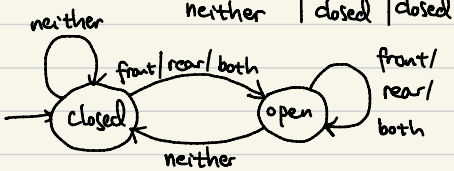
Why? ① useful

② 与 Turing Machine 比较一步

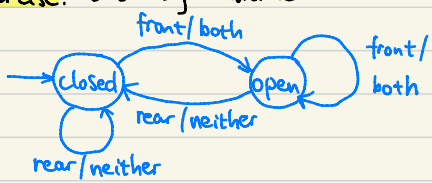
③ computability, 线性语言可判定 / 不可判定性

2. Finite Automaton

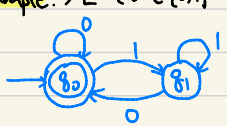
	closed	open
front only	open	open
rear only	open	open
both	open	open
neither	closed	closed



Exercise. one-way entrance



Example. 1) $L = \{w \in \{0,1\}^* : w = w_1 w_2 \cdots w_n,\ w_n = 0\}$



⓪：接收状态

2) (divisibility by 3) $L = \{w \in \{0,1\}^* : w = w_1 w_2 \cdots w_n,\ \text{s.t.} \sum_{i=1}^{n} 2^{n-i} w_i \equiv 0 \pmod 3\}$

Binary string, most significant bit first

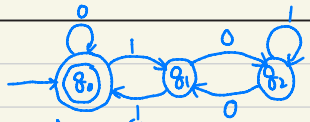
1 0 0 1

1 ①

1 0 ②

1 0 0 ①

1 0 0 1 ⓪



Suppose $(a_1a_2\cdots a_i)_2 \equiv r \pmod 3$. After reading a_{i+1}, we have $(a_1a_2\cdots a_{i+1}) \equiv 2r + a_{i+1} \pmod 3$.

Exercise

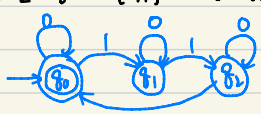
0) $L = \{w \in \{0,1\}^* : \text{number of 1's is a multiple of 3}\}$

1) $L = \{w \in \{0,1\}^* : w \text{ starts with } 01\}$

2) $L = \{w \in \{0,1\}^* : w \text{ ends with } 00\}$

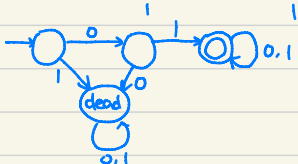
3) $L = \{w \in \{0,1\}^* : w = w^R\}$ w^R denotes the reversal of string w.

0)

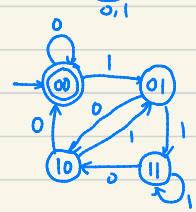


Suppose the number of 1's in $(a_1a_2\cdots a_i)_2$ is r. After reading a_{i+1}, the number of 1's in $(a_1a_2\cdots a_{i+1}) \equiv r+1 \pmod 3$

1)



2)



3) 无解!

Def 2.3 (finite automaton) A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1) Q is a finite set called states.
2) Σ is a finite set called alphabet
3) $\delta: Q \times \Sigma \to Q$ is the transition function
4) $q_0 \in Q$ is the start state.
5) $F \subseteq Q$ is the set of accept states.

Def 2.4 Let $M=(Q, \Sigma, \delta, q_0, F)$ be a FA. Let $w = w_1 w_2 \cdots w_n$ be a string, where $w_i \in \Sigma$. Say <u>M accepts w</u> if there exists a sequence of states $r_0, r_1, \cdots, r_n \in Q$ such that

1) $r_0 = q_0$
2) $\delta(w_i, w_{i+1}) = r_{i+1}$, for $i = 0, 1, \cdots, n-1$
3) $r_n \in F$.

Def 2.5 If L is the set of all strings that an FA M accepts, we say L is the <u>language accepted by M</u>, and we write $L(M) = L$
We say <u>M accepts L</u> or <u>M recognizes L</u>.

Def 2.6 (regular language) A language L is a regular language if some FA accepts it.

Let $A, B \subseteq \Sigma^*$

Union $A \cup B = \{x : x \in A \text{ or } x \in B\}$

Intersection $A \cap B = \{x : x \in A \text{ and } x \in B\}$

Complement $\bar{A} = \{x \in \Sigma^* : x \notin A\}$

A composed with B, A circle B

Concatenation $A \circ B$ or $AB = \{xy : x \in A \text{ and } y \in B\}$

Star $A^* = \{x_1 x_2 \cdots x_k : k \geq 0, (\forall i)(x_i \in A)\}$.

$A = \{0^i : i \geq 0\}$. $B = \{1^j : j \geq 0\}$ 几个0连在一起, 接1...

$A \circ B = \{0^i 1^j : i, j \geq 0\}$

$A^* = A$ $B^* = B$.

$(A \circ B)^* = \{0,1\}^*$

Next, we show that regular languages are closed under these operations.

Theorem 2.7 If L is a regular language, then $\bar{L}$ is also a regular language.

Proof. Since L is regular, there is an FA $M = (Q, \Sigma, \delta, q_0, F)$ that accepts L.

把 F 不接受的

Let $M' = (Q, \Sigma, \delta, q_0, F')$, where $F' = Q \setminus F$

It is clear that $L(M') = \overline{L(M)} = \bar{L}$. □

So, $\bar{L}$ is also regular

Theorem 2.8 If L_1 and L_2 are regular, then $L_1 \cup L_2$ is also regular.

Cartesian Product

Proof. The idea is to simulate M_1 and M_2 at the same time.

Let M_1 recognize L_1, where $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$

Let M_2 recognize L_2, where $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

Construct $M = (Q, \Sigma, \delta, q_0, F)$ to recognize $L_1 \cup L_2$, where

1) $Q = Q_1 \times Q_2 = \{(r_1, r_2) : r_1 \in Q_1, r_2 \in Q_2\}$

9.24

2) Σ is the same

3) $\delta : Q \times \Sigma \to Q$ is defined as $\delta((r_1, r_2), s) = (\delta_1(r_1, s), \delta_2(r_2, s))$

4) $q_0 = (q_1, q_2)$

5) $F = \{(r_1, r_2) : r_1 \in F_1$ and/or $r_2 \in F_2\}$

□

Theorem 2.9 If L_1 and L_2 are regular languages, then $L_1 \cap L_2$ is also regular.

Proof. Similar to Thm 2.8 except that $F = \{(r_1, r_2) : r_1 \in F_1$ and $r_2 \in F_2\}$

□

Next: $L_1 L_2$ is also regular

$x = x_1 ; x_2$ for loop ✓ but finite automaton ✗

非确定性有限自动机

NFA 造不出来，假想模型

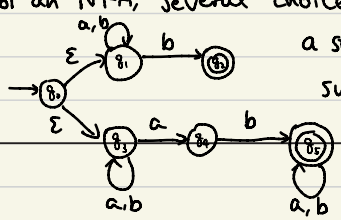
"直觉都对的人"

同时探索很多条路，有一个接收就接收.

2.2 Nondeterministic finite automaton

For a DFA, the next state is determined,

for an NFA, several choices may exist.



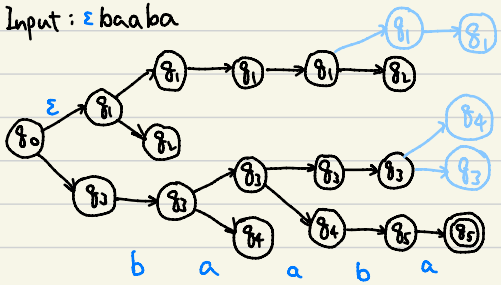
a string ends with b or contains a substring ab.

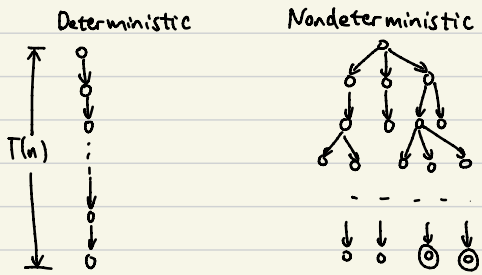
空跳转 ε：可跳非可不跳，可能任意多次.

实际所有路

但不可能实现，因为无法同时处于多个状态

Input : ε baaba





Exercise. 0) Construct an NFA to recognize

$L = \{w \in \{0,1\}^* : w = w_1 \cdots w_n, n \geq 3, w_{n-2} = 0\}$.

1) $L = \{0^k : k$ is a multiple of 2 or a multiple of 3$\}$

ε transition is allowed

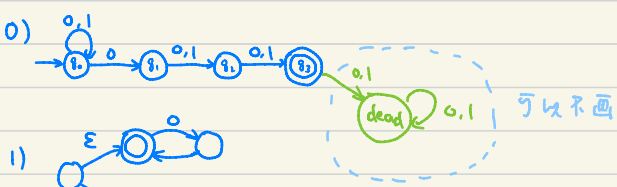
$\delta(q_0, 0) = \{q_0, q_1\}$

$\delta(q_0, 1) = \{q_0\}$

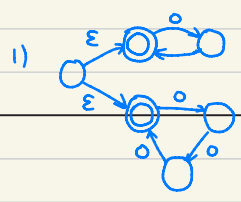
$\delta(q_3, 0) = \phi$

$\delta(q_3, 1) = \phi$

0)



1)



For any set Q, write $P(Q)$ to be the collection of all subsets of Q, called the power set.

$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

DFA $\delta : Q \times \Sigma \to Q$

NFA $\delta : Q \times \Sigma_\varepsilon \to P(Q)$

Def 2.11 A nondeterministic finite automaton (NFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set of states
2. Σ is the alphabet
3. $\delta : Q \times \Sigma_\varepsilon \to P(Q)$ is the transition function.
4. $q_0 \in Q$ is the start state
5. $F \subseteq Q$ is the set of accept state.

Def 2.12 Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA. Let $w \in \Sigma^*$, Say **N accepts w** if we can write $w = y_1 y_2 \cdots y_m$, where $y_i \in \Sigma_\varepsilon$, and there exists $r_0, r_1, \cdots, r_m \in Q$ such that

1. $r_0 = q_0$
2. $r_{i+1} \in \delta(r_i, y_{i+1})$ for $i = 0, \cdots, m-1$ and
3. $r_m \in F$.

Theorem 2.13 Every NFA has an equivalent DFA.

Proof. Let $N = (Q, \Sigma, \delta, q_0, F)$ be the NFA recognizing L.

We construct a DFA rec. L.

To deal with ε arrows, we define

闭包 $E(R) = \{q \in Q : q$ can be reached from R by travelling along zero or more ε arrows$\}$.

We define DFA $M = (Q', \Sigma, \delta', q_0', F')$ as follows:

1. $Q' = \mathcal{P}(Q)$
2. For each $R \in Q'$ and $s \in \Sigma$, let $\delta'(R,s) = \bigcup_{r \in R} E(\delta(r,s))$
3. $q_0' = E(\{q_0\})$
4. $F' = \{R \in Q' : R \cap F \neq \phi\}$. □

$L(N) = L$

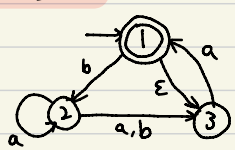
① $L \subseteq L(N)$

② $L(N) \subseteq L$

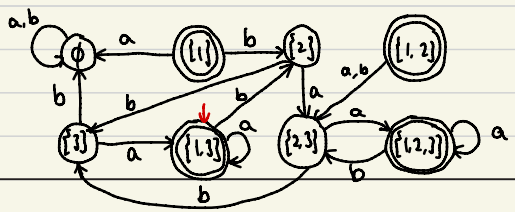
Definition
Proposition
Lemma
Theorem
Corollary

Cor 2.10 A language is regular if and only if some NFA recognizes it.

NFA



DFA



$\{1,2\} \xrightarrow{a,b} \{2,3\}$, 是因为 $\{1\} \xrightarrow{a} \phi$ $\{2\} \xrightarrow{a} \{2,3\}$ $\phi \cup \{2,3\} = \{2,3\}$

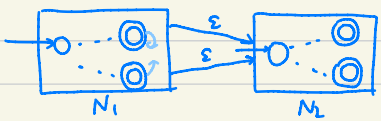
$\{1\} \xrightarrow{b} \{2\}$ $\{2\} \xrightarrow{b} \{3\}$ $\{2\} \cup \{3\} = \{2,3\}$.

用NFA instead of DFA: ε

Theorem 2.14 Regular languages are closed under concatenation.

Proof. Let NFA $N_1=(Q_1, \Sigma, \delta_1, q_1, F_1)$ rec. L_1

Let NFA $N_2=(Q_2, \Sigma, \delta_2, q_2, F_2)$ rec. L_2



Construct $N=(Q, \Sigma, \delta, q_0, F)$ to recognize L_1L_2 as follows:

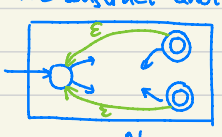
1. $Q=Q_1 \cup Q_2$
2. $q_0=q_1$
3. $F=F_2$
4. For any $q \in Q$ and $a \in \Sigma_\varepsilon$, define

$$\delta(q,a)=\begin{cases}\delta_1(q,a), & \text{if } q \in Q_1 \setminus F_1 \\ \delta_1(q,a), & \text{if } q \in F_1 \text{ and } a \neq \varepsilon \quad \text{// 在 } L_1 \text{ 内部转移} \\ \delta_1(q,a) \cup \{q_2\}, & \text{if } q \in F_1 \text{ and } a=\varepsilon, \\ \delta_2(q,a), & \text{if } q \in Q_2\end{cases}$$

Theorem 2.15 Regular languages are closed under the star operation.

Proof. Let NFA $N=(Q, \Sigma, \delta, q_0, F)$ recognize L.

We construct another NFA that recognizes L^*.



Construct $N'=(Q', \Sigma, \delta', q_0, F')$ as follows:

1. $Q'=Q$
2. q_0 is the start state
3. $F' = F \cup \{q_0\}$ L^* 允許取0个 string 連起來
4. For any $q \in Q$ and $a \in \Sigma_\varepsilon$, define

$$\delta'(q,a) = \begin{cases} \delta(q,a) & \text{if } q \in Q \setminus F \\ \delta(q,a) & \text{if } q \in F \text{ and } a \neq \varepsilon \\ \delta(q,a) \cup \{q_0\} & \text{if } q \in F \text{ and } a = \varepsilon \end{cases}$$

□

2.3 Regular expressions

Examples

1) $(0 \cup 1)0^*$ First symbol is 0 or 1, appended by zero or more 0s.
2) $(0 \cup 1)^* = \{0,1\}^*$
3) $(0\Sigma^*) \cup (\Sigma^* 1)$ first symbol is 0 or last symbol is 1.

Recursive def.

Def 2.16 (Regular expression) R is a regular expression if R is

1. a symbol in Σ Parentheses may be omitted
2. ε
3. $\emptyset$
4. $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions
5. $(R_1 R_2)$ ----
6. (R_1^*), where R_1 is a regular expression
7. $(R_1)^+$

至少一次 $R_1 R_1 R_1 \cdots$

Examples.

1) $0^*10^* = \{w \in \{0,1\}^* : w \text{ contains exactly a single } 1\}$

2) $\Sigma^*1\Sigma^* = \{$ . . - - - - - at least a 1 $\}$

3) $\Sigma^*001\Sigma^* = \{w : w \text{ contains a substring } 001\}$.

4) $1^*(01^+)^* = \{w : \text{every } 0 \text{ is followed by at least one } 1\}$

5) $(\Sigma\Sigma)^* = \{w : w \text{ has even length}\}$

6) $01 \cup 10 = \{01, 10\}$

7) $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$

8) $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$

$\phi^+ = \phi$ 9) $1^*\phi = \phi$ // 1^*中的string和ϕ中的string拼起来，但ϕ中无string. 所以 $= \phi$.

10) $\phi^* = \{\varepsilon\}$ // 0或多个ϕ中string拼起来，若0个，为ε.

$\Sigma^* = \varepsilon \cup \Sigma^1 \cup \Sigma^2 \cup \cdots$

Exercise

0) length of w is a multiple of 3 $(\Sigma\Sigma\Sigma)^*$

1) w does contain substring 110 $\Sigma^*110\Sigma^*$

2) w doesn't not contain substring 00

3) the number of 1's is a multiple of 3

4) w contains at least two 1's and one 0's.

$\Sigma^*1\Sigma^*1\Sigma^*0\Sigma^* \cup \Sigma^*0\Sigma^*1\Sigma^*1\Sigma^* \cup \Sigma^*1\Sigma^*0\Sigma^*1$

2) $(0 \cup \varepsilon)(10 \cup 1)^*$ $1^*01^+01^+\cdots1^+01^*$

3) $(0^*10^*10^*1)^*0^*$

$a \in \Sigma$, ε, ϕ.

$L_1 L_2$, $L_1 \cup L_2$, L_1^*, L_1^+

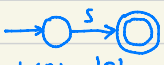
Theorem 2.17 A language is regular if and only if some regular expression describes it.

Lemma 2.18 If a language is described by a regular expression, then it is a regular language.

Proof. Start with a regular expression for L, then build an NFA to recognize L.

By the definition of regular expression,

1. $R = s$, where $s \in \Sigma$. $L(R) = \{s\}$.

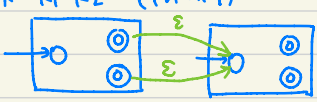


2. $R = \varepsilon$. $L(R) = \{\varepsilon\}$

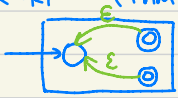
3. $R = \phi$. $L(R) = \phi$

4. $R = R_1 \cup R_2$ (Th 2.8

5. $R = R_1 \circ R_2$ (Th 2.14)



6. $R = R_1^*$ (Thm 2.15)

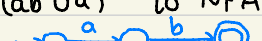


7. $R = R_1^+ = R_1^* \circ R_1$ (by (6), (5))

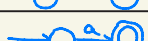
□

Example. $(ab \cup a)^*$ to NFA.

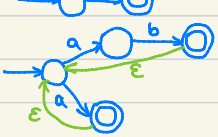
ab

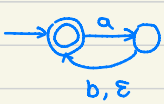


a



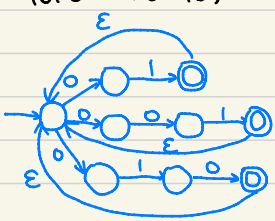
$(ab \cup a)^*$



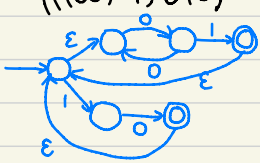


Exercise. Regular expression to NFA.

0) $(01 \cup 001 \cup 010)^*$



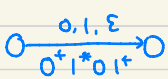
1) $((00)^*1) \cup 10)^*$



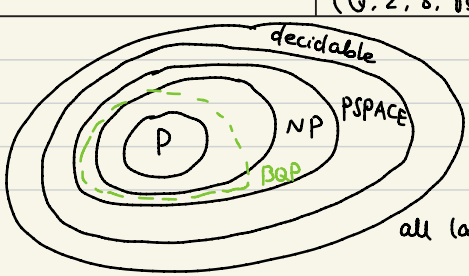
Lemma 2.19 If a language is regular, then it is described by a regular expression.

idea: reduce the number of states in a GNFA one by one. Finally, we get a two-state GNFA and we are done.

GNFAs are simply NFAs where the transition arrows may have any regular expression as labels instead of Σ_ε.



Def 2.20 (GNFA) A generalized nondeterministic finite automaton (GNFA) is a 5-tuple $(Q, \Sigma, \delta, q_{start}, q_{accept})$ where

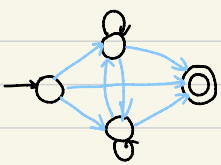


量子计算机多项式时间内可解决的问题

Bounded-error quantum polynomial time

q_{accept} 无出边, q_{start} 无入边

1. Q is a finite number of states
2. Σ is the alphabet → R
3. $\delta: (Q - \{q_{accept}\}) \times R \longrightarrow (Q - \{q_{start}\})$ is the transition function, where R denotes the set of regular expressions.
4. q_{start} is the start state
5. q_{accept} is the accept state.



A DFA can be converted to a GNFA as follows:

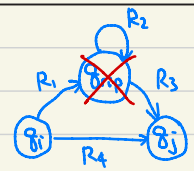
1. Add a new start state, with an ε-arrow to the old start state.
2. Add a new accept state, with ε-arrows from the old accept states.
3. Replace multiple edges by one, using $\cup$
4. Add arrows labled by ϕ between states that had no arrows.

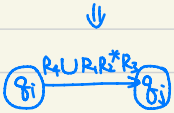
Proof. Overall strategy

1) Convert an NFA to a GNFA
2) Given a GNFA G, construct an equivalent GNFA G' by removing one state
3) Repeat step 2) until there are 2 states, from which we obtain the desired regular expression.

Let $G = (Q, \Sigma, \delta, q_{start}, q_{accept})$ be a GNFA.

Let $G' = (Q', \Sigma, \delta', q_{start}, q_{accept})$ be a GNFA by removing one state





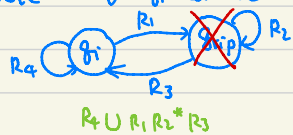
from G, called q_{rip}. Where $q_{rip} \in Q \setminus \{q_{start}, q_{accept}\}$ is arbitrary.

Let $Q' = Q \setminus \{q_{rip}\}$.

For any $q_i \in Q \setminus \{q_{accept}\}$, for any $q_j \in Q \setminus \{q_{start}\}$, let

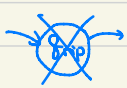
$\delta'(q_i, q_j) = \delta(q_i, q_j) \cup \delta(q_i, q_{rip})\, \delta(q_{rip}, q_{rip})^* \delta(q_{rip}, q_j)$

Note that q_i can be equal to q_j.



$R_4 \cup R_1 R_2^* R_3$

We want to prove $L(G) = L(G')$



Claim 2.21 $L(G) \subseteq L(G')$

Proof (of the claim): Let $w \in L(G)$. We want to prove $w \in L(G')$.

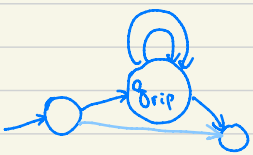
Since $w \in L(G)$, $\exists$ states $r_1, r_2, \cdots, r_{m+1}$, and $\exists w_1, \cdots, w_m \in \Sigma^*$ such that $w = w_1 w_2 \cdots w_m$ satisfying

1. $r_1 = q_{start}$
2. $r_{m+1} = q_{accept}$
3. $w_i \in \delta(r_i, r_{i+1})$ for $i = 1, 2, \cdots, m$

Case 1. None of $r_1, \cdots, r_{m+1}$ is q_{rip}. Thus, each r_i is also a state in G'. By the definition δ', $\delta(r_i, r_{i+1})$ is a subset of $\delta'(r_i, r_{i+1})$. Therefore, $r_1, \cdots, r_m \in Q'$ is also a valid transition for G'.

Case 2. At least one of $r_1, \cdots, r_m$ is q_{rip}.

Suppose q_{rip} appears as k consecutive blocks, each of length $t_1, \cdots, t_k \geq 1$. That is,



$L_1: r_{i_1} = q_{rip},\ r_{i_1+1} = q_{rip},\ \cdots,\ r_{i_1+t_1-1} = q_{rip}$

$L_2:$

$\vdots$

$L_k: r_{i_k} = q_{rip},\ r_{i_k+1} = q_{rip},\ \cdots,\ r_{i_k+t_k-1} = q_{rip}$

$r_1, r_2, \cdots, r_m$

$\cdots$ L_1 $\cdots$ L_2 $\cdots$ L_3 $\cdots$

By removing all occurrences of q_{rip}, i.e. $L_1, \cdots, L_k$, we obtain a new sequence of states.

$r_1, r_2, \cdots, r_{i-1}$

$r_{i_1+t_1}, r_{i_1+t_1+1}, \cdots, r_{i_2-1}$

$r_{i_2+t_2}, r_{i_2+t_2+1}, \cdots, r_{i_3-1},$

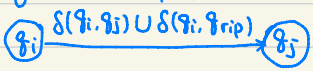
$\cdots$

$r_{i_k+t_k}, r_{i_k+t_k+1}, \cdots, r_m$

} valid state transition

Claim 2.22 $L(G') \subseteq L(G)$

Proof of the claim. Let $w \in L(G')$. We prove $w \in L(G)$



There exists a sequence of states, denoted by $r_1, r_2, \cdots, r_{m+1}$ strings $w_1, w_2, \cdots, w_m \in \Sigma^*$, where $w = w_1 w_2 \cdots w_m$, such that

1. $r_1 = q_{start}$
2. $r_{m+1} = q_{accept}$,
3. $w_i \in \delta'(r_i, r_{i+1})$ for all $i = 1, 2, \cdots, m$.

By the definition of $\delta'(r_i, r_{i+1})$, we have either $w_i \in \delta(r_i, r_{i+1})$

or $w_i \in \delta(r_i, q_{rip})\,\delta(q_{rip}, q_{rip})^* \delta(q_{rip}, r_{i+1})$.
If $w_i \in \delta(r_i, r_{i+1})$, we do nothing.
If $w_i \in \delta(r_i, q_{rip})\delta(q_{rip}, q_{rip})^u \delta(q_{rip}, r_{i+1})$ for some integer $u \geq 0$,
we replace states r_i, r_{i+1} in the sequence by
$r_i, q_{rip}, q_{rip}, \cdots q_{rip}, r_{i+1}$

□

2.4 Nonregular languages

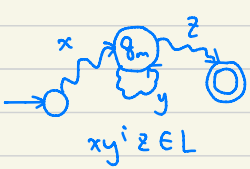
"counting"

Theorem 2.23 Almost all languages are nonregular.

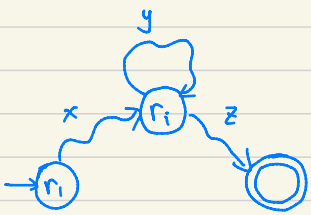
languages $= 2^{|\mathbb{N}|} = \mathcal{N}_1$
$\Sigma^* = \mathcal{N}_0$
regular languages $= \mathcal{N}_0$.

Lemma 2.24 (Pumping lemma) Let L be a regular language. Then there exists $p \in \mathbb{N}$ such that, for any string s with $|s| \geq p$, we can write $s = xyz$, $x, y, z \in \Sigma^*$, satisfying

1. $xy^i z \in L$ for any $i \geq 0$
2. $|y| > 0$
3. $|xy| \leq p$



Proof. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA that recognizes L.
Let $p = |Q|$. Let $s = s_1 s_2 \cdots s_n$ be a string of length n,



where $n \geq p$. Let $r_1, r_2, \cdots, r_{n+1} \in Q$ be the states visited.
That is, $r_1 = q_0$, $r_{i+1} = \delta(r_i, s_i)$ for $i = 1, \cdots, n$.
By the pigeonhole principle, two states must be the same.
Consider the first occurrence of repeated states. Say $r_i = r_j$, $i < j$.

Let $x = s_1 \cdots s_i$ $\quad xy^iz \in L$
$y = s_{i+1} \cdots s_j$ $\quad |y| > 0$
$z = s_{j+1} \cdots s_n$ $\quad |xy| \leq p$ ✓

One can verify 1), 2), 3) are satisfied.

□

Example. $\{0^n 1^n : n \geq 0\}$ is not regular.
Assume it is regular. By the Pumping Lemma,
$\exists p$ s.t. these conditions are satisfied.
Let $s = 0^p 1^p$

$x = 0^k$, $y = 0^t$
$xy^iz = 0^{p+j} 1^p \in L$. $\quad \begin{cases} xy^iz \in L \\ |y| > 0 \\ |xy| \leq p. \end{cases}$

contradiction.

Exercise
1) $\{w : w$ has an equal number of 0's and 1's$\}$
2) $\{ww : w \in \Sigma^*\}$
3) Palindrome $\{w \in \Sigma^* : w = w^R\}$

1. The string y consists only of 0s. In this case, the string $xyyz$ has more 0s than 1s and so is not a member of B, violating condition 1 of the pumping lemma. This case is a contradiction.
2. The string y consists only of 1s. This case also gives a contradiction.
3. The string y consists of both 0s and 1s. In this case, the string $xyyz$ may have the same number of 0s and 1s, but they will be out of order with some 1s before 0s. Hence it is not a member of B, which is a contradiction.

Thus a contradiction is unavoidable if we make the assumption that B is regular, so B is not regular. Note that we can simplify this argument by applying condition 3 of the pumping lemma to eliminate cases 2 and 3.

In this example, finding the string s was easy because any string in B of length p or more would work. In the next two examples, some choices for s do not work so additional care is required. ■