by a regular expression involving at most K operations $*, +, \bullet$, L has an NFA

show: any L described by reg expression with K operations has an NFA

(i) L has a regular expression $\rightarrow$ L has an NFA

Inductive step:

IND hyp: For any L described by a regular expression involving at most K operations $*, +, \cdot$, L has an NFA

show: any L described by reg expression with K operations has an NFA

3 cases: (i) $L = (L_1)^*$

(ii) $L = L_1 + L_2$

(iii) $L = L_1 \cdot L_2$

where L_1, L_2 described by regular expressions using $\leq K$ operations

(i) follows by closure property 2
(ii) " " " " 3
(iii) " " " " 4

} see first slide from this lecture