Homework 1 - 5

劉韋成 蘇俊杰

Menu

HW#1 Problem 1

(Exercise 0.7; 30 points) For each part, give a binary relation that satisfies the condition. *Please illustrate the relation using a directed graph.*

(a) Reflexive and symmetric but not transitive

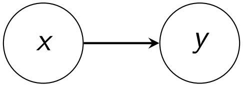
(b) Reflexive and transitive but not symmetric

(c) Symmetric and transitive but not reflexive

HW#1 Problem 1

Directed graph of a binary relation $\mathcal{R}$:

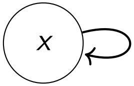
$x \mathcal{R} y$:



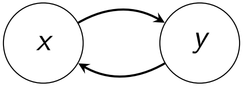
HW#1 Problem 1

Let $\mathcal{R}$ be a binary relation on a set S:

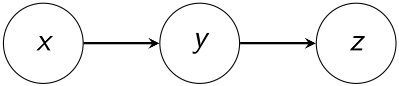
Reflexive: $\forall x \in S,\ x \,\mathcal{R}\, x$.



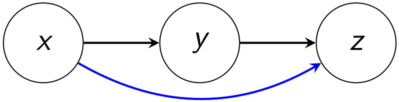
Symmetric: $\forall x, y \in S,\ x \,\mathcal{R}\, y$ iff $y \,\mathcal{R}\, x$.



Transitive: $\forall x, y, z \in S,\ x \,\mathcal{R}\, y$ and $y \,\mathcal{R}\, z$ implies $x \,\mathcal{R}\, z$.

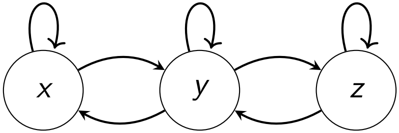


implies



HW#1 Problem 1

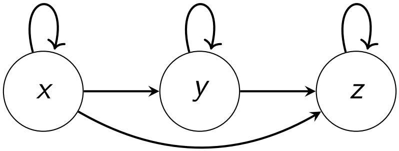
(a) Reflexive and symmetric but not transitive



$$\begin{tikzcd}
x \arrow[loop above] \arrow[r, bend left] & y \arrow[loop above] \arrow[l, bend left] \arrow[r, bend left] & z \arrow[loop above] \arrow[l, bend left]
\end{tikzcd}$$

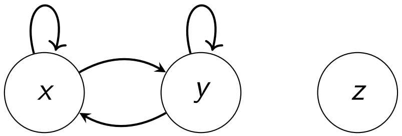
HW#1 Problem 1

(b) Reflexive and transitive but not symmetric


$$\begin{tikzcd}x \arrow[loop above] \arrow[r] \arrow[rr, bend right] & y \arrow[loop above] \arrow[r] & z \arrow[loop above]\end{tikzcd}$$

HW#1 Problem 1

(c) Symmetric and transitive but not reflexive


$$\begin{tikzcd} x \arrow[loop above] \arrow[r, bend left] & y \arrow[loop above] \arrow[l, bend left] & z \end{tikzcd}$$

HW#1 Problem 2

(20 points) For each part, determine whether the binary relation on the set of reals or integers is an equivalence relation. If it is, please provide a proof; otherwise, please give a counterexample.

(a) The two numbers have a common divisor other than 1.

(b) For a fixed non-zero divisor, the two numbers have the same remainder. (Note: for instance, suppose 2 is the divisor. Numbers 4 and 6 have the same remainder, while 4 and 5 do not.)

HW#1 Problem 2

A binary relation $\mathcal{R}$ on a set S is an equivalence relation if

- $\mathcal{R}$ is reflexive: $\forall x \in S$, $x \mathcal{R} x$,
- $\mathcal{R}$ is symmetric: $\forall x, y \in S$, $x \mathcal{R} y$ iff $y \mathcal{R} x$, and
- $\mathcal{R}$ is transitive: $\forall x, y, z \in S$, $x \mathcal{R} y$ and $y \mathcal{R} z$ implies $x \mathcal{R} z$.

HW#1 Problem 2

(a) $\mathcal{R}$: The two numbers have a common divisor other than 1.

- Reflexive: satisfied, $\forall x > 1 \in \mathbb{N}$, x and x must have a common divisor, which is x itself.
- Symmetric: satisfied, $\forall x, y > 1 \in \mathbb{N}$, if x and y have a common divisor d other than 1, y and x must have a common divisor d, too.
- Transitive: violated, counterexample: 4 and 6 have a common divisor 2, 6 and 9 have a common divisor 3, but 4 and 9 do not have any common divisors other than 1.

So $\mathcal{R}$ is not an equivalence relation.

HW#1 Problem 2

(b) $\mathcal{R}_d$: For a fixed non-zero divisor d, the two numbers have the same remainder.

- Reflexive: satisfied, $\forall x > 1 \in \mathbb{N}$, x and x must have the same remainder for any fixed non-zero divisors.
- Symmetric: satisfied, $\forall x, y > 1 \in \mathbb{N}$, if x and y have the same remainder for a fixed non-zero divisor, y and x must have, too.
- Transitive: satisfied, $x \, \mathcal{R}_d \, y$ means that x and y have a same remainder r_1 for the divisor d, and $y \, \mathcal{R}_d \, z$ means that y and z have a same remainder r_2 for the divisor d. Through y, we can know r_1 and r_2 are all the remainders of dividing y by d, namely, $r_1 = r_2$, so x and z have the same remainder, that is, $x \, \mathcal{R}_d \, z$.

So $\mathcal{R}_d$ is an equivalence relation.

HW#1 Problem 3

(20 points) In class, following Sipser's book, we first studied the formal definition of a function and then treated relations as special cases of functions. Please give instead a direct definition of relations and then define functions as special cases of relations. Your definitions should cover the arity of a relation or function and also the meaning of the notation $f(a) = b$.

HW#1 Problem 3

- A relation $\mathcal{R}$ is a subset of Cartesian product of several sets.
- A relation $\mathcal{R} \subseteq A_1 \times A_2 \times \cdots \times A_k$ is called a *k-ary* relation.
- A 2-*ary* relation is usually called a binary relation.

HW#1 Problem 3

- A function is a binary relation that follows the form $f \subseteq (A_1 \times \cdots \times A_k) \times B$, namely the first element of a function is also a *k-ary* relation.
- For all the first elements t of a function f, $[(t, a), (t, b) \in f]$ implies $a = b$.
- A function with a *k-ary* relation as its first element is called a *k-ary* function.
- We write $f(a) = b$ if $(a, b) \in f$. Similarly, we write $f(a_1, a_2, \cdots, a_k) = b$ if $((a_1, a_2, \cdots, a_k), b) \in f$

HW#1 Problem 4

Proved by contradiction.

Supposed that there is a graph having no nodes with the same degree.

No nodes with the same degree means that : each node has distinct degree from 0 to $n-1$.
the node with $n-1$ degree must be connected by every other nodes in this graph because no self loop in this gragh.
but in our assumption, there must be a node with 0 degree.
Contradiction happens.

HW#1 Problem 5

The proof is by induction on the number n of players.

Base case ($n = 2$): There are exactly two players, say A and B. Either A beat B, in which case we order them as A, B, or B beat A, in which case we order them as B, A.

HW#1 Problem 5

Induction step ($n > 2$): Pick any of the n players, say A. From the induction hypothesis, the other $n-1$ players can be ordered as $p_1, p_2, \cdots, p_{n-1}$ such that p_1 beat p_2, p_2 beat p_3, $\cdots$, and p_{n-2} beat p_{n-1}. We now exam the result of the match played between A and p_1. If A beat p_1, then we get a satisfying order $A, p_1, p_2, \cdots, p_{n-1}$. Otherwise ($p_1$ beat A), we continue to exam the result of the match played between A and p_2. If A beat p_2, then we get a satisfying order $p_1, A, p_2, \cdots, p_{n-1}$. Otherwise ($p_2$ beat A), we continue as before. We end up either with $p_1, p_2, \cdots, p_{i-1}, A, p_i, \cdots, p_{n-1}$ for some $i \leq n-1$ or eventually with $p_1, p_2, \cdots, p_{n-1}, A$ if A is beaten by every other player, in particular p_{n-1}.

HW#2 Problem 1

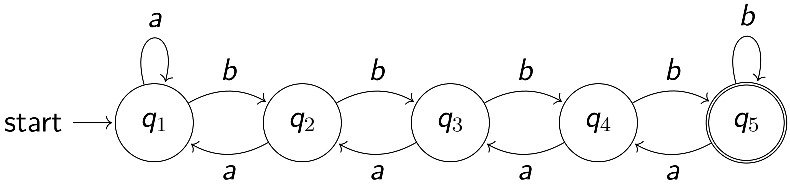
(Exercise 1.3; 10 points) The formal definition of a DFA M is $(\{q_1, q_2, q_3, q_4, q_5\}, \{\mathtt{a}, \mathtt{b}\}, \delta, q_1, \{q_5\})$ where δ is given by the following table. Draw the state diagram of M and give an intuitive characterization of the strings that M accepts.

	a	b
q_1	q_1	q_2
q_2	q_1	q_3
q_3	q_2	q_4
q_4	q_3	q_5
q_5	q_4	q_5

HW#2 Problem 1

$$\begin{tikzcd}
\text{start} \arrow[r] & q_1 \arrow[loop above, "a"] \arrow[r, bend left, "b"] & q_2 \arrow[l, bend left, "a"] \arrow[r, bend left, "b"] & q_3 \arrow[l, bend left, "a"] \arrow[r, bend left, "b"] & q_4 \arrow[l, bend left, "a"] \arrow[r, bend left, "b"] & q_5 \arrow[l, bend left, "a"] \arrow[loop above, "b"]
\end{tikzcd}$$

HW#2 Problem 1


$$\begin{tikzcd}
\text{start} \arrow[r] & q_1 \arrow[loop above, "a"] \arrow[r, bend left, "b"] & q_2 \arrow[l, bend left, "a"] \arrow[r, bend left, "b"] & q_3 \arrow[l, bend left, "a"] \arrow[r, bend left, "b"] & q_4 \arrow[l, bend left, "a"] \arrow[r, bend left, "b"] & q_5 \arrow[l, bend left, "a"] \arrow[loop above, "b"]
\end{tikzcd}$$

Intuitive characterization of the strings that M accepts:

Let n_a, n_b be the number of a and b. The DFA M accepts the strings that contain at least one suffix $X = x_1x_2x_3 \cdots x_k$ such that:

- $n_b = n_a + 4$ in X, and
- for all suffixes of X, $n_b \geq n_a$.

HW#2 Problem 1

$$\begin{tikzcd}
\text{start} \arrow[r] & q_1 \arrow[loop above, "a"] \arrow[r, bend left, "b"] & q_2 \arrow[l, bend left, "a"] \arrow[r, bend left, "b"] & q_3 \arrow[l, bend left, "a"] \arrow[r, bend left, "b"] & q_4 \arrow[l, bend left, "a"] \arrow[r, bend left, "b"] & \boxed{q_5} \arrow[l, bend left, "a"] \arrow[loop above, "b"]
\end{tikzcd}$$

e.q.

aaaaaaabbbbab: we can find a suffix $X = bbbbab$ such that:

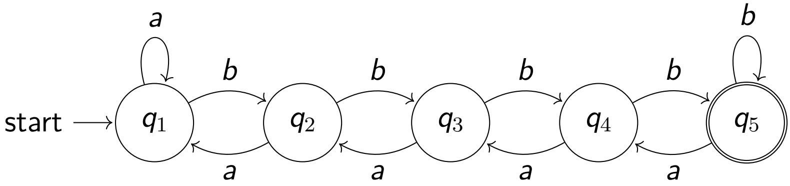
- $n_b = 5 = 1 + 4 = n_a + 4$, and
- for all suffixes of X: b, ab, bab, $bbab$, $bbbab$, $bbbbab$, $n_b \geq n_a$.

HW#2 Problem 1

$$\begin{tikzcd}
\text{start} \arrow[r] & q_1 \arrow[loop above, "a"] \arrow[r, bend left, "b"] & q_2 \arrow[l, bend left, "a"] \arrow[r, bend left, "b"] & q_3 \arrow[l, bend left, "a"] \arrow[r, bend left, "b"] & q_4 \arrow[l, bend left, "a"] \arrow[r, bend left, "b"] & \boxed{q_5} \arrow[l, bend left, "a"] \arrow[loop above, "b"]
\end{tikzcd}$$

What if *aaaaaaabbbbba*? We can only find one suffix $X = bbbbba$ such that $n_b = n_a + 4$, but the suffix *a* in X does not satisfy $n_b \geq n_a$.
What if *aaaaabbbbbaab*? We can only find one suffix $X = bbbbbaab$ such that $n_b = n_a + 4$, but the suffix *aab* in X does not satisfy $n_b \geq n_a$.
What if *aaaaabbbbaab*? We can not find any suffixes such that $n_b = n_a + 4$.

HW#2 Problem 1



$$\begin{tikzcd}
\text{start} \arrow[r] & q_1 \arrow[loop above, "a"] \arrow[r, bend left, "b"] & q_2 \arrow[l, bend left, "a"] \arrow[r, bend left, "b"] & q_3 \arrow[l, bend left, "a"] \arrow[r, bend left, "b"] & q_4 \arrow[l, bend left, "a"] \arrow[r, bend left, "b"] & q_5 \arrow[l, bend left, "a"] \arrow[loop above, "b"]
\end{tikzcd}$$

So the first condition [$n_b = n_a + 4$ in X] guarantee the input string ends in state q_5, and the second condition [for all suffixes of X, $n_b \geq n_a$] guarantee that even if it leaves from q_5, it will come back eventually.

HW#2 Problem 2

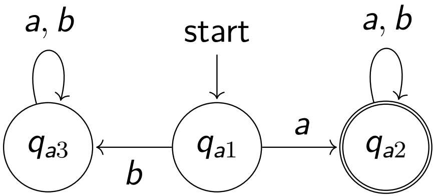
(Exercise 1.4; 20 points) Each of the following languages is the intersection of two simpler regular languages. In each part, construct DFAs for the simpler languages, then combine them using the construction discussed in class (see also Footnote 3 in Page 46 of [Sipser 2006, 2013]) to give the state diagram of a DFA for the language given. In all parts, the alphabet is $\{\texttt{a}, \texttt{b}\}$.

(a) $\{w \mid w$ starts with an $\texttt{a}$ and has at most one $\texttt{b}\}$.

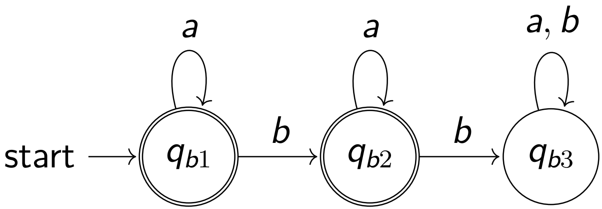
(b) $\{w \mid w$ has an odd number of $\texttt{a}$'s and ends with a $\texttt{b}\}$.

HW#2 Problem 2 (a)

Simpler language: $\{w \mid w \text{ starts with an } a\}$.

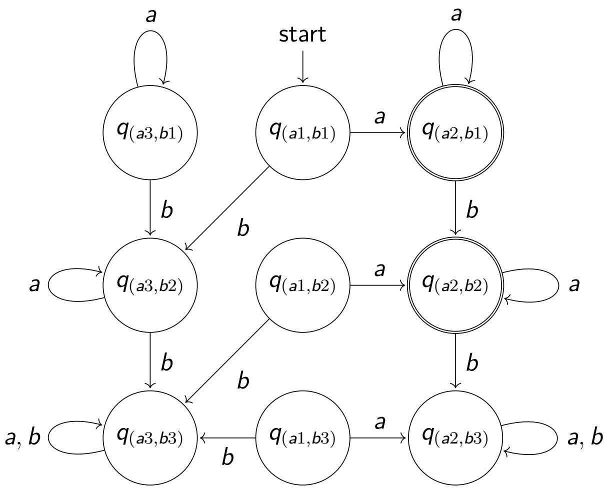


Simpler language: $\{w \mid w \text{ has at most one } b\}$.



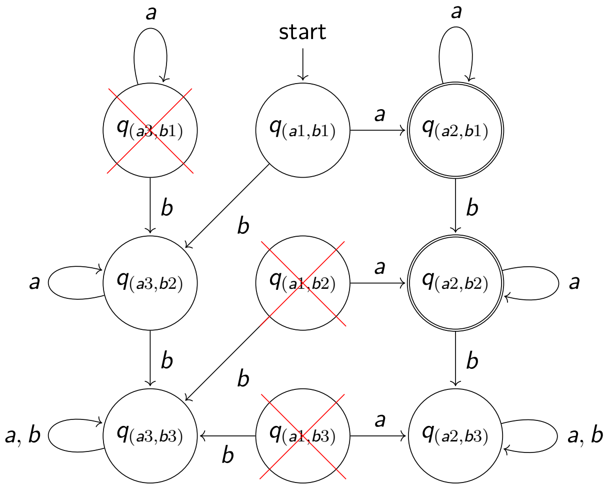
HW#2 Problem 2 (a)

Language: $\{w \mid w \text{ starts with an } a \text{ and has at most one } b\}$.



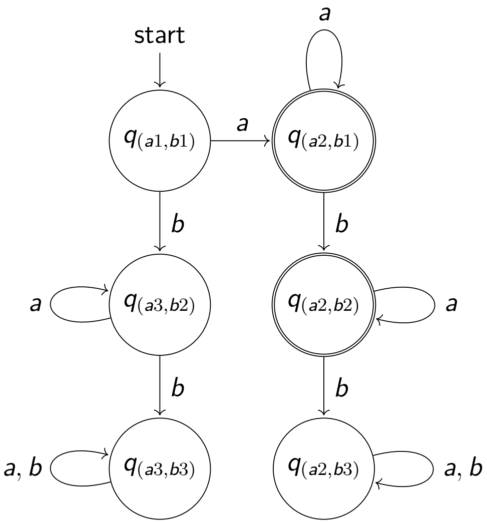
HW#2 Problem 2 (a)

Language: $\{w \mid w \text{ starts with an } a \text{ and has at most one } b\}$.



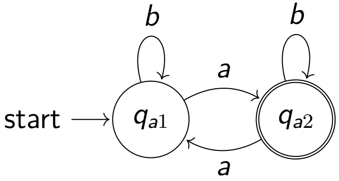
HW#2 Problem 2 (a)

Language: $\{w \mid w \text{ starts with an } a \text{ and has at most one } b\}$.

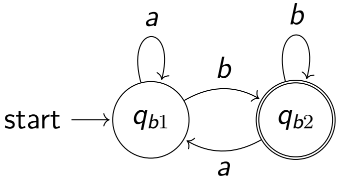


HW#2 Problem 2 (b)

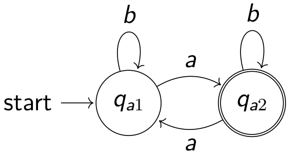
Simpler language: $\{w \mid w \text{ has an odd number of } a\text{'s}\}$.

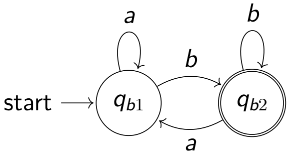


Simpler language: $\{w \mid w \text{ ends with a } b\}$.

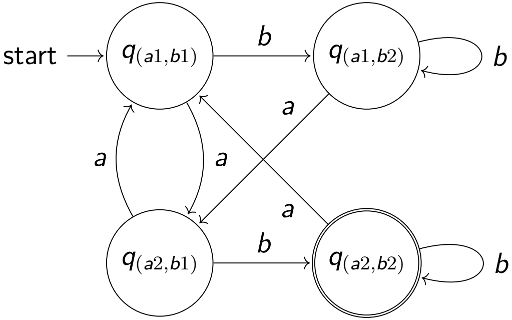


HW#2 Problem 2 (b)





Language: $\{w \mid w$ has an odd number of a's and ends with a $b\}$.



HW#2 Problem 3

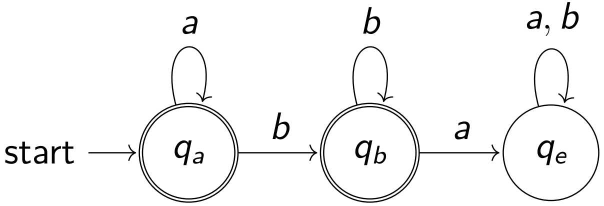
(Exercise 1.5; 20 points) Each of the following languages is the complement of a simpler regular language. In each part, construct a DFA for the simpler language, then use it to give the state diagram of a DFA for the language given. In all parts, the alphabet is $\{\mathtt{a}, \mathtt{b}\}$.

(a) $\{w \mid w$ is any string not in $\mathtt{a}^*\mathtt{b}^*\}$. (Note: $\mathtt{a}^*\mathtt{b}^*$ is a regular expression denoting $\{\mathtt{a}\}^* \circ \{\mathtt{b}\}^*$.)

(b) $\{w \mid w$ is any string that doesn't contain exactly two $\mathtt{a}$'s$\}$.

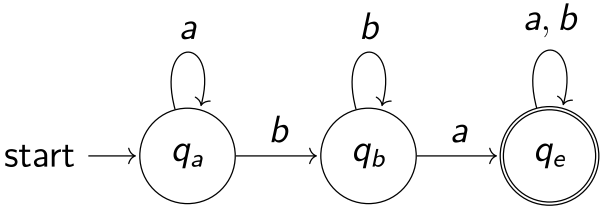
HW#2 Problem 3 (a)

Simpler language: $\{w \mid w \text{ is any string in } a^*b^*\}$.



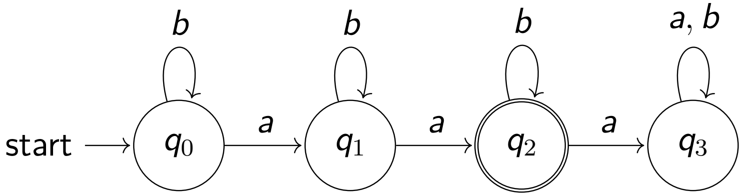
HW#2 Problem 3 (a)

Simpler language: $\{w \mid w \text{ is any string } \textbf{not} \text{ in } a^*b^*\}$.



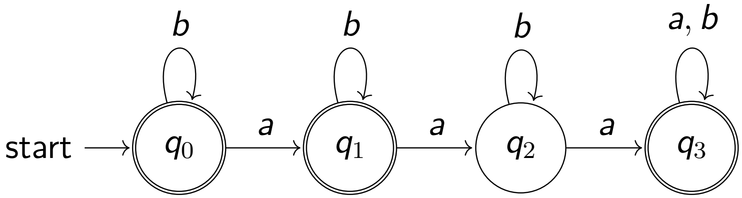
HW#2 Problem 3 (b)

Simpler language:
$\{w \mid w$ is any string that contains exactly two a's$\}$.


$$\begin{tikzcd}
\text{start} \arrow[r] & q_0 \arrow[loop above, "b"] \arrow[r, "a"] & q_1 \arrow[loop above, "b"] \arrow[r, "a"] & q_2 \arrow[loop above, "b"] \arrow[r, "a"] & q_3 \arrow[loop above, "{a, b}"]
\end{tikzcd}$$

HW#2 Problem 3 (b)

Simpler language:
$\{w \mid w$ is any string that doesn't contain exactly two a's$\}$.



HW#2 Problem 4

(Problem 1.36; 10 points) For any string $w = w_1 w_2 \cdots w_n$, the *reverse* of w, written w^R, is the string w in reverse order, $w_n \cdots w_2 w_1$. For any language A, let $A^R = \{w^R \mid w \in A\}$. Show that if A is regular, so is A^R.

HW#2 Problem 4

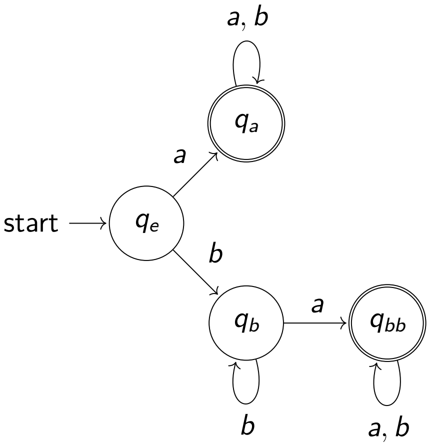
Let DFA M recognizes the language A, and we can construct a NFA M^R which recognizes A^R according to the following:

- M^R's states and alphabet are as same as M.
- Reverse all the transitions of M as the transitions of M^R. e.q. $\delta(q_1, a) = q_2 \rightarrow \delta(q_2, a) = q_1$.
- The accepting state of M^R is M's initial state.
- Add an additional initial state q_0 to M^R. Construct the translations from q_0 to all the accepting states of M with the label ϵ.

Because M^R recognizes A^R, A^R is regular.

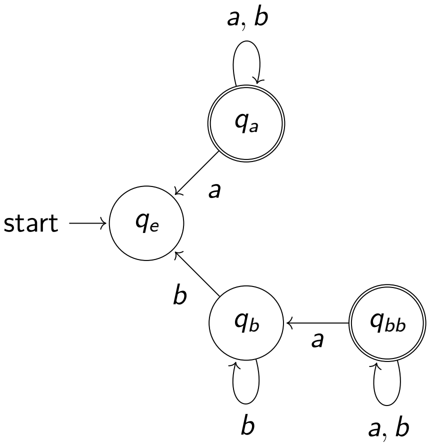
HW#2 Problem 4

e.q. : M



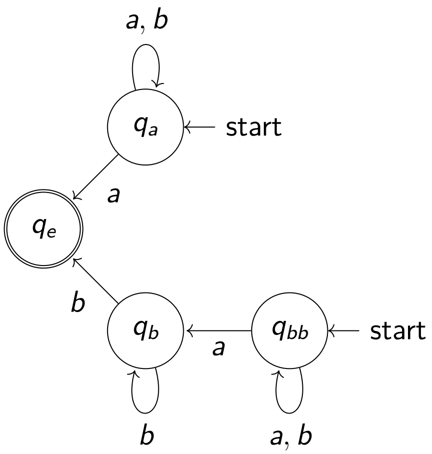
HW#2 Problem 4

Reverse all the translations:



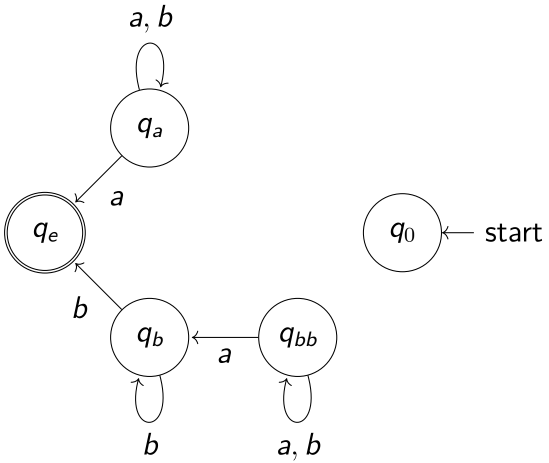
HW#2 Problem 4

Change the initial state into accepting state:



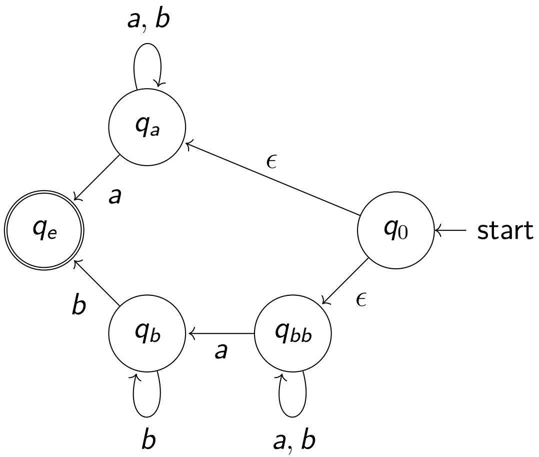
HW#2 Problem 4

Add an additional initial state q_0:



HW#2 Problem 4

Construct the translations from q_0 to all the accepting states of M with the label ϵ, then we can get the NFA M^R:



HW#2 Problem 5

(Problem 1.37; 20 points) Let

$$\Sigma_3 = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \ldots, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}.$$

Σ_3 contains all size 3 columns of `0`s and `1`s. A string of symbols in Σ_3 gives three rows of `0`s and `1`s. Consider each row to be a binary number and let

$$B = \{w \in \Sigma_3^* \mid \text{the bottom row of } w \text{ is the sum of the top two rows}\}.$$

For example,

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \in B, \text{but} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \notin B.$$

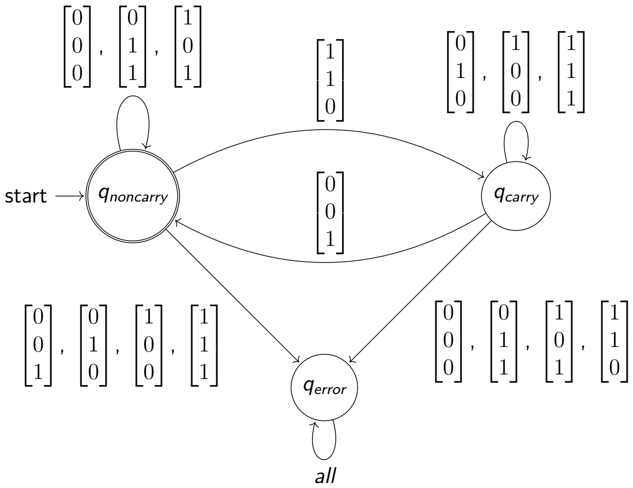
Show that B is regular. (Hint: working with B^R is easier. You may assume the result claimed in the previous problem (Problem 1.36).)

HW#2 Problem 5

Consider the situation of carry, starting from the tail of B is easier than starting from the head. So we first show that B^R is regular. We can construct a DFA that recognizes B^R when considering the carry and the correctness of calculation.

HW#2 Problem 5

The DFA that recognizes B^R:



$\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}$
$\begin{bmatrix}1\\1\\0\end{bmatrix}$
$\begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix}$
start
$q_{noncarry}$
$\begin{bmatrix}0\\0\\1\end{bmatrix}$
q_{carry}
$\begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix}$
$\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}$
q_{error}
all

HW#2 Problem 5

Because there is a DFA that recognizes B^R, B^R is regular. According to the result claimed in Problem 4 (if A is regular, so is A^R), we can say that $(B^R)^R = B$ is regular.

HW#2 Problem 6

(20 points) Generalize the proof of Theorem 1.25 of [Sipser 2006, 2013] (Pages 45–47) to handle A_1 and A_2 with different alphabets.

HW#2 Problem 6

Suppose $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ recognizes A_1 and $M_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$ recognizes A_2.
Construct $M = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$:

- $Q = \{(Q_1 \cup \{q_f\}) \times (Q_2 \cup \{q_f\})\}$.
- $\Sigma = \Sigma_1 \cup \Sigma_2$.
- $\delta((r_1, r_2), a) =$
$$\begin{cases} (\delta_1(r_1, a), \delta_2(r_2, a)) & \text{if } (r_1, r_2 \neq q_f) \text{ and } (a \in (\Sigma_1 \cap \Sigma_2)) \\ (\delta_1(r_1, a), q_f) & \text{if } (r_1 \neq q_f \wedge a \in \Sigma_1) \text{ and } (r_2 = q_f \vee a \notin \Sigma_2) \\ (q_f, \delta_2(r_2, a)) & \text{if } (r_2 \neq q_f \wedge a \in \Sigma_2) \text{ and } (r_1 = q_f \vee a \notin \Sigma_1) \\ (q_f, q_f) & \text{if } (r_1 = q_f \vee a \notin \Sigma_1) \text{ and } (r_2 = q_f \vee a \notin \Sigma_2) \end{cases}$$
- $q_0 = (q_1, q_2)$.
- $F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$.

HW#2 Problem 6

Why we need q_f?

Because when we read a character a that in Σ_1 but not in Σ_2, A_2 cannot recognize a so M_2 must fail and never accept. If there's no q_f, M cannot find out this situation.

HW#3 Problem 1

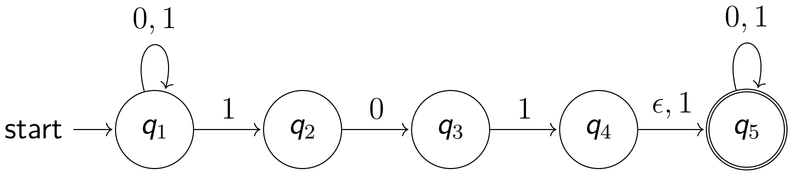
(Exercise 1.7; 10 points) For each of the following languages, give the state diagram of an NFA, with the specified number of states, that recognizes the language. In all parts, the alphabet is $\{\mathtt{0}, \mathtt{1}\}$.

(a) The language $\{w \mid w$ contains `101` or `1011` as a substring, i.e., $w = x(\mathtt{101}|\mathtt{1011})y$ for some x and $y\}$ with five states

(b) The language $1^*0^+1^*$ with three states

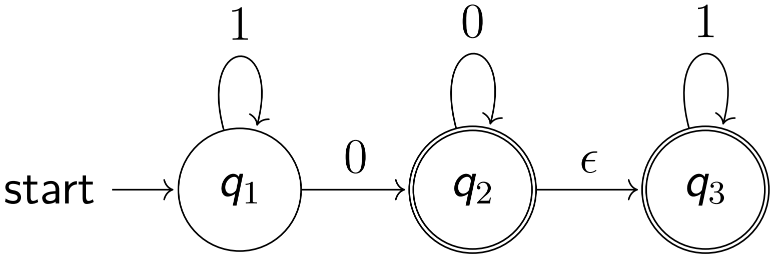
HW#3 Problem 1

(a) The language $\{\omega \mid \omega$ contains 101 or 1011 as a substring, i.e., $\omega = x(101|1011)y$ for some x and $y\}$ with five states.


$$\begin{tikzcd}
\text{start} \arrow[r] & q_1 \arrow[loop above, "{0,1}"] \arrow[r, "1"] & q_2 \arrow[r, "0"] & q_3 \arrow[r, "1"] & q_4 \arrow[r, "{\epsilon,1}"] & q_5 \arrow[loop above, "{0,1}"]
\end{tikzcd}$$

HW#3 Problem 1

(b) The language $1^*0^+1^*$ with three states.

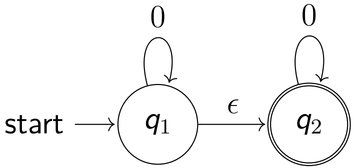

$$\begin{tikzcd}
\text{start} \arrow[r] & q_1 \arrow[loop above, "1"] \arrow[r, "0"] & q_2 \arrow[loop above, "0"] \arrow[r, "\epsilon"] & q_3 \arrow[loop above, "1"]
\end{tikzcd}$$

HW#3 Problem 2

(Exercise 1.14; 10 points) Show by giving an example that, if M is an NFA that recognizes language C, swapping the accept and nonaccept states in M doesn't necessarily yield a new NFA that recognizes the complement of C. Is the class of languages recognized by NFAs closed under complement? Explain you answer.

HW#3 Problem 2

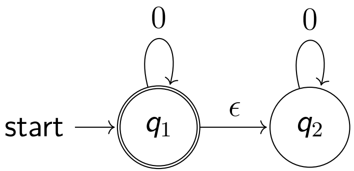
Give a example that swapping the accept and nonaccept states in an NFA does not necessarily yield a new NFA that recognizes the complement of the original language:


$$\begin{tikzcd}
\text{start} \arrow[r] & q_1 \arrow[loop above, "0"] \arrow[r, "\epsilon"] & q_2 \arrow[loop above, "0"]
\end{tikzcd}$$

The above NFA recognizes the string 0^*.

HW#3 Problem 2

Give a example that swapping the accept and nonaccept states in an NFA does not necessarily yield a new NFA that recognizes the complement of the original language:


$$\begin{tikzcd} \text{start} \arrow[r] & q_1 \arrow[loop above, "0"] \arrow[r, "\epsilon"] & q_2 \arrow[loop above, "0"] \end{tikzcd}$$

The above NFA still recognizes the string 0^*.

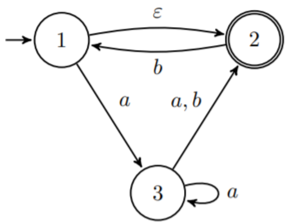
HW#3 Problem 2

Show that the class of languages recognized by NFAs is closed under complement:

Let the language L be the language recognized by an NFA M.
According to Theorem 1.39 on the slides, every NFA has an equivalent DFA. Let N be the equivalent DFA of M, the complement of N (written $\overline{N}$) recognizes the complement of L.
Similarly, every DFA has an equivalent NFA, so $\overline{N}$ must have an equivalent NFA, called D. In conclusion, the complement of L is still recognized by an NFA D, so the class of languages recognized by NFAs is closed under complement.

HW#3 Problem 3

(Exercise 1.16; 20 points) Use the construction given in Theorem 1.39 (every NFA has an equivalent DFA) to convert the following NFA into an equivalent DFA.

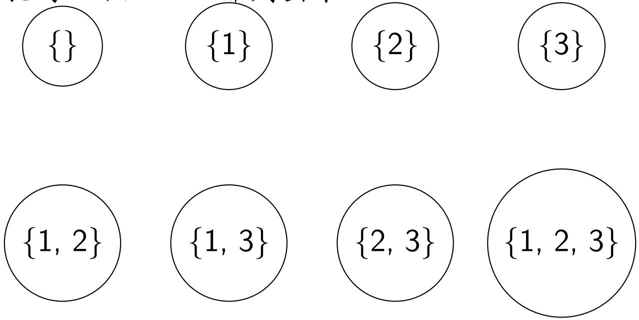


HW#3 Problem 3

利用 Th 1.39 的方法（subset construction）建構出等價的 DFA

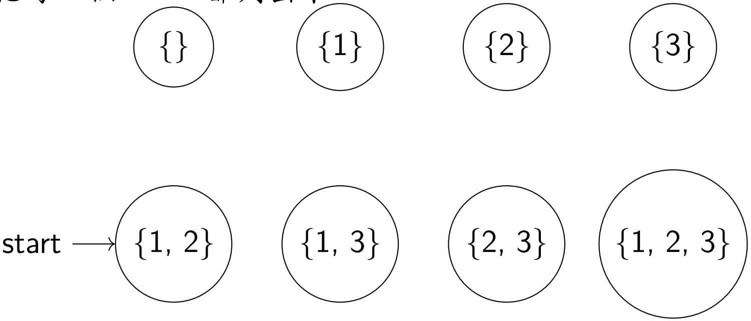
HW#3 Problem 3

把每一個 state 都列出來



HW#3 Problem 3

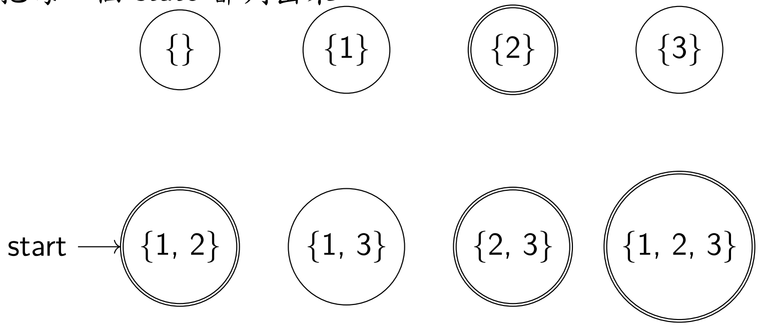
把每一個 state 都列出來



$q_0' = E(\{1\}) = \{1,2\}$

HW#3 Problem 3

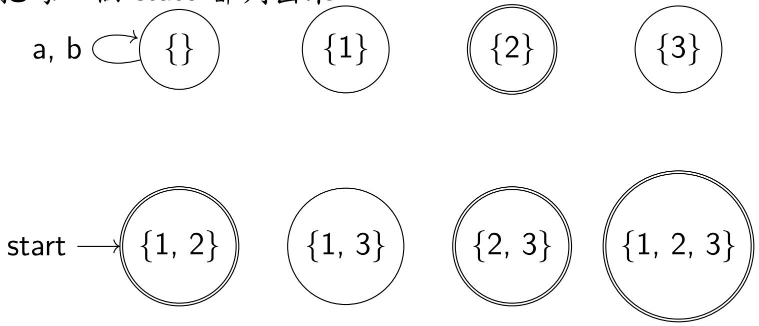
把每一個 state 都列出來



$F' = \{\{2\}, \{1,2\}, \{2,3\}, \{1,2,3\}\}$

HW#3 Problem 3

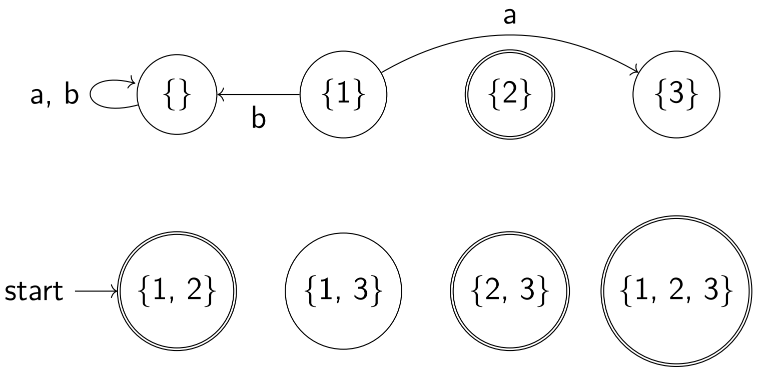
把每一個 state 都列出來



$\delta'(\{\}, a) = \{\}$
$\delta'(\{\}, b) = \{\}$

HW#3 Problem 3

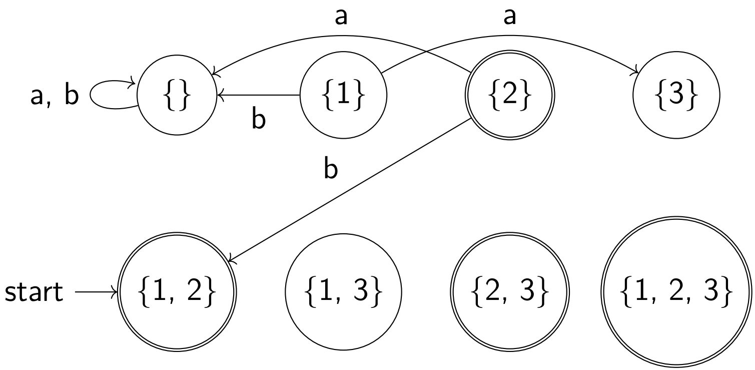
把每一個 state 都列出來



$\delta'(\{1\}, a) = \{3\}$
$\delta'(\{1\}, b) = \{\}$
注意我們並沒有把 {1} 偷偷展開成 {1, 2}

HW#3 Problem 3

把每一個 state 都列出來

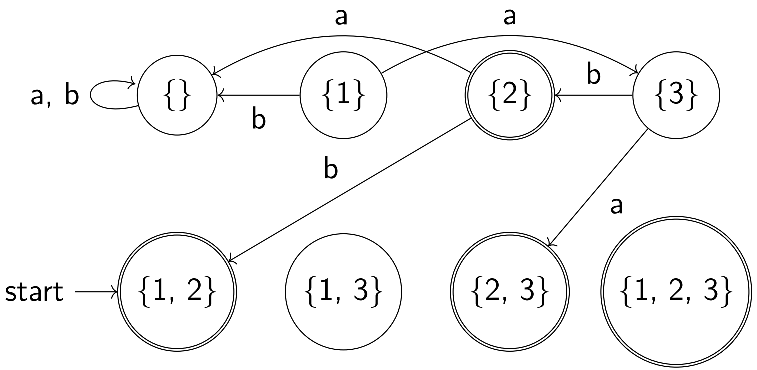


$$\begin{tikzcd}
\{\} \arrow[loop left, "a,b"] & \{1\} \arrow[l, "b"] \arrow[rr, bend left, "a"] & \{2\} \arrow[ll, bend right, "a"'] \arrow[dll, "b"] & \{3\} \\
\{1,2\} & \{1,3\} & \{2,3\} & \{1,2,3\}
\end{tikzcd}$$

$\delta'(\{2\}, a) = \{\}$
$\delta'(\{2\}, b) = \{1, 2\}$

HW#3 Problem 3

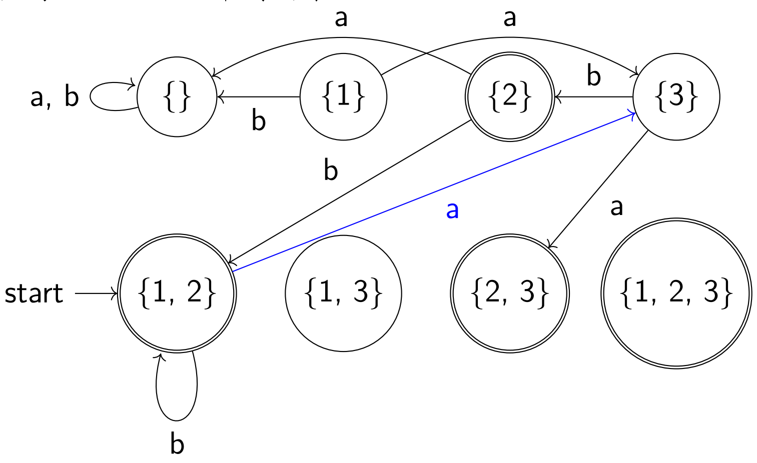
把每一個 state 都列出來



$$\delta'(\{3\}, a) = \{2, 3\}$$
$$\delta'(\{3\}, b) = \{2\}$$

HW#3 Problem 3

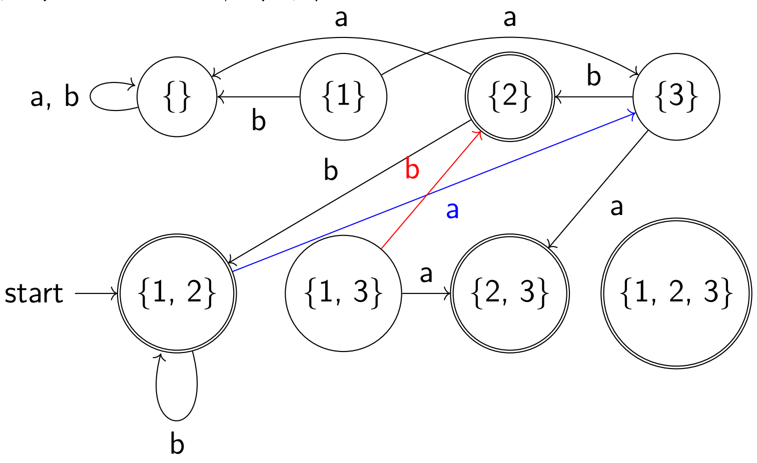
把每一個 state 都列出來



$\delta'(\{1,2\}, a) = \{3\}$
$\delta'(\{1,2\}, b) = \{1,2\}$

HW#3 Problem 3

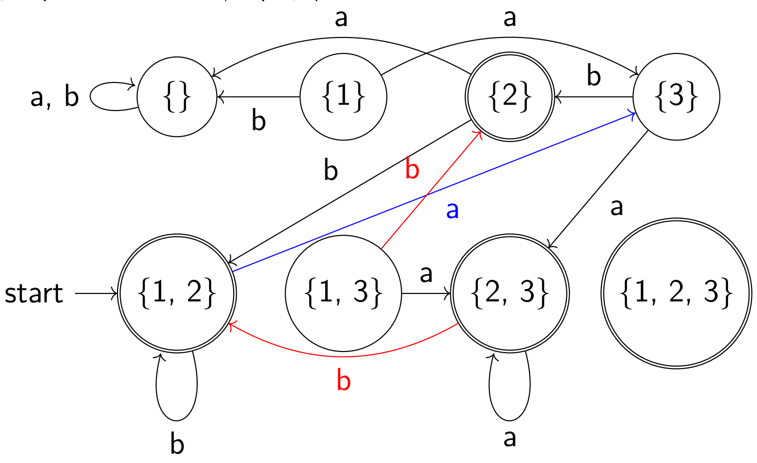
把每一個 state 都列出來



$$\delta'(\{1,3\}, a) = \{2,3\}$$
$$\delta'(\{1,3\}, b) = \{2\}$$

HW#3 Problem 3

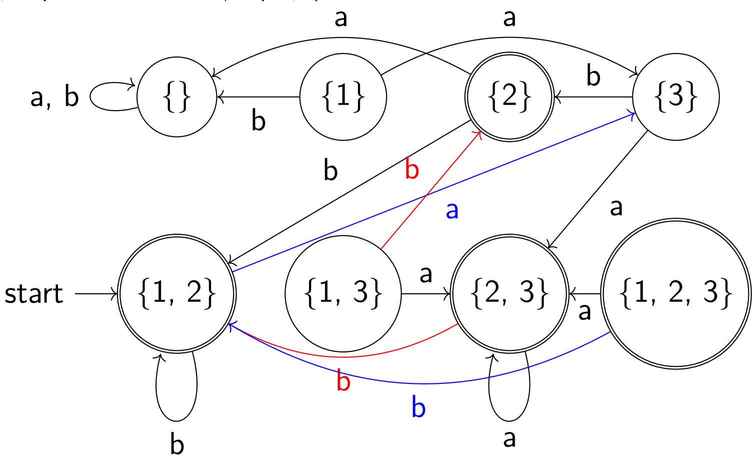
把每一個 state 都列出來



$\delta'(\{2,3\}, a) = \{2,3\}$
$\delta'(\{2,3\}, b) = \{1,2\}$

HW#3 Problem 3

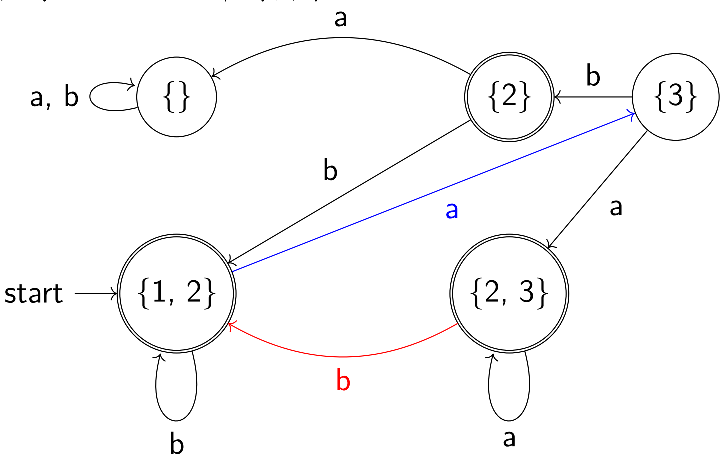
把每一個 state 都列出來



$$\delta'(\{1,2,3\}, a) = \{2,3\}$$
$$\delta'(\{1,2,3\}, b) = \{1,2\}$$

HW#3 Problem 3

把每一個 state 都列出來



刪去無法到達的狀態

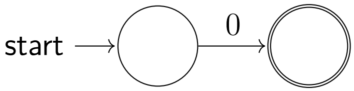
{1}，{1, 3} 和 {1, 2, 3} 這三個節點一定無法到達

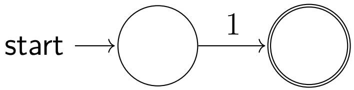
HW#3 Problem 4

(Exercise 1.18; 10 points) Use the procedure described in Lemma 1.55 to convert the regular expression $(0 \cup 1)^*110(0 \cup 1)^*$ into an NFA.

HW#3 Problem 4

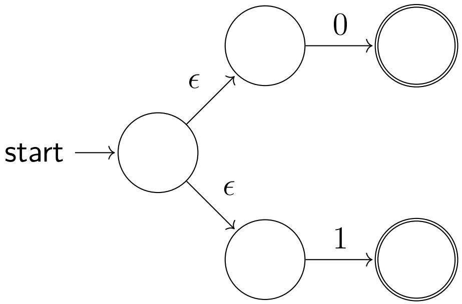
0 1





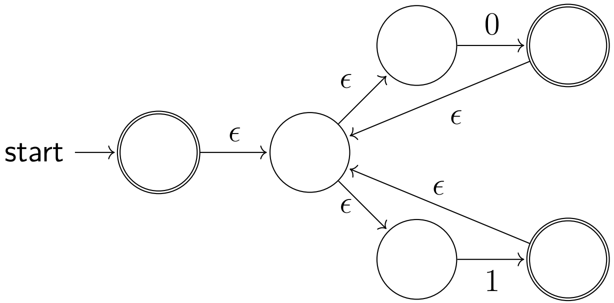
HW#3 Problem 4

$0 \cup 1$



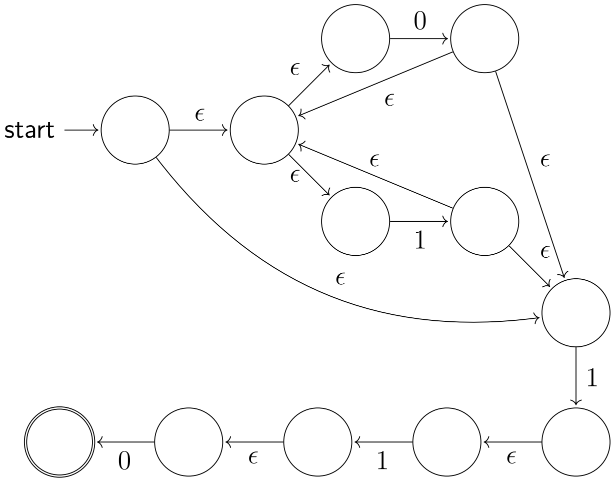
HW#3 Problem 4

$(0 \cup 1)^*$



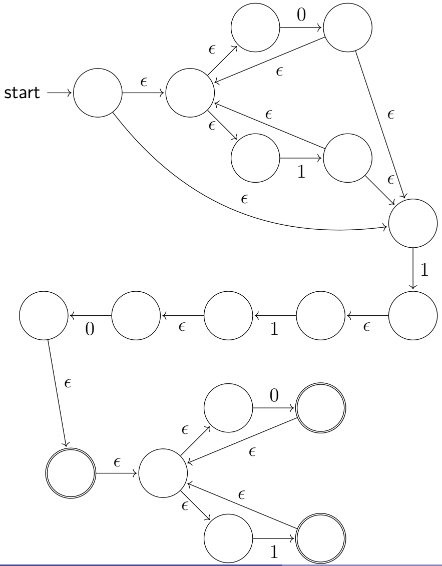
HW#3 Problem 4

$(0 \cup 1)^*110$



HW#3 Problem 4

$(0 \cup 1)^*110(0 \cup 1)^*$



5. (Exercise 1.20; 10 points) Give regular expressions generating the following languages, where the alphabet is $\{\texttt{0}, \texttt{1}\}$:

   (a) $\{w \mid \text{every odd position of } w \text{ is a } \texttt{1}\}$ (Note: see w as $w_1 w_2 \cdots w_n$, where $w_i \in \{\texttt{0}, \texttt{1}\}$)

   (b) $\{w \mid w \text{ doesn't contain the substring } \texttt{011}\}$

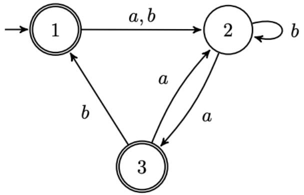
HW#3 Problem 5

第一小題：奇數位為 1
只需要注意的是：偶數為不受任何限制
我們分兩部分處理：
第一部分處理一般情況
第二部分處理 w 只有一位數的情況 $(1(0 \cup 1))^*(1 \cup \epsilon)$

HW#3 Problem 5

第二小題：字串不包含子字串 011
只要出現 0，後面就不能出現 2 個以上的 1
暗示第一次遇見 0 之前可以有一堆 1（開頭為 1^*）
後面則可以看成有一堆 0，隨便在其右方加上一個 1
有被添加 1 的 0，把它看做一個整體 01，沒有的則是 0
於是右邊可以變成 $(0 \cup 01)^*$
合起來就是 $1^*(0 \cup 01)^*$

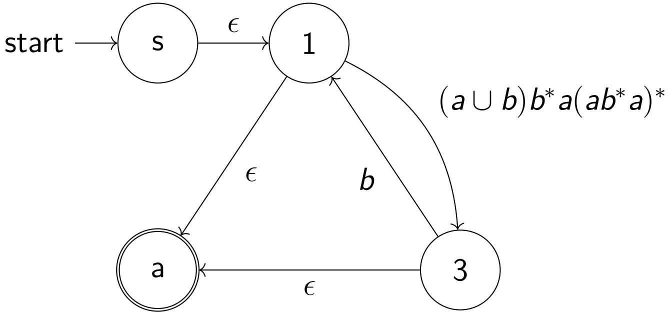
6. (Exercise 1.21; 20 points) Use the procedure described in Lemma 1.60 to convert the following finite automaton into a regular expression.



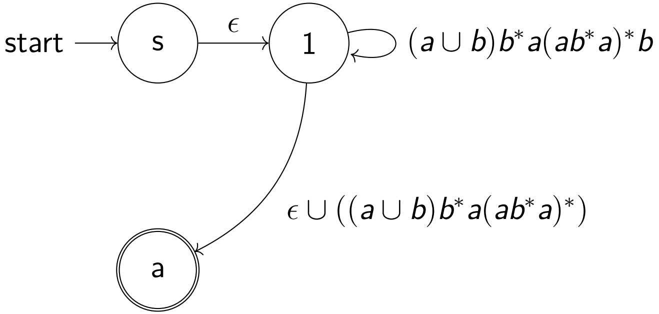
HW#3 Problem 6

$$\begin{tikzcd}
\text{start} \arrow[r] & s \arrow[r, "\epsilon"] & 1 \arrow[rr, "a \cup b"] \arrow[dl, "\epsilon"] & & 2 \arrow[loop right, "b"] \arrow[dl, bend left, "a"] \\
 & a & & 3 \arrow[ll, "\epsilon"] \arrow[ul, "b"] \arrow[ur, bend left, "a"] &
\end{tikzcd}$$

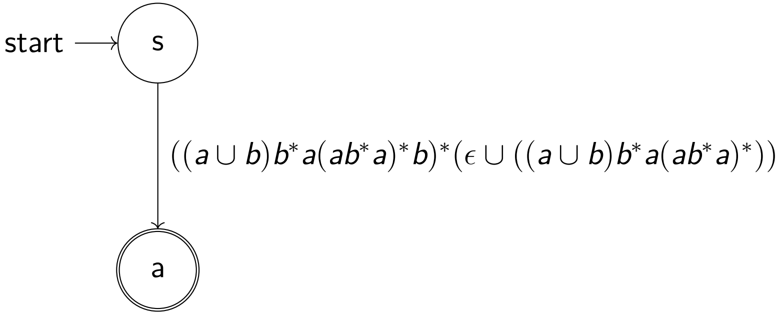
HW#3 Problem 6



HW#3 Problem 6

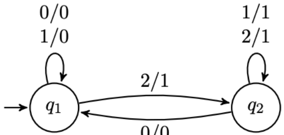

start
s
ϵ
1
$(a \cup b)b^*a(ab^*a)^*b$
$\epsilon \cup ((a \cup b)b^*a(ab^*a)^*)$
a

HW#3 Problem 6

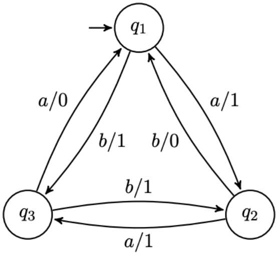

start
s
$((a \cup b)b^* a(ab^* a)^* b)^* (\epsilon \cup ((a \cup b)b^* a(ab^* a)^*))$
a

本題根據不同的刪除 state 過程，而有不同的答案！

7. (Exercise 1.24; 10 points) A *finite-state transducer* (FST) is a type of deterministic finite automaton whose output is a string rather than *accept* or *reject*. The following are state diagrams of finite state transducers T_1 and T_2.



T_1



T_2

Each transition of an FST is labeled with two symbols, one designating the input symbol for that transition and the other designating the output symbol. The two symbols are written with a slash, /, separating them. In T_1, the transition from q_1 to q_2 has input symbol 2 and output symbol 1. Some conditions may have multiple input-output pairs, such as the transition in T_1 from q_1 to itself. When an FST computes on an input string w, it takes the input symbols $w_1 \cdots w_n$ one by one and, starting from the start state, follows the transitions by matching the input labels with the sequence of symbols $w_1 \cdots w_n = w$. Every time it goes along a transition, it outputs the corresponding output symbol. For example, on

input 2212011, machine T_1 enters the sequence of states $q_1, q_2, q_2, q_2, q_2, q_1, q_1, q_1$ and produces output 1111000. On input `abbb`, T_2 outputs 1011. Give the sequence of states entered and the output produced in each of the following parts.

(a) T_1 on input 120221

(b) T_2 on input `baabba`

HW#3 Problem 7

q_1
q_1 吃 1 輸出 0 跑到 q_1
q_1 吃 2 輸出 1 跑到 q_2
q_2 吃 0 輸出 0 跑到 q_1
q_2 吃 2 輸出 1 跑到 q_2
q_1 吃 2 輸出 1 跑到 q_2
q_2 吃 1 輸出 1 跑到 q_2
輸出 010111

HW#3 Problem 7

q_1
q_1 吃 b 輸出 1 跑到 q_3
q_3 吃 a 輸出 0 跑到 q_1
q_1 吃 a 輸出 1 跑到 q_2
q_2 吃 b 輸出 0 跑到 q_1
q_3 吃 b 輸出 1 跑到 q_3
q_2 吃 a 輸出 0 跑到 q_1
輸出 101010

8. (Exercise 1.25; 10 points) Read the informal definition of the finite state transducer given in Exercise 1.24. Give a formal definition of this model, following the patterns in Definition 1.5 (Page 35 in Sipser's book or Page 7 of the slides). Assume that an FST has an input alphabet Σ and an output alphabet Γ but not a set of accept states. Include a formal definition of the computation of an FST. (Hint: an FST is a 5-tuple. Its transition function is of the form $\delta : Q \times \Sigma \longrightarrow Q \times \Gamma$.)

HW#3 Problem 8

An FST T is a 5-tuple $(Q, \Sigma, \Gamma, \delta, q_0)$
Q is a finite set of states
Σ is a finite set of input symbols
Γ is a finite set of output symbols
$\delta : Q \times \Sigma \to Q \times \Gamma$ is the transition function
$q_0 \in Q$ is the start state

Let $w = w_1 w_2 ... w_n$ be a string over Σ and $x = x_1 x_2 ... x_n$ a string over Γ
We say T produces output x on input w with the sequence of states $r_0, r_1, ..., r_n$ when

- $r_0 = q_0$
- $\delta(r_i, w_{i+1}) = (r_{i+1}, x_{i+1})$ for $i = 0, 1, ..., i-1$

HW#3 Problem 8

最容易錯的點：
沒注意到題目有說要寫出 FST 的運作過程

HW#4 Problem 1

(Problem 1.43; 10 points) An *all*-NFA M is a 5-tuple $(Q, \Sigma, \delta, q, F)$ that accepts $x \in \Sigma^*$ if every possible state that M could be after reading input x is a state from F. Note, in contrast, that an ordinary NFA accepts a string if *some* state among these possible states is an accept state. Prove that all-NFAs recognize the class of regular languages.

HW#4 Problem 1

We need to prove the following two claims:

- All regular languages can be recognized by an *all*-NFA.
- All languages *all*-NFAs recognize are regular.

Claim: All regular languages can be recognized by an *all*-NFA.

Proof: All regular languages are recognized by a DFA, and DFA is also an *all*-NFA because DFA has only one run for each input string, namely, all the accepting runs (only one) terminate at the accepting states.

HW#4 Problem 1

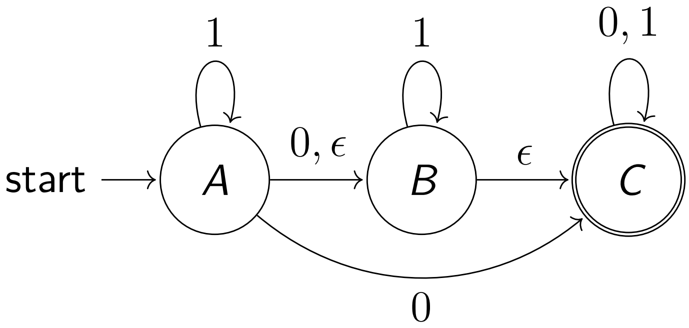
Claim: All languages *all*-NFAs recognize are regular.

Proof: Suppose that A is the language that an *all*-NFA $N = (Q, \Sigma, \delta, q, F)$ recognizes. Now we can construct a DFA $M = (Q', \Sigma, \delta', q', F')$ that recognizes A as follows:

- $Q' = P(Q)$ (the power set of Q).
- δ' is the ϵ-closure of transitions from the elements of the state-set.
- $q' = \{q\}$.
- $F' = P(F)$.

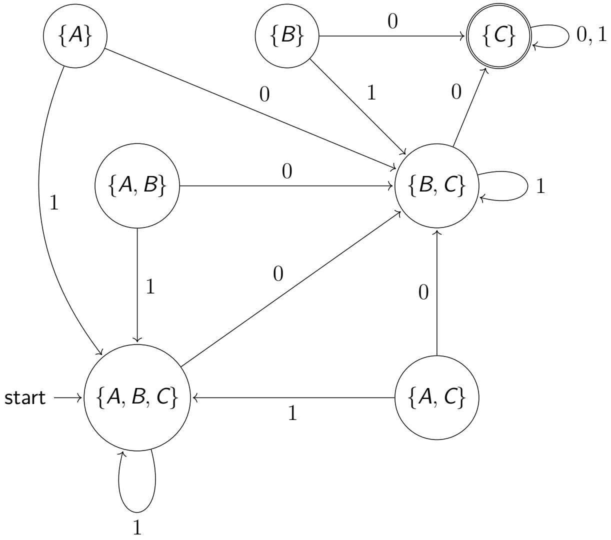
HW#4 Problem 1

For example: *all*-NFA N:



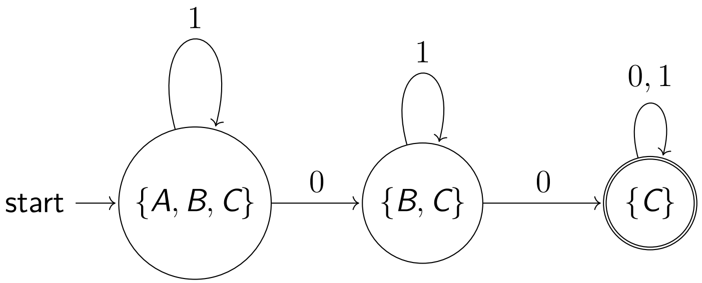
HW#4 Problem 1

For example: DFA M:



HW#4 Problem 1

Simplify M:



$$\begin{tikzcd}
\text{start} \arrow[r] & \{A,B,C\} \arrow[r, "0"] \arrow[loop above, "1"] & \{B,C\} \arrow[r, "0"] \arrow[loop above, "1"] & \{C\} \arrow[loop above, "0,1"]
\end{tikzcd}$$

HW#4 Problem 2

(Problem 1.32; 20 points) For languages A and B, let the *shuffle* of A and B be the language $\{w \mid w = a_1b_1 \cdots a_kb_k$, where $a_1 \cdots a_k \in A$ and $b_1 \cdots b_k \in B$, each $a_i, b_i \in \Sigma^*\}$. Show that the class of regular languages is closed under shuffle.

HW#4 Problem 2

Let $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$ and $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$ be two DFAs that recognize two regular languages A and B, respectively. Now we can construct a NFA $N = (Q, \{\Sigma \cup \{\epsilon\}\}, \delta, q, F)$ that recognizes the shuffle of A and B as follows:

- $Q = Q_A \times Q_B \times \{A, B\}$.
- $q = (q_A, q_B, A)$.
- $\delta((x, y, A), a) = \begin{cases} (x, y, B) & \text{if } a = \epsilon \\ (\delta_A(x, a), y, A) & \text{if } a \neq \epsilon \end{cases}$
- $\delta((x, y, B), a) = \begin{cases} (x, y, A) & \text{if } a = \epsilon \\ (x, \delta_B(y, a), B) & \text{if } a \neq \epsilon \end{cases}$
- $F = F_A \times F_B \times \{A\}$.

HW#4 Problem 3

(Problem 1.42; 20 points) Let $C_n = \{x \mid x \text{ is a binary number that is a multiple of } n\}$. Show that, for each $n \geq 1$, the language C_n is regular.

HW#4 Problem 3

Multiplication and addition of remainders satisfy the following rules:

- If $A \mod n = r$, $2A \mod n = (2r \mod n)$
- If $A \mod n = r$, $(A + p) \mod n = (r + p) \mod n$

The binary numbers satisfy the following rules:

Let x_2 be a binary number of a decimal number x_{10}:

- The value of $x_2 0 = 2 \times x_{10}$
- The value of $x_2 1 = 2 \times x_{10} + 1$

HW#4 Problem 3

So we can construct a DFA $D = (Q, \Sigma, \delta, q, F)$ for C_n as follows:

- $Q = \{q_s\} \cup \{q_i\}$ for $0 \leq i < n$,
- $\Sigma = \{0, 1\}$,
- $\delta(q_s, 0) = q_s$ and $\delta(q_s, 1) = q_{(1 \mod n)}$ (not always q_1 because $n = 1$ will reach q_0),
- $\delta(q_i, a) = \begin{cases} q_{(2i \mod n)} & \text{if } a = 0 \\ q_{((2i+1) \mod n)} & \text{if } a = 1 \end{cases}$
- $q = q_s$, and
- $F = \{q_0\}$.

HW#4 Problem 4

(Problem 1.66; 20 points) Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA and let h be a state of M called its "home". A *synchronizing sequence* for M and h is a string $s \in \Sigma^*$ where $\delta(q, s) = h$ for every $q \in Q$. Say that M is *synchronizable* if it has a synchronizing sequence for some state h. Prove that, if M is a k-state synchronizable DFA, then it has a synchronizing sequence of length at most k^3. (Note: $\delta(q, s)$ equals the state where M ends up, when M starts from state q and reads input s.)

HW#4 Problem 4

We first start from two states q_A and q_B of Q.

Let s_{AB} be a string that leads q_A and q_B into the same state g. Why?

The length of s_{AB} is at most $k * (k-1)$. Because the pairs of different two states in Q are at most $k * (k-1)$, if the length of s_{AB} is $k * (k-1) + 1$, there must be two repeated pairs, which means that the substring between them could be removed.

For example: if s_{AB} can be divided as $s_1 s_2 s_3$ such that

$$(q_A, q_B) \xrightarrow{s_1} (q'_A, q'_B) \xrightarrow{s_2} (q'_A, q'_B) \xrightarrow{s_3} (g, g)$$

Then s_2 can be removed.

HW#4 Problem 4

Now we have k states in Q. We can first run s_{AB} with the length at most $k * (k-1)$ so that q_A and q_B will transfer to the same state. Then, we can similarly run s_{BC} to make q_B and q_C transfer to the same state, which means that q_A, q_B and q_C are in the same state.

By repeating the steps above $k-1$ times, all k states will be transferred to the same state, which is h. And we can obtain our synchronizing sequence s with the length at most $k * (k-1)^2 \leq k^3$.

HW#4 Problem 5

5. (Problem 1.40; 20 points) Let

$$\Sigma_2 = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}.$$

Here, Σ_2 contains all columns of 0s and 1s of length two. A string of symbols in Σ_2 gives two rows of 0s and 1s.

Consider the top and bottom rows to be strings of 0s and 1s and let

$$E = \{w \in \Sigma_2^* \mid \text{the bottom row of } w \text{ is the reverse of the top row of } w\}.$$

Show that E is not regular.

HW#4 Problem 5

Use the pumping lemma:

Let s be $\begin{bmatrix}0\\1\end{bmatrix}^p\begin{bmatrix}1\\0\end{bmatrix}^p$, where p is the pumping length for E.

When dividing s as xyz, because $|xy| < p$, y must consist of $\begin{bmatrix}0\\1\end{bmatrix}$s.

And obviously, $xy^2z \notin E$ (the number of 0 is different between the top and the bottom rows).

HW#4 Problem 6

(Problem 1.51; 10 points) Prove that the language $\{w \in \{0,1\}^* \mid w \text{ is not a palindrome}\}$ is not regular. You may use the pumping lemma and the closedness of the class of regular languages under union, intersection, and complement. (Note: a *palindrome* is a string that reads the same forward and backward.)

HW#4 Problem 6

Let $\bar{A} = \{w \in \{0, 1\}* \mid w \text{ is not a palindrome}\}$.

Because the class of regular languages is closed under complement, if A is regular, $\bar{A}$ must be regular. On the other hand, if A is not regular, $\bar{A}$ must not be regular.

HW#4 Problem 6

Prove that $A = \{w \in \{0,1\}* \mid w \text{ is a palindrome}\}$ is not regular.

Use the pumping lemma:

Let s be 0^p10^p, where p is the pumping length for A.
When dividing s as xyz, because $|xy| < p$, y must consist of 0s.
And obviously, $xy^2z \notin A$ (the number of 0 is different on both sides of 1).

HW#5 Problem 1

(Exercise 2.1; 20 points) Consider the following CFG discussed in class, where for convenience the variables have been renamed with single letters.

$$\begin{array}{rcl} E & \to & E + T \mid T \\ T & \to & T \times F \mid F \\ F & \to & (E) \mid a \end{array}$$

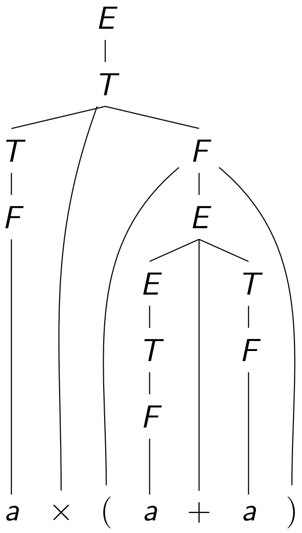
Give (leftmost) derivations and the corresponding parse trees for the following strings.

(a) $a \times (a + a)$

(b) $((a) + a)$

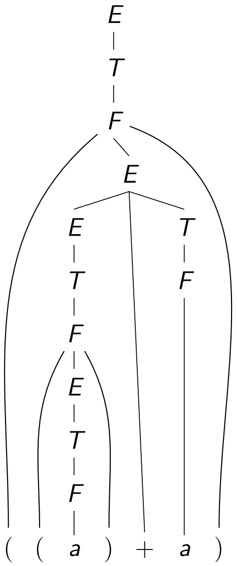
HW#5 Problem 1 (a)

$a \times (a + a)$



HW#5 Problem 1 (b)

$((a) + a)$



HW#5 Problem 2

(Exercise 2.4; 20 points) Give context-free grammars that generate the following languages. In all parts the alphabet Σ is $\{0, 1\}$.

(a) $\{w \mid \text{the length of } w \text{ is odd}\}$

(b) $\{w \mid w = w^R, \text{ that is, } w \text{ is a palindrome}\}$

HW#5 Problem 2 (a)

$\{\omega \mid$ the length of ω is odd $\}$

$S \rightarrow CB$
$B \rightarrow CCB \mid \epsilon$
$C \rightarrow 0 \mid 1$

HW#5 Problem 2 (b)

$\{\omega \mid \omega = \omega^R$, that is, ω is palindrome $\}$

$S \to 0S0 \mid 1S1 \mid C \mid \epsilon$
$C \to 0 \mid 1$

HW#5 Problem 3

(Exercise 2.6d; 10 points) Give a context-free grammar that generates the language $\{x_1\#x_2\#\cdots\#x_k \mid k \geq 1,\ \text{each } x_i \in \{a,b\}^*,\ \text{and for some } i \text{ and } j,\ x_i = x_j^R\}$.

HW#5 Problem 3

The pattern of generated string can be considered as the following:

$L\ x_i\ M\ x_j\ R$, where $x_i = x_j^R$.

Let $X = \{a, b\}^*$.
L can generate:

1. ϵ
2. $\cdots X\#X\#X\#$

R can generate:

1. ϵ
2. $\#X\#X\#X\cdots$

HW#5 Problem 3

$L\ x_i\ M\ x_j\ R$

M can generate:

1. $\#X\#X\#\cdots\#X\#X\#$
2. $\#$
3. ϵ, a, b (when $i = j$, namely $x_i = x_j$ is a palindrome)

Remember: x_i and x_j can also be ϵ because $\epsilon^R = \epsilon$.

HW#5 Problem 3

$$
\begin{aligned}
&S \to LM'R \\
&M' \to aM'a \mid bM'b \mid M \\
&M \to \#XMX\# \mid \# \mid a \mid b \mid \epsilon \\
&L \to X\#L \mid \epsilon \\
&R \to R\#X \mid \epsilon \\
&X \to Xa \mid Xb \mid \epsilon
\end{aligned}
$$

HW#5 Problem 4

4. (Exercise 2.8; 10 points) Show that the string "**`the boy likes the girl with a flower`**" has two different leftmost derivations in the following CFG.

$$
\begin{array}{rcl}
\langle\text{SENTENCE}\rangle & \to & \langle\text{NOUN-PHRASE}\rangle\langle\text{VERB-PHRASE}\rangle \\
\langle\text{NOUN-PHRASE}\rangle & \to & \langle\text{CMPLX-NOUN}\rangle \mid \\
 & & \langle\text{CMPLX-NOUN}\rangle\langle\text{PREP-PHRASE}\rangle \\
\langle\text{VERB-PHRASE}\rangle & \to & \langle\text{CMPLX-VERB}\rangle \mid \\
 & & \langle\text{CMPLX-VERB}\rangle\langle\text{PREP-PHRASE}\rangle \\
\langle\text{PREP-PHRASE}\rangle & \to & \langle\text{PREP}\rangle\langle\text{CMPLX-NOUN}\rangle \\
\langle\text{CMPLX-NOUN}\rangle & \to & \langle\text{ARTICLE}\rangle\langle\text{NOUN}\rangle \\
\langle\text{CMPLX-VERB}\rangle & \to & \langle\text{VERB}\rangle \mid \langle\text{VERB}\rangle\langle\text{NOUN-PHRASE}\rangle \\
\langle\text{ARTICLE}\rangle & \to & \texttt{a} \mid \texttt{the} \\
\langle\text{NOUN}\rangle & \to & \texttt{boy} \mid \texttt{girl} \mid \texttt{flower} \\
\langle\text{VERB}\rangle & \to & \texttt{touches} \mid \texttt{likes} \mid \texttt{sees} \\
\langle\text{PREP}\rangle & \to & \texttt{with}
\end{array}
$$

HW#5 Problem 4

$\mathrm{S} \Rightarrow \mathrm{NP}\,\mathrm{VP} \Rightarrow \mathrm{CN}\,\mathrm{VP} \Rightarrow \mathrm{A}\,\mathrm{N}\,\mathrm{VP} \Rightarrow$ the $\mathrm{N}\,\mathrm{VP} \Rightarrow$ the boy $\mathrm{VP} \Rightarrow$
the boy $\mathrm{CV} \Rightarrow$ the boy $\mathrm{V}\,\mathrm{NP} \Rightarrow$ the boy likes $\mathrm{NP} \Rightarrow$
the boy likes $\mathrm{CN}\,\mathrm{PP} \Rightarrow$ the boy likes $\mathrm{A}\,\mathrm{N}\,\mathrm{PP} \Rightarrow$
the boy likes the $\mathrm{N}\,\mathrm{PP} \Rightarrow$ the boy likes the girl $\mathrm{PP} \Rightarrow$
the boy likes the girl $\mathrm{P}\,\mathrm{CN} \Rightarrow$ the boy likes the girl with $\mathrm{CN} \Rightarrow$
the boy likes the girl with $\mathrm{A}\,\mathrm{N} \Rightarrow$ the boy likes the girl with a $\mathrm{N} \Rightarrow$
the boy likes the girl with a flower

HW#5 Problem 4

$\mathrm{S} \Rightarrow \mathrm{NP\ VP} \Rightarrow \mathrm{CN\ VP} \Rightarrow \mathrm{A\ N\ VP} \Rightarrow$ the $\mathrm{N\ VP} \Rightarrow$ the boy $\mathrm{VP} \Rightarrow$
the boy $\mathrm{CV\ PP} \Rightarrow$ the boy $\mathrm{V\ NP\ PP} \Rightarrow$ the boy likes $\mathrm{NP\ PP} \Rightarrow$
the boy likes $\mathrm{CN\ PP} \Rightarrow$ the boy likes $\mathrm{A\ N\ PP} \Rightarrow$
the boy likes the $\mathrm{N\ PP} \Rightarrow$ the boy likes the girl $\mathrm{PP} \Rightarrow$
the boy likes the girl $\mathrm{P\ CN} \Rightarrow$ the boy likes the girl with $\mathrm{CN} \Rightarrow$
the boy likes the girl with $\mathrm{A\ N} \Rightarrow$ the boy likes the girl with a $\mathrm{N} \Rightarrow$
the boy likes the girl with a flower

HW#5 Problem 5

5. (Exercise 2.9; 20 points) Give a context-free grammar that generates the language

$$A = \{a^i b^j c^k \mid i = j \text{ or } j = k \text{ where } i, j, k \geq 0\}.$$

Is your grammar ambiguous? Why or why not?

HW#5 Problem 5

設計一個 CFG 生成 $a^i b^j c^k$ 其中 $i = j \vee j = k$
我們可以走兩條路線：$i = j$ 路線或 $j = k$ 路線
對 $i = j$ 路線而言，左半邊要有等量生成的 a 與 b，右邊的 c 則是任意生成
$j = k$ 路線也是以此類推

HW#5 Problem 5

$S \to UC \mid AV$
$U \to aUb \mid \epsilon$
$V \to bVc \mid \epsilon$
$A \to aA \mid \epsilon$
$C \to cC \mid \epsilon$

解釋這個 CFG 是否為 ambiguous
考慮字串 abc，可以有兩條路線：
$S \Rightarrow UC \Rightarrow aUbC \Rightarrow abC \Rightarrow abcC \Rightarrow abc$
$S \Rightarrow AV \Rightarrow aAV \Rightarrow aV \Rightarrow abVc \Rightarrow abc$

HW#5 Problem 6

6. (Exercise 2.14; 20 points) Convert the following CFG (where A is the start variable) into an equivalent CFG in Chomsky normal form, using the procedure given in Theorem 2.9.

$$\begin{array}{lcl} A & \rightarrow & BAB \mid B \mid \varepsilon \\ B & \rightarrow & 0B1 \mid \varepsilon \end{array}$$

HW#5 Problem 6

$$A \to BAB \mid B \mid \epsilon$$
$$B \to 0B1 \mid \epsilon$$

HW#5 Problem 6

第一程序：增加新的 start symbol

加上 $S_0 \rightarrow A$

$S_0 \rightarrow A$

$A \rightarrow BAB \mid B \mid \epsilon$

$B \rightarrow 0B1 \mid \epsilon$

HW#5 Problem 6

第二程序：去除 ϵ rule

去除 $B \to \epsilon$

$S_0 \to A$

$A \to BAB \mid B \mid \epsilon \mid BA \mid AB \mid A$

$B \to 0B1$

HW#5 Problem 6

第二程序：去除 ϵ rule
去除 $A \to \epsilon$
$S_0 \to A \mid \epsilon$
$A \to BAB \mid B \mid BA \mid AB \mid A \mid BB$
$B \to 0B1$

HW#5 Problem 6

第三程序：去除 unit rule

去除 $A \to A$

$S_0 \to A \mid \epsilon$

$A \to BAB \mid B \mid BA \mid AB \mid BB$

$B \to 0B1$

HW#5 Problem 6

第三程序：去除 unit rule

去除 $A \to B$

$S_0 \to A \mid \epsilon$

$A \to BAB \mid BA \mid AB \mid BB \mid 0B1$

$B \to 0B1$

HW#5 Problem 6

第三程序：去除 unit rule

去除 $S \to A$

$S_0 \to BAB \mid BA \mid AB \mid BB \mid 0B1 \mid \epsilon$

$A \to BAB \mid BA \mid AB \mid BB \mid 0B1$

$B \to 0B1$

HW#5 Problem 6

第四程序：分割其它 rule

去除 $S_0 \to BAB$ 與 $A \to BAB$

$S_0 \to BC_1 \mid BA \mid AB \mid BB \mid 0B1 \mid \epsilon$

$A \to BC_2 \mid BA \mid AB \mid BB \mid 0B1$

$B \to 0B1 \quad C_1 \to AB$

$C_2 \to AB$

HW#5 Problem 6

第四程序：分割其它 rule

去除 $S \rightarrow 0B1$、$A \rightarrow 0B1$ 與 $B \rightarrow 0B1$

$S_0 \rightarrow BC_1 \mid BA \mid AB \mid BB \mid C_3 1 \mid \epsilon$

$A \rightarrow BC_2 \mid BA \mid AB \mid BB \mid C_4 1$

$B \rightarrow C_5 1 \quad C_1 \rightarrow AB$

$C_2 \rightarrow AB \quad C_3 \rightarrow 0B$

$C_4 \rightarrow 0B$

$C_5 \rightarrow 0B$

HW#5 Problem 6

$$\begin{aligned}
&S_0 \to BC_1 \mid BA \mid AB \mid BB \mid C_3 I_1 \mid \epsilon \\
&A \to BC_2 \mid BA \mid AB \mid BB \mid C_4 I_2 \\
&B \to C_5 I_3 \quad C_1 \to AB \\
&C_2 \to AB \quad\quad C_3 \to O_1 B \\
&C_4 \to O_2 B \\
&C_5 \to O_3 B \\
&I_1 \to 1 \\
&I_2 \to 1 \\
&I_3 \to 1 \\
&O_1 \to 0 \\
&O_2 \to 0 \\
&O_3 \to 0
\end{aligned}$$