Homework 1 - 5

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 $\mathcal{R}$:

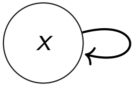
$x \mathcal{R} y$:

$$\begin{tikzcd} x \arrow[r] & y \end{tikzcd}$$

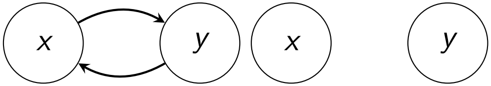
HW#1 Problem 1

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

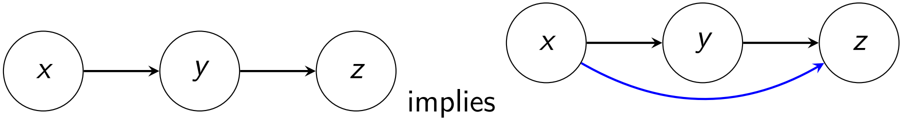
Reflexive: $\forall x \in S,\ x \mathcal{R} x$.



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

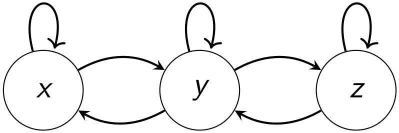


Transitive: $\forall x, y, z \in S,\ x \mathcal{R} y$ and $y \mathcal{R} z$ implies $x \mathcal{R} z$.



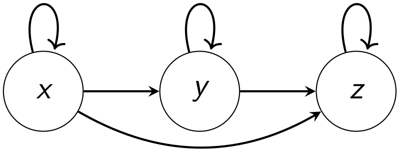
HW#1 Problem 1

(a) Reflexive and symmetric but not transitive



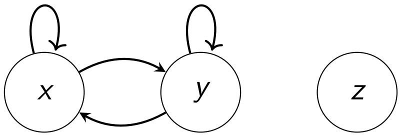
HW#1 Problem 1

(b) Reflexive and transitive but not symmetric

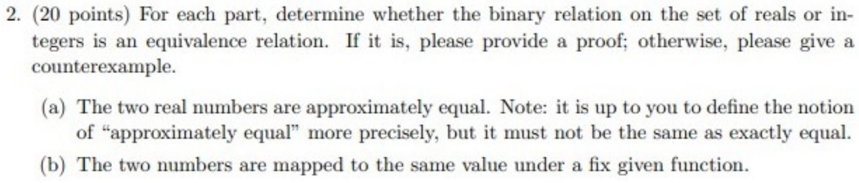


HW#1 Problem 1

(c) Symmetric and transitive but not reflexive



HW#1 Problem 2



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 real numbers are approximately equal. Note: it is up to you to define the notion of "approximately equal" more precisely, but it must not be the same as exactly equal.
   (b) The two numbers are mapped to the same value under a fix given function.

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 real numbers are approximately equal.
Suppose we define that two real numbers x and y are approximately equal if $|x - y| \leq 0.1$

- Reflexive: satisfied, $\forall x \in \mathbb{R}$, $|x - x| = 0 \leq 0.1$
- Symmetric: satisfied, $\forall x, y \in \mathbb{R}$, if $|x - y| \leq 0.1$, then $|y - x| = |x - y| \leq 0.1$
- Transitive: violated, counterexample: $|0.1 - 0.2| \leq 0.1$, $|0.2 - 0.3| \leq 0.1$, but $|0.1 - 0.3| = 0.2 > 0.1$

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

HW#1 Problem 2

(b) $\mathcal{R}$: The two numbers are mapped to the same value under a fix given function.

- Reflexive: satisfied, the same input always produces the same output.
- Symmetric: satisfied, if $f(x) = a = f(y)$ then $f(y) = a = f(x)$.
- Transitive: satisfied, if $f(x) = a = f(y)$ and $f(y) = b = f(z)$, we can prove by contradiction that $a = b$. Therefore $f(x) = f(z)$.

So, $\mathcal{R}$ 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 the 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 element of a function is a tuple, and the first element of the tuple is also a k-tuple.
- For all $a \in (A_1 \times \cdots \times A_k)$, exists $b \in B$ such that $(a, b) \in f$.
- For all the first elements t of the tuples in f, $[(t, p), (t, q) \in f]$ implies $p = q$.
- A function with a *k-ary* relation as its first element of the tuple 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

(Problem 0.10; 20 points) Show that every graph having two or more nodes contains two nodes with the same degree. (Note: we assume that every graph is simple and finite, unless explicitly stated otherwise.)

HW#1 Problem 4

Proved by contradiction.

Supposed that there is a graph with n nodes 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

(Problem 0.11; 10 points) Find the error in the following proof that all horses are the same color.

CLAIM: In any set of h horses, all horses are the same color.

PROOF: By induction on h.

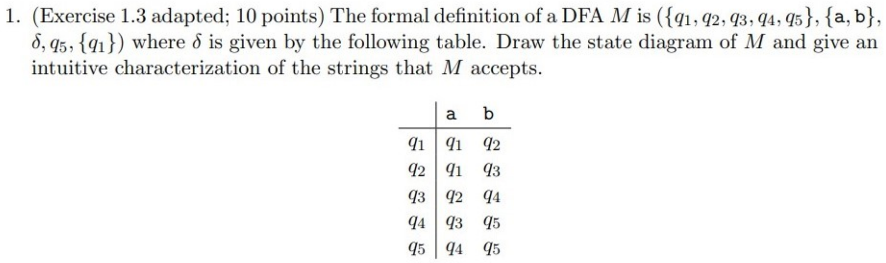
Basis ($h = 1$): In any set containing just one horse, all horses clearly are the same color.

Induction step ($h > 1$): We assume that the claim is true for $h = k$ ($k \geq 1$) and prove that it is true for $h = k + 1$. Take any set H of $k + 1$ horses. We show that all the horses in this set are the same color. Remove one horse from this set to obtain the set H_1 with just k horses. By the induction hypothesis, all the horses in H_1 are the same color. Now replace the removed horse and remove a different one to obtain the set H_2. By the same argument, all the horses in H_2 are the same color. Therefore all the horses in H must be the same color, and the proof is complete.

HW#1 Problem 5

The error occurs in the last sentence(it's right only if H_1 and H_2 have a horse in common). Consider the case that H contains exactly 2 horses, then H_1 and H_2 each has exactly 1 horse, but do not have a horse in common. Although all the horses in H_1 are the same color and so are those in H_2, we cannot conclude that the horse in H_1 has the same color as the horse in H_2. So, the 2 horses in H may not be colored the same.

HW#2 Problem 1



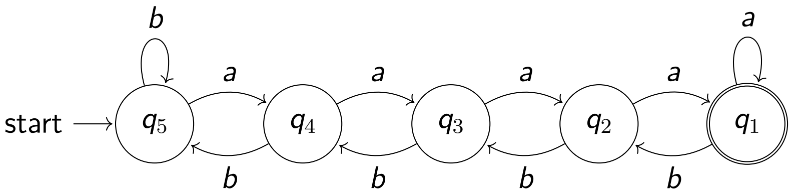
1. (Exercise 1.3 adapted; 10 points) The formal definition of a DFA M is $(\{q_1, q_2, q_3, q_4, q_5\}, \{\texttt{a}, \texttt{b}\}, \delta, q_5, \{q_1\})$ where δ is given by the following table. Draw the state diagram of M and give an intuitive characterization of the strings that M accepts.

	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

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

HW#2 Problem 1

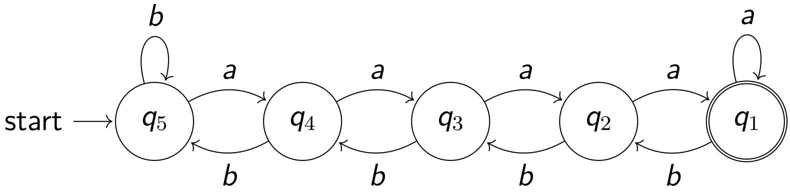


Intuitive characterization of the strings that M accepts:

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:

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

HW#2 Problem 1



e.q.

bbbbbbbaaaaba: we can find a suffix $X = aaaaba$ such that:

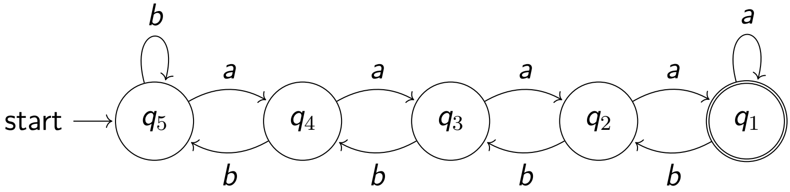
- $n_a = 5 = 1 + 4 = n_b + 4$, and
- for all suffixes of X: a, ba, aba, $aaba$, $aaaba$, $aaaaba$, $n_a \geq n_b$.

HW#2 Problem 1

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

What if *bbbbbbbaaaaab*? We can only find one suffix $X = aaaaab$ such that $n_a = n_b + 4$, but the suffix *b* in X does not satisfy $n_a \geq n_b$.
What if *bbbbbaaaaabba*? We can only find one suffix $X = aaaaabba$ such that $n_a = n_b + 4$, but the suffix *bba* in X does not satisfy $n_a \geq n_b$.
What if *bbbbbbaaaabba*? We can not find any suffixes such that $n_a = n_b + 4$.

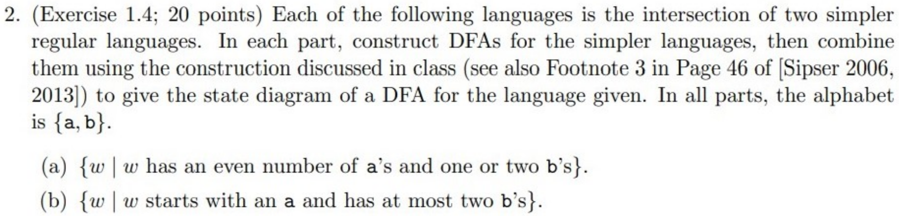
HW#2 Problem 1



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

So the first condition [$n_a = n_b + 4$ in X] guarantees the input string ends in state q_1, and the second condition [for all suffixes of X, $n_a \geq n_b$] guarantees that even if it leaves from q_5, it will come back eventually.

HW#2 Problem 2



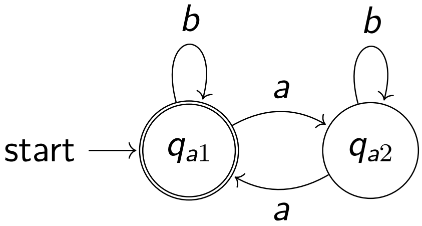
2. (Exercise 1.4; 20 points) Each of the following languages is the intersection of two simpler regular languages. In each part, construct DFAs for the simpler languages, then combine them using the construction discussed in class (see also Footnote 3 in Page 46 of [Sipser 2006, 2013]) to give the state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

   (a) $\{w \mid w$ has an even number of a's and one or two b's$\}$.

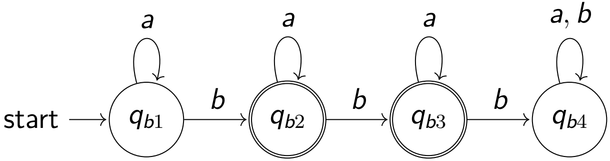
   (b) $\{w \mid w$ starts with an a and has at most two b's$\}$.

HW#2 Problem 2 (a)

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

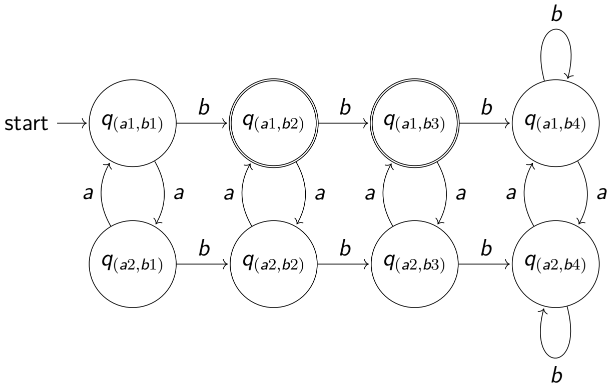


Simpler language: $\{w \mid w \text{ has one or two } b\text{'s}\}$.



HW#2 Problem 2 (a)

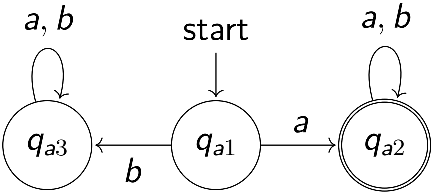
Language: $\{w \mid w \text{ has an even number of } a\text{'s and has one or two } b\text{'s}\}$.



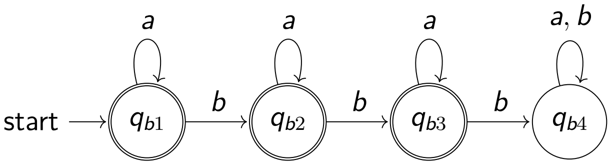
$$\begin{tikzcd}
\text{start} \arrow[r] & q_{(a1,b1)} \arrow[r, "b"] \arrow[d, bend left, "a"] & q_{(a1,b2)} \arrow[r, "b"] \arrow[d, bend left, "a"] & q_{(a1,b3)} \arrow[r, "b"] \arrow[d, bend left, "a"] & q_{(a1,b4)} \arrow[loop above, "b"] \arrow[d, bend left, "a"] \\
& q_{(a2,b1)} \arrow[r, "b"] \arrow[u, bend left, "a"] & q_{(a2,b2)} \arrow[r, "b"] \arrow[u, bend left, "a"] & q_{(a2,b3)} \arrow[r, "b"] \arrow[u, bend left, "a"] & q_{(a2,b4)} \arrow[loop below, "b"] \arrow[u, bend left, "a"]
\end{tikzcd}$$

HW#2 Problem 2 (b)

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

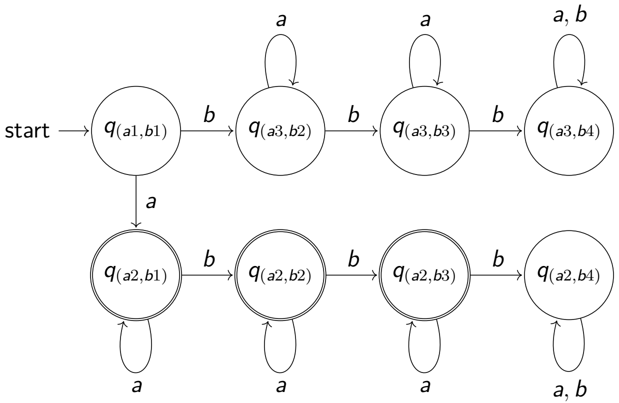


Simpler language: $\{w \mid w \text{ has at most two } b\text{'s}\}$.

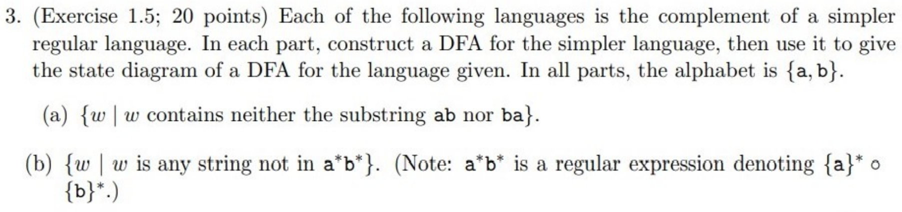


HW#2 Problem 2 (b)

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



HW#2 Problem 3



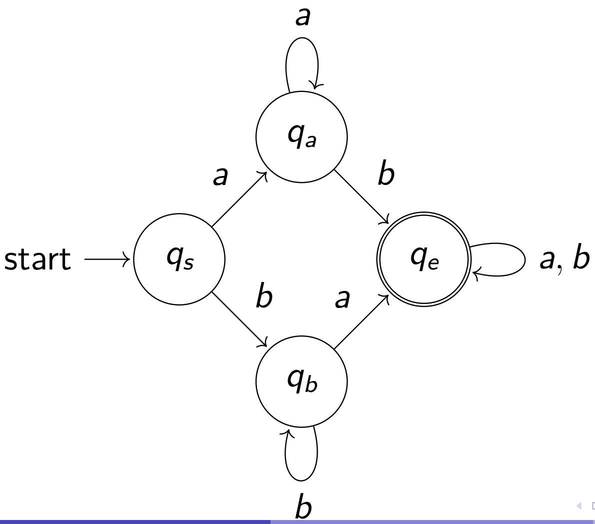
3. (Exercise 1.5; 20 points) Each of the following languages is the complement of a simpler regular language. In each part, construct a DFA for the simpler language, then use it to give the state diagram of a DFA for the language given. In all parts, the alphabet is $\{\mathtt{a}, \mathtt{b}\}$.

   (a) $\{w \mid w$ contains neither the substring `ab` nor `ba`$\}$.

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

HW#2 Problem 3 (a)

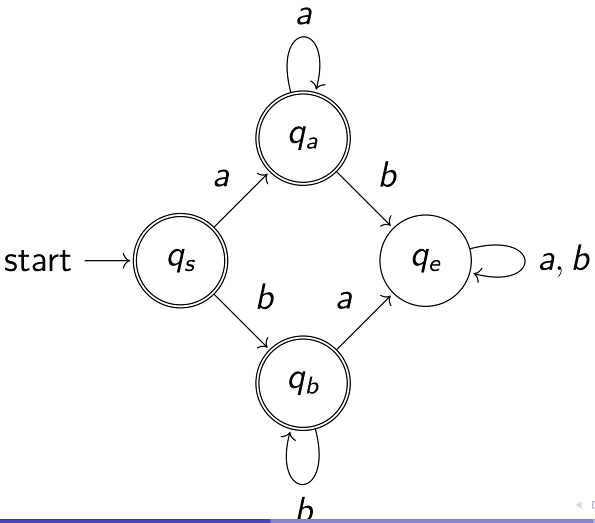
Simpler language:
$\{w \mid w$ contains the substring ab or $ba\}$.



HW#2 Problem 3 (a)

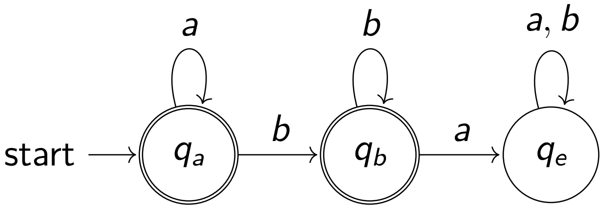
Language:

$\{w \mid w$ contains neither the substring ab or $ba\}$.



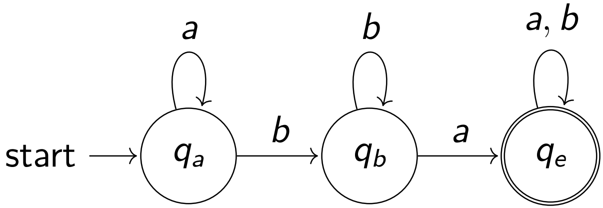
HW#2 Problem 3 (b)

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


$$\begin{tikzcd}
\text{start} \arrow[r] & q_a \arrow[r, "b"] \arrow[loop above, "a"] & q_b \arrow[r, "a"] \arrow[loop above, "b"] & q_e \arrow[loop above, "{a, b}"]
\end{tikzcd}$$

HW#2 Problem 3 (b)

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



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

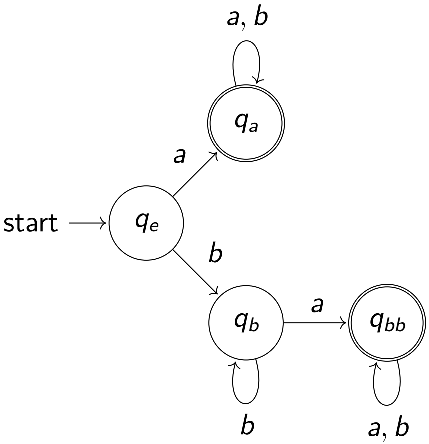
Let DFA M recognizes the language A, and we can construct a NFA M^R which recognizes A^R according to the following:

- M^R conserves all states from M and M^R's alphabet is as same as M.
- Reverse all the transitions of M as the transitions of M^R. e.q. $\delta(q_1, a) = q_2 \to \delta(q_2, a) = q_1$.
- 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.

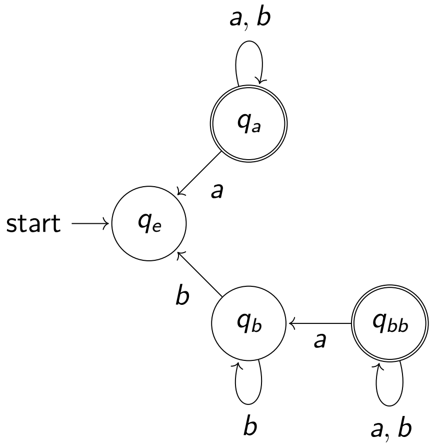
HW#2 Problem 4

e.q. : M



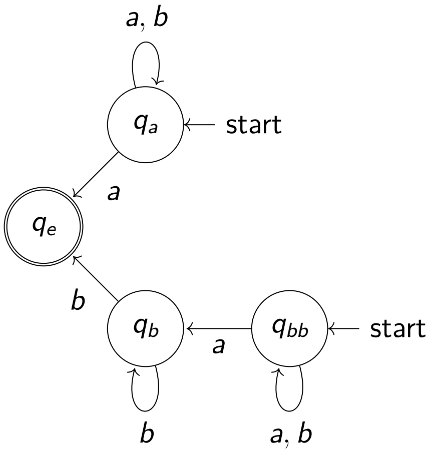
HW#2 Problem 4

Reverse all the transitions:



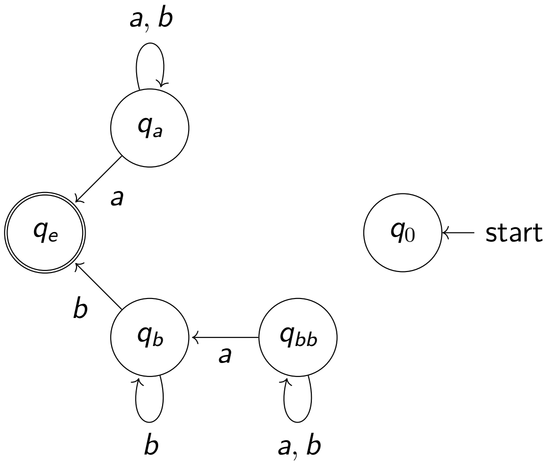
HW#2 Problem 4

Change the initial state into accepting state:



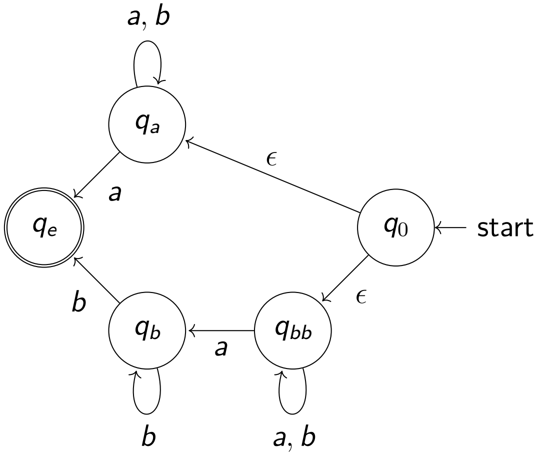
HW#2 Problem 4

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

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

Σ_3 contains all size 3 columns of 0s and 1s. A string of symbols in Σ_3 gives three rows of 0s and 1s. 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,

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

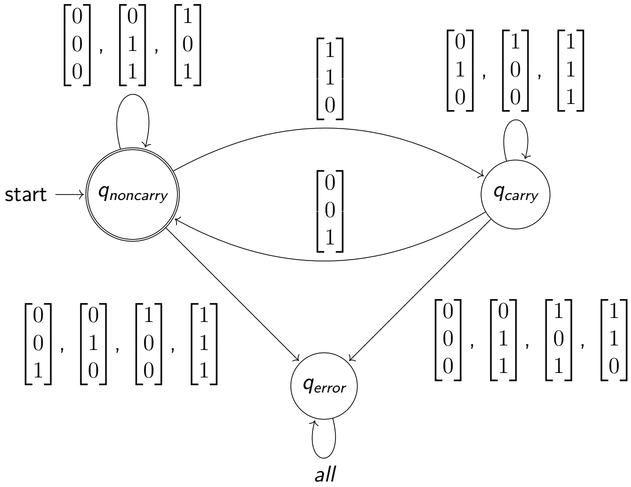
Show that B is regular. (Hint: working with B^R is easier. You may assume the result claimed in the previous problem (Problem 1.36).)

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:



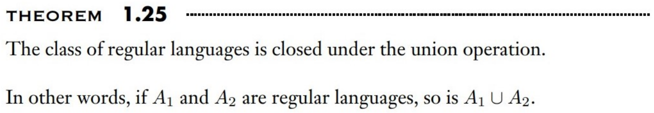
$\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}$
$\begin{bmatrix}1\\1\\0\end{bmatrix}$
$\begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix}$
start
$q_{noncarry}$
$\begin{bmatrix}0\\0\\1\end{bmatrix}$
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.


THEOREM 1.25

The class of regular languages is closed under the union operation.

In other words, if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$.

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 $A_1 \cup A_2$:

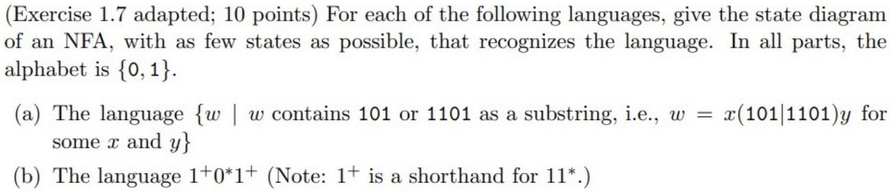
- $Q = \{(Q_1 \cup \{q_f\}) \times (Q_2 \cup \{q_f\})\}$.
- $\Sigma = \Sigma_1 \cup \Sigma_2$.
- $\delta((r_1, r_2), a) =$
$$\begin{cases} (\delta_1(r_1,a), \delta_2(r_2,a)) & \text{if } (r_1, r_2 \neq q_f) \text{ and } (a \in (\Sigma_1 \cap \Sigma_2)) \\ (\delta_1(r_1,a), q_f) & \text{if } (r_1 \neq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

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



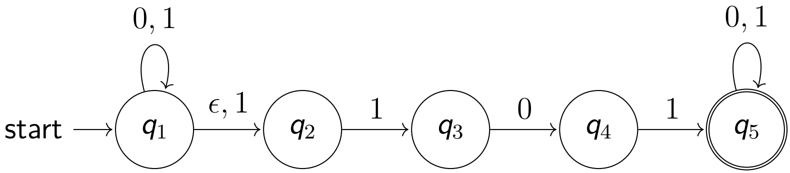
(Exercise 1.7 adapted; 10 points) For each of the following languages, give the state diagram of an NFA, with as few states as possible, that recognizes the language. In all parts, the alphabet is $\{\mathtt{0},\mathtt{1}\}$.

(a) The language $\{w \mid w$ contains `101` or `1101` as a substring, i.e., $w = x(\mathtt{101}|\mathtt{1101})y$ for some x and $y\}$

(b) The language $1^+0^*1^+$ (Note: 1^+ is a shorthand for 11^*.)

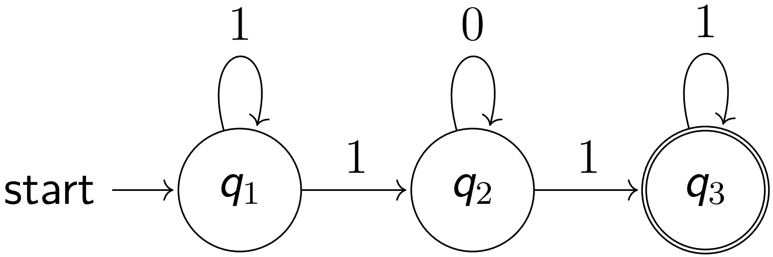
HW#3 Problem 1

(a) The language $\{\omega \mid \omega$ contains 101 or 1101 as a substring, i.e., $\omega = x(101|1101)y$ for some x and $y\}$ with five states.



HW#3 Problem 1

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

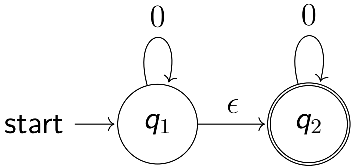


HW#3 Problem 2

(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

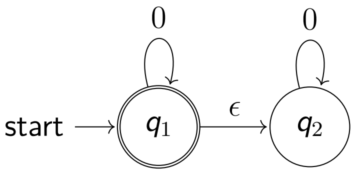
Give a example that swapping the accept and nonaccept states in an NFA does not necessarily yield a new NFA that recognizes the complement of the original language:


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

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:



The above NFA still recognizes the string 0^*.

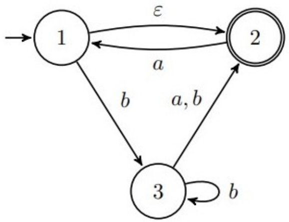
HW#3 Problem 2

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 adapted; 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.

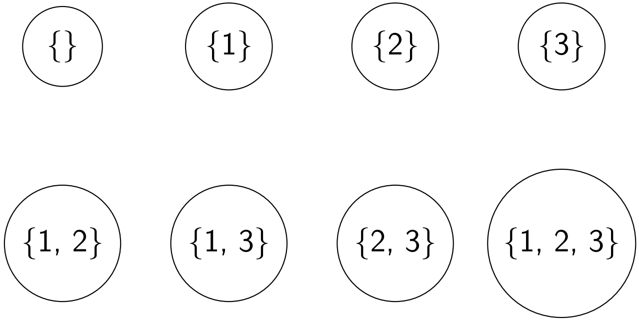


HW#3 Problem 3

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

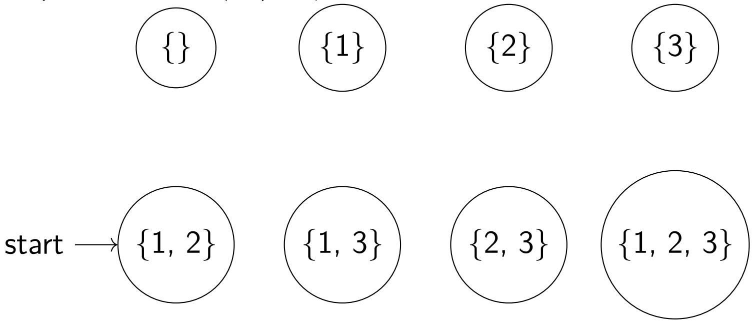
HW#3 Problem 3

把每一個 state 都列出來



HW#3 Problem 3

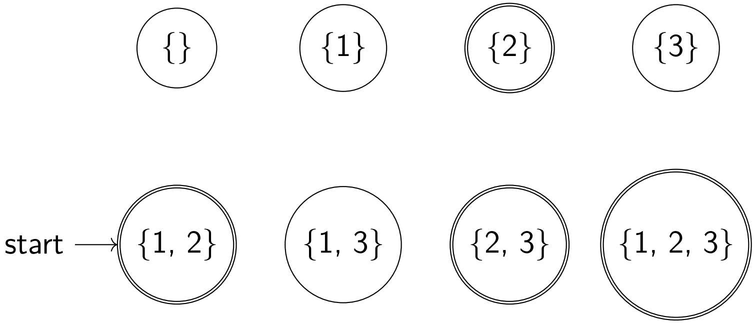
把每一個 state 都列出來



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

HW#3 Problem 3

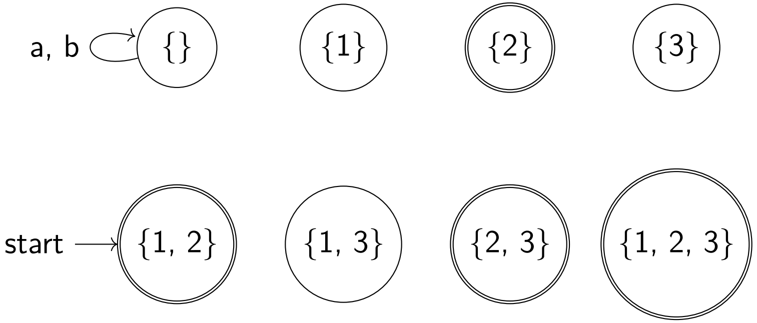
把每一個 state 都列出來



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

HW#3 Problem 3

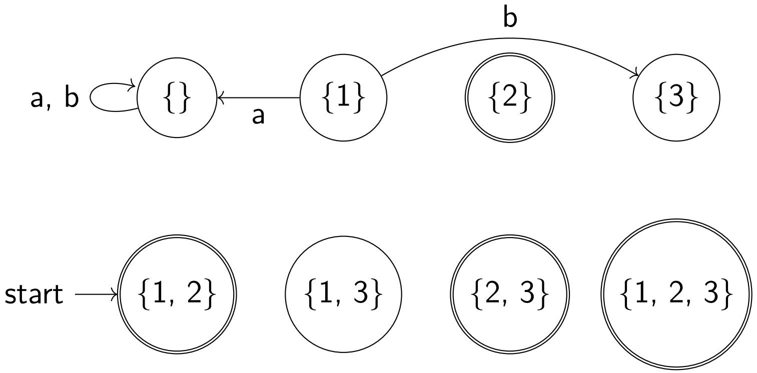
把每一個 state 都列出來



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

HW#3 Problem 3

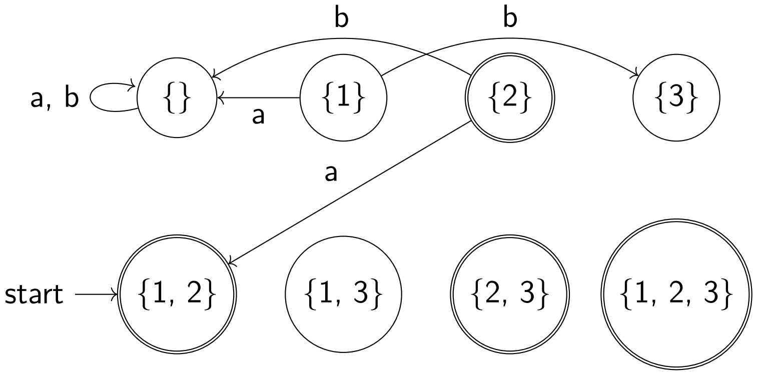
把每一個 state 都列出來



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

HW#3 Problem 3

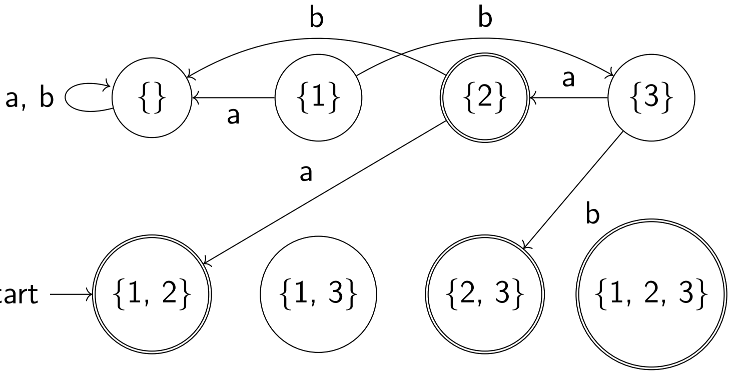
把每一個 state 都列出來



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

HW#3 Problem 3

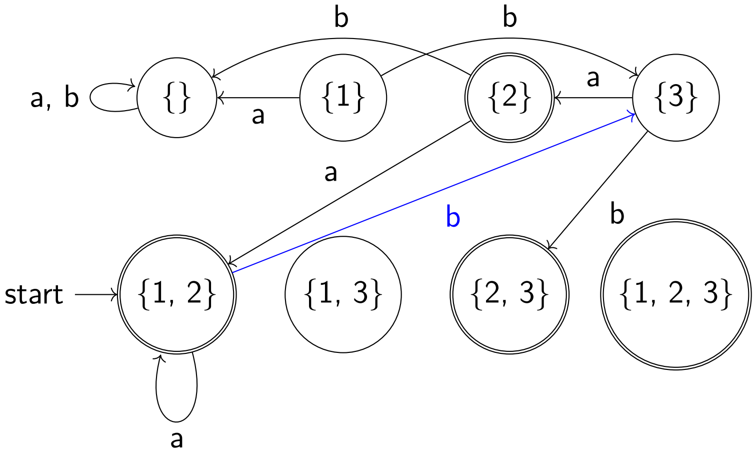
把每一個 state 都列出來



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

HW#3 Problem 3

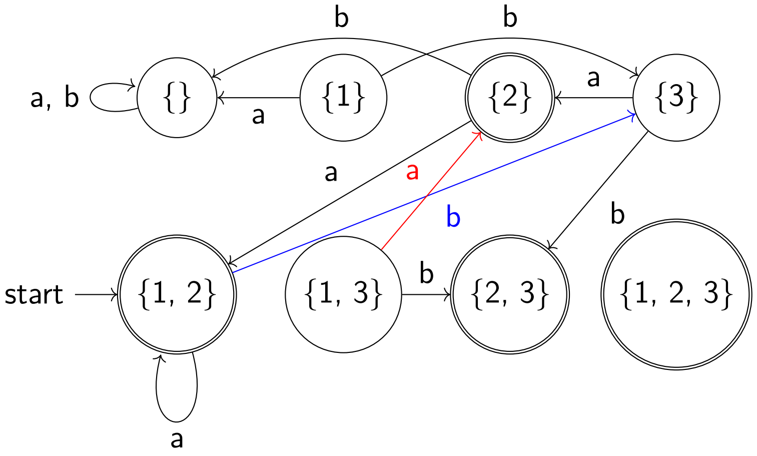
把每一個 state 都列出來



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

HW#3 Problem 3

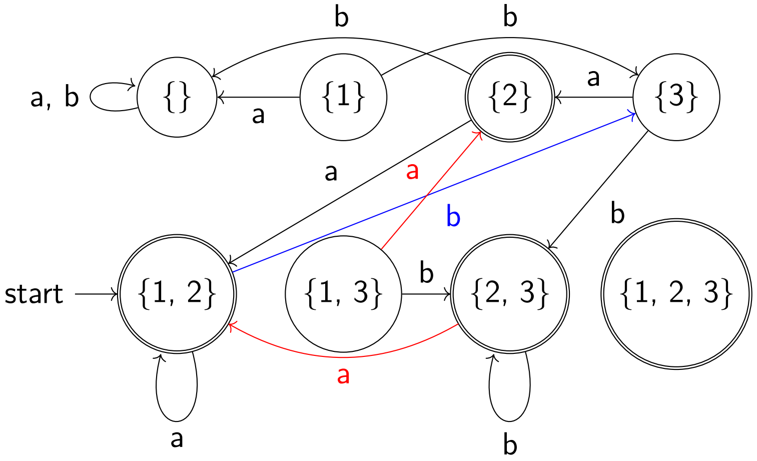
把每一個 state 都列出來



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

HW#3 Problem 3

把每一個 state 都列出來



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

HW#3 Problem 3

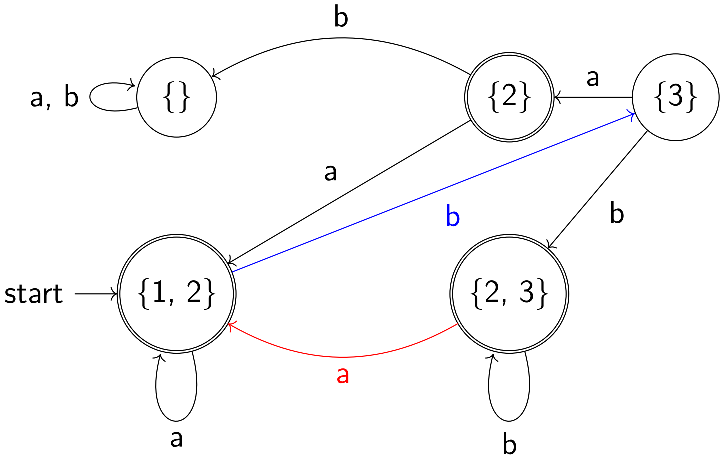
把每一個 state 都列出來

$$\begin{tikzcd}
 & \{\} \arrow["{a, b}"', loop, in=215, out=145, distance=2em] & \{1\} \arrow[l, "a"'] \arrow[rr, bend left, "b"] & \{2\} \arrow[ll, bend right, "b"'] \arrow[dll, "a"'] & \{3\} \arrow[l, "a"'] \arrow[dl, "b"] \\
\text{start} \arrow[r] & \{1, 2\} \arrow["a", loop, below] \arrow[urrr, "b", color=blue] & \{1, 3\} \arrow[ur, "a", color=red] \arrow[r, "b"] & \{2, 3\} \arrow[ll, bend left, "a", color=red] \arrow["b", loop, below] & \{1, 2, 3\} \arrow[l, "b"] \arrow[lll, bend left=40, "a", color=blue]
\end{tikzcd}$$

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

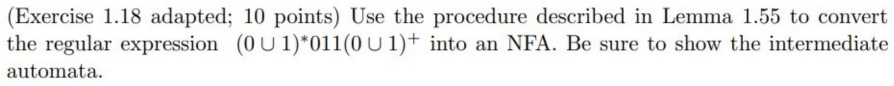
HW#3 Problem 3

把每一個 state 都列出來



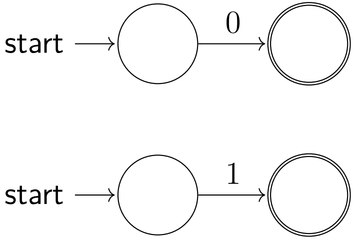
刪去無法到達的狀態
{1}，{1, 3} 和 {1, 2, 3} 這三個節點一定無法到達

HW#3 Problem 4


(Exercise 1.18 adapted; 10 points) Use the procedure described in Lemma 1.55 to convert the regular expression $(0 \cup 1)^*011(0 \cup 1)^+$ into an NFA. Be sure to show the intermediate automata.

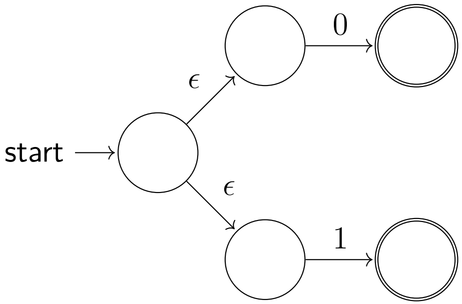
HW#3 Problem 4

$0 \qquad 1$



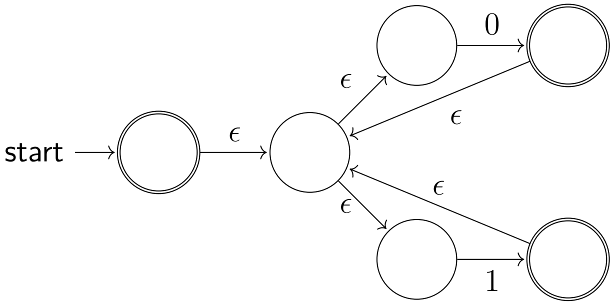
HW#3 Problem 4

$0 \cup 1$



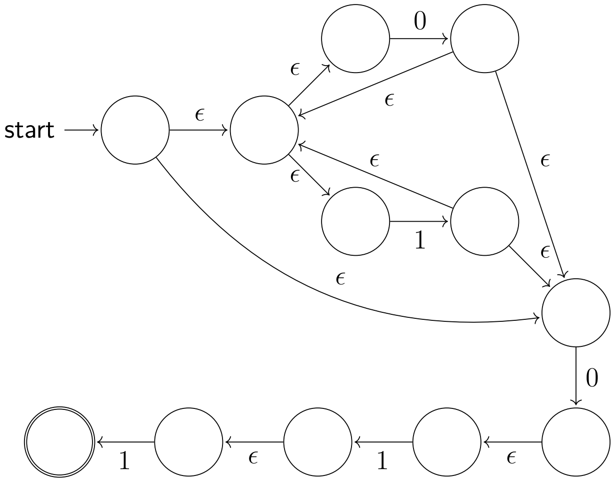
HW#3 Problem 4

$(0 \cup 1)^*$



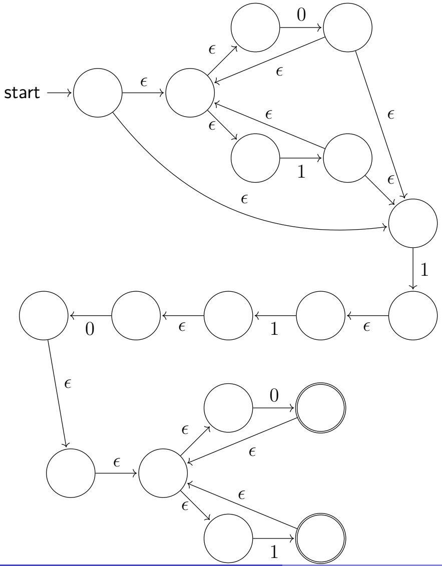
HW#3 Problem 4

$(0 \cup 1)^*011$



HW#3 Problem 4

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



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

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

(b) $\{w \mid w \text{ doesn't contain the substring } 001\}$

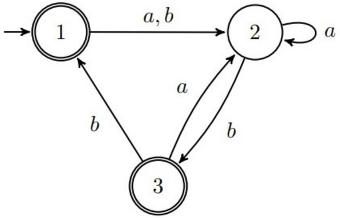
HW#3 Problem 5

第一小題：偶數位爲 1
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 爲奇位數的情況 $((0 \cup 1)1)^*(0 \cup 1 \cup \epsilon)$

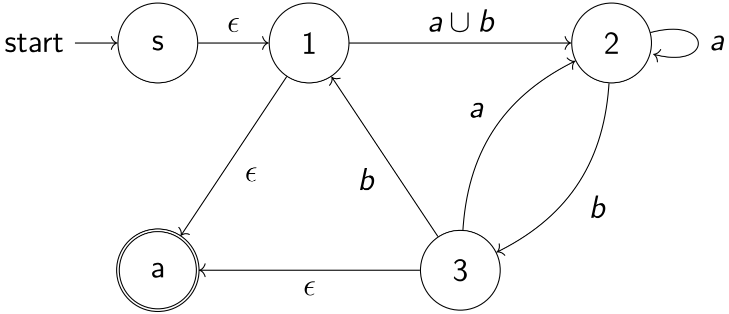
HW#3 Problem 5

第二小題：字串不包含子字串 001
出現 0 之後分爲兩種情形
1. 0 後面是 1 的話可以接 0 或 1
2. 0 後面是 0 的話就只能一直是 0
合起來就是 $(1 \cup 01)^*0^*$

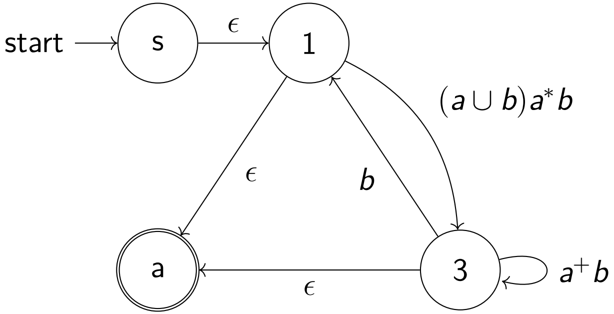
(Exercise 1.21 adapted; 20 points) Use the procedure described in Lemma 1.60 to convert the following finite automaton into a regular expression.



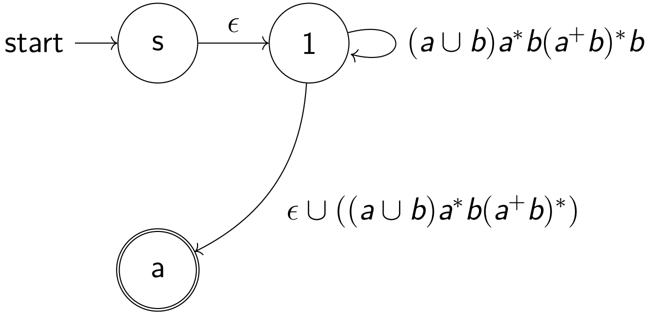
HW#3 Problem 6



HW#3 Problem 6

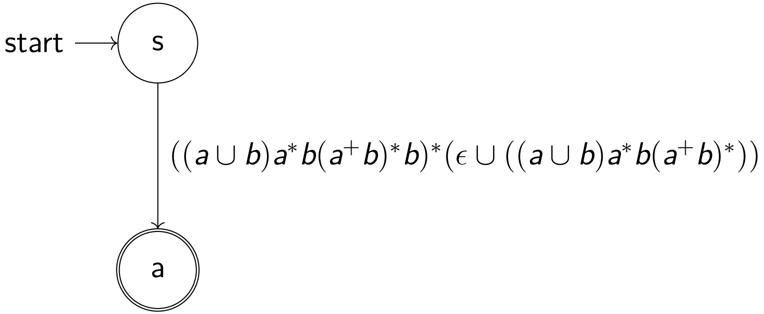

$$\begin{tikzcd}
\text{start} \arrow[r] & s \arrow[r, "\epsilon"] & 1 \arrow[dl, "\epsilon"] \arrow[dr, bend left, "(a \cup b)a^* b"] & \\
& a & & 3 \arrow[ul, "b"] \arrow[ll, "\epsilon"] \arrow[loop right, "a^+ b"]
\end{tikzcd}$$

HW#3 Problem 6



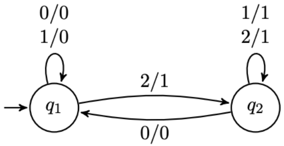
$$\begin{tikzcd}
\text{start} \arrow[r] & s \arrow[r, "\epsilon"] & 1 \arrow[loop right, "(a \cup b)a^*b(a^+b)^*b"] \arrow[dl, bend left, "\epsilon \cup ((a \cup b)a^*b(a^+b)^*)"] \\
& a &
\end{tikzcd}$$

HW#3 Problem 6

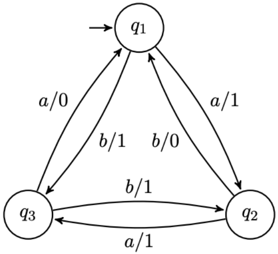

start
s
$((a \cup b)a^*b(a^+b)^*b)^*(\epsilon \cup ((a \cup b)a^*b(a^+b)^*))$
a

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

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 122021

(b) T_2 on input `abbaab`

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 2 輸出 1 跑到 q_2
q_2 吃 0 輸出 0 跑到 q_1
q_1 吃 2 輸出 1 跑到 q_2
q_2 吃 1 輸出 1 跑到 q_2
輸出 011011

HW#3 Problem 7

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_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
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
$\delta : Q \times \Sigma \rightarrow Q \times \Gamma$ 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#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

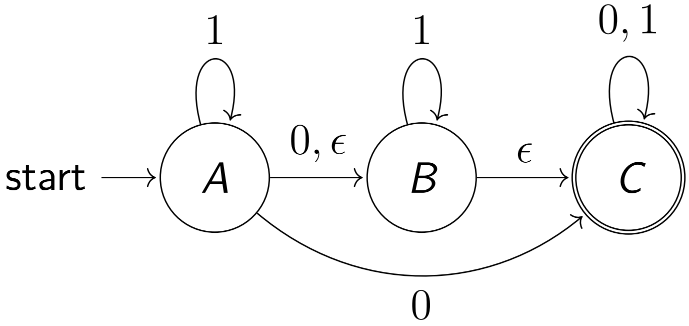
Claim: All languages *all*-NFAs recognize are regular.

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.
- $q' = \{q\}$.
- $F' = P(F)$.

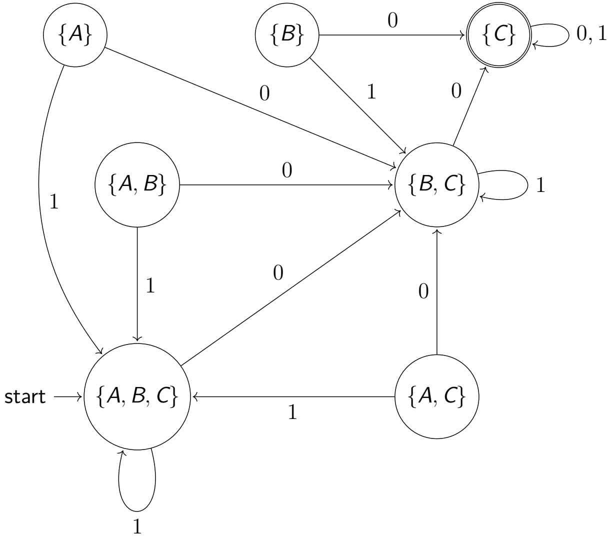
HW#4 Problem 1

For example: *all*-NFA N:



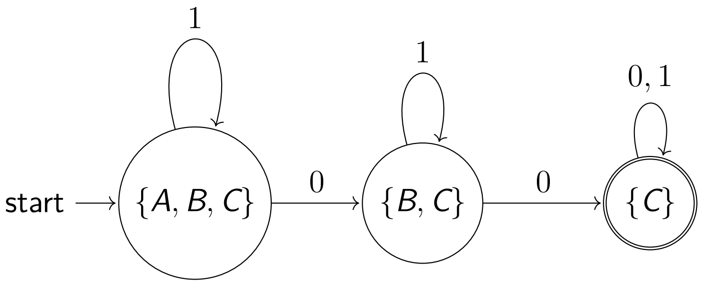
HW#4 Problem 1

For example: DFA M:



HW#4 Problem 1

Simplify M:



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

HW#4 Problem 2

(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 2

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

Let s_{AB} be a string with the minimum length that leads q_A and q_B into the same state g.

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

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

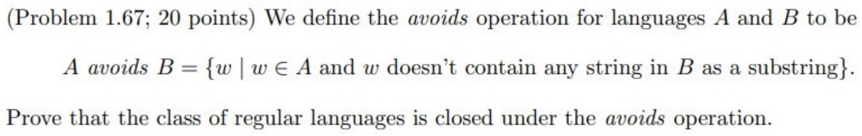
Then s_2 can be removed.

HW#4 Problem 2

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 3



(Problem 1.67; 20 points) We define the *avoids* operation for languages A and B to be

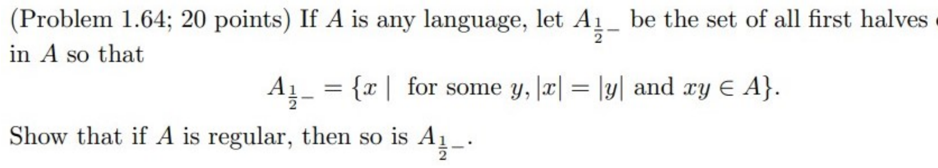
$$A \textit{ avoids } B = \{w \mid w \in A \text{ and } w \text{ doesn't contain any string in } B \text{ as a substring}\}.$$

Prove that the class of regular languages is closed under the *avoids* operation.

HW#4 Problem 3

Let A, B be regular languages.
We can write A *avoids* B as $A \cap \overline{(\Sigma^* B \Sigma^*)}$. Since we know regular languages are closed under concatenation, intersection, and complement, and that A, B, and Σ^* are regular, so $A \cap \overline{(\Sigma^* B \Sigma^*)}$ is regular.

HW#4 Problem 4



(Problem 1.64; 20 points) If A is any language, let $A_{\frac{1}{2}-}$ be the set of all first halves
in A so that

$$A_{\frac{1}{2}-} = \{x \mid \text{ for some } y, |x| = |y| \text{ and } xy \in A\}.$$

Show that if A is regular, then so is $A_{\frac{1}{2}-}$.

HW#4 Problem 4

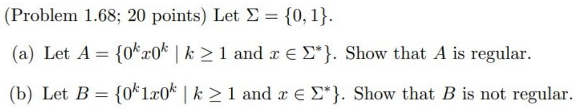
Suppose that A is the language that an DFA $D = (Q, \Sigma, \delta, q, F)$ recognizes. Now we can construct a NFA $N = (Q', \Sigma, \delta', q', F')$ that recognizes $A_{\frac{1}{2}-}$ as follows:

- $Q' = \{Q \times Q\} \cup \{q_0\}$.
- $\delta'(q_0, \epsilon) = (q, r)$ for all $r \in F$
  $\delta'((r_1, r_2), a) = (\delta(r_1, a), z)$ for any z such that there exists some $c \in \Sigma$ with $\delta(z, c) = r_2$.
- $q' = q_0$.
- $F' = \{(r, r) | r \in Q\}$.

HW#4 Problem 4

The idea is that two DFA works simultaneously, one starts from the start state q and recognizes A, and the other starts from one of the accepting states $r \in F$ and recognizes A^R. Whenever the former DFA reads in an input a, we feed a letter c to the latter DFA to let both DFAs move forward for one step.
So, if both DFAs stop at the same state, we know that the two strings are of same length and the concatenation of them are in A.

HW#4 Problem 5



(Problem 1.68; 20 points) Let $\Sigma = \{0, 1\}$.

(a) Let $A = \{0^k x 0^k \mid k \geq 1 \text{ and } x \in \Sigma^*\}$. Show that A is regular.

(b) Let $B = \{0^k 1 x 0^k \mid k \geq 1 \text{ and } x \in \Sigma^*\}$. Show that B is not regular.

HW#4 Problem 5

(a) Observe that $A = \{0x0 | x \in \Sigma^*\}$, thus $A = 0(0 \cup 1)^*0$ which is regular.
(b) Use the pumping lemma:
Let s be 0^p10^p, where p is the pumping length for B.
When dividing s as xyz, because $|xy| < p$, x and y must consist of only 0s, and $xy^2z = 0^l10^p$ where $l > p$. Therefore $xy^2z \notin B$.

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 $\bar{A}$ is regular, $\bar{\bar{A}} = A$ must be regular. On the other hand, if A is not regular, $\bar{A}$ must not be regular.

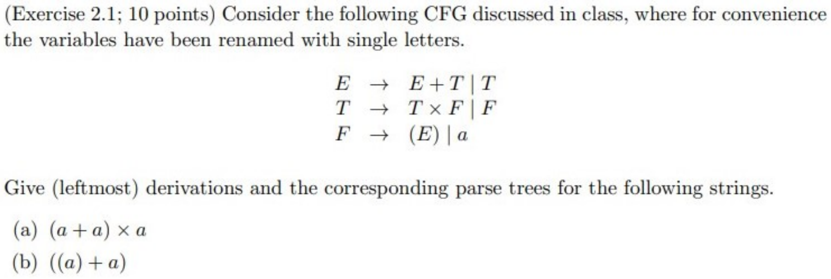
HW#4 Problem 6

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

$$\begin{aligned} E &\rightarrow E + T \mid T \\ T &\rightarrow T \times F \mid F \\ F &\rightarrow (E) \mid a \end{aligned}$$

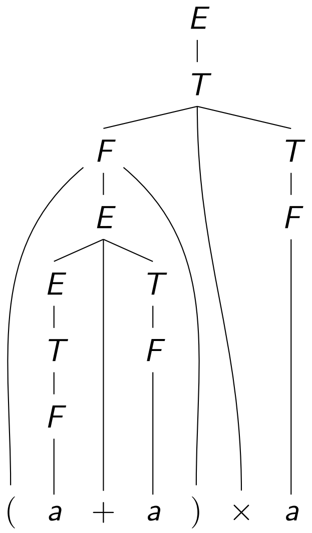
Give (leftmost) derivations and the corresponding parse trees for the following strings.

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

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

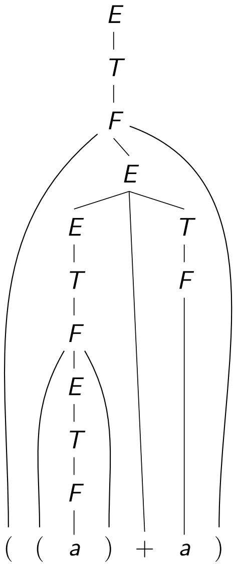
HW#5 Problem 1 (a)

$(a + a) \times a$



HW#5 Problem 1 (b)

$((a) + a)$



HW#5 Problem 2

(Exercise 2.4; 10 points) Give CFGs 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 $\}$

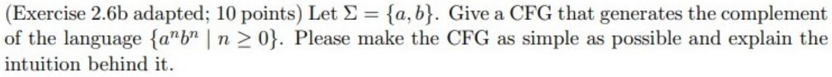
$S \rightarrow CB$
$B \rightarrow CCB \mid \epsilon$
$C \rightarrow 0 \mid 1$

HW#5 Problem 2 (b)

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

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

HW#5 Problem 3


(Exercise 2.6b adapted; 10 points) Let $\Sigma = \{a, b\}$. Give a CFG that generates the complement of the language $\{a^n b^n \mid n \geq 0\}$. Please make the CFG as simple as possible and explain the intuition behind it.

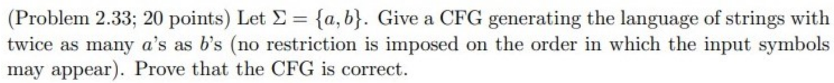
HW#5 Problem 3

Consider two cases: (1) The string starts with b or ends with a, (2)The string starts with a and ends with b, but contains a substring in form of case 1.

$S \rightarrow bA \mid Aa \mid aSb$

$A \rightarrow Aa \mid Ab \mid \epsilon$

HW#5 Problem 4



(Problem 2.33; 20 points) Let $\Sigma = \{a, b\}$. Give a CFG generating the language of strings with twice as many a's as b's (no restriction is imposed on the order in which the input symbols may appear). Prove that the CFG is correct.

HW#5 Problem 4

The CFG G generates the language $C = \{w \mid w \text{ contains twice as many } a\text{'s as } b\text{'s}\}$:
$S \to aaSb \mid aSbSa \mid bSaa \mid SS \mid \epsilon$

Let the string $s \in C$ is of length k, we can prove that G generates s by strong induction on k :

Base case($k = 0$): $s = \epsilon \in L(G)$.
Inductive step: Let $s = s_1 \cdots s_k$ and $c_i =$ the number of a's minus twice the number of b's in $s_1 \cdots s_i$, consider two cases:
(1) There exists $c_i = 0$ for some $0 < i < k$, then we can let $s = pq$ where p is the first i letters of s, by induction hypothesis we know both p and $q \in L(G)$. Therefore the rule $S \to SS$ generates s.

HW#5 Problem 4

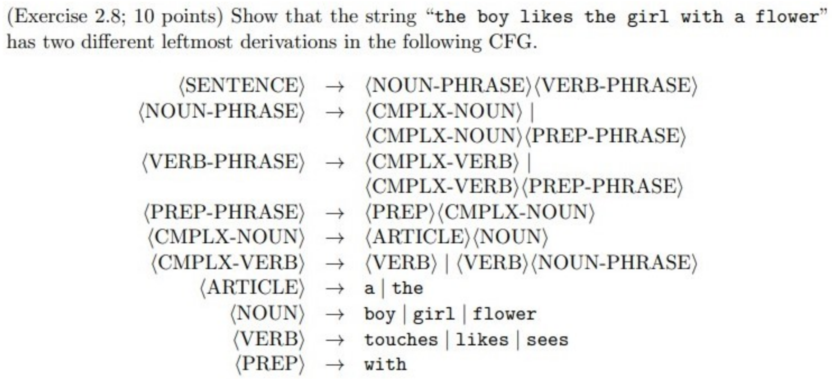
(2) $c_i \neq 0$ for all $0 < i < k$, then there are three subcases:

(i) s starts with b, then $c_i < 0$ for all $0 < i < k$, and s must end with aa. Therefore $s = bpaa$ where $p \in L(G)$, and the rule $S \to bSaa$ generates s.

(ii) s starts with a and $c_i >= 0$ for all $0 < i < k$, then $s = aapb$ where $p \in L(G)$, and the rule $S \to aaSb$ generates s.

(iii) s starts with a and $c_i < 0$ for some $0 < i < k$, then $s = apbqa$ where $p, q \in L(G)$, and the rule $S \to aSbSa$ generates s.

HW#5 Problem 5



(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.

⟨SENTENCE⟩	→	⟨NOUN-PHRASE⟩⟨VERB-PHRASE⟩
⟨NOUN-PHRASE⟩	→	⟨CMPLX-NOUN⟩ | ⟨CMPLX-NOUN⟩⟨PREP-PHRASE⟩
⟨VERB-PHRASE⟩	→	⟨CMPLX-VERB⟩ | ⟨CMPLX-VERB⟩⟨PREP-PHRASE⟩
⟨PREP-PHRASE⟩	→	⟨PREP⟩⟨CMPLX-NOUN⟩
⟨CMPLX-NOUN⟩	→	⟨ARTICLE⟩⟨NOUN⟩
⟨CMPLX-VERB⟩	→	⟨VERB⟩ | ⟨VERB⟩⟨NOUN-PHRASE⟩
⟨ARTICLE⟩	→	`a` | `the`
⟨NOUN⟩	→	`boy` | `girl` | `flower`
⟨VERB⟩	→	`touches` | `likes` | `sees`
⟨PREP⟩	→	`with`

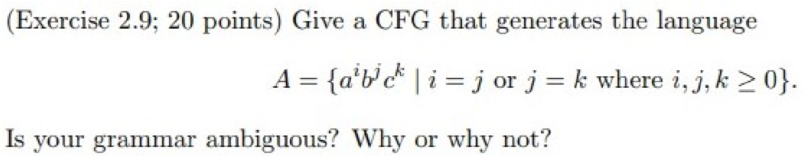
HW#5 Problem 5

$\mathrm{S} \Rightarrow \mathrm{NP}\,\mathrm{VP} \Rightarrow \mathrm{CN}\,\mathrm{VP} \Rightarrow \mathrm{A}\,\mathrm{N}\,\mathrm{VP} \Rightarrow$ 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 5

$\mathrm{S} \Rightarrow \mathrm{NP}\,\mathrm{VP} \Rightarrow \mathrm{CN}\,\mathrm{VP} \Rightarrow \mathrm{A}\,\mathrm{N}\,\mathrm{VP} \Rightarrow$ the $\mathrm{N}\,\mathrm{VP} \Rightarrow$ the boy $\mathrm{VP} \Rightarrow$
the boy $\mathrm{CV}\,\mathrm{PP} \Rightarrow$ the boy $\mathrm{V}\,\mathrm{NP}\,\mathrm{PP} \Rightarrow$ the boy likes $\mathrm{NP}\,\mathrm{PP} \Rightarrow$
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 6


(Exercise 2.9; 20 points) Give a CFG 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 6

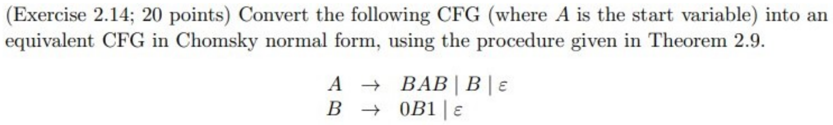
設計一個 CFG 生成 $a^ib^jc^k$ 其中 $i = j \vee j = k$
我們可以走兩條路線：$i = j$ 路線或 $j = k$ 路線
對 $i = j$ 路線而言，左半邊要有等量生成的 a 與 b，右邊的 c 則是任意生成
$j = k$ 路線也是以此類推

HW#5 Problem 6

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

解釋這個 CFG 是否爲 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 7



(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.

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

HW#5 Problem 7

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

HW#5 Problem 7

第一程序：增加新的 start symbol

加上 $S_0 \to A$

$S_0 \to A$

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

$B \to 0B1 \mid \epsilon$

HW#5 Problem 7

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

HW#5 Problem 7

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 7

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 7

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 7

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 7

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 7

第四程序：分割其它 rule

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

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

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

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

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

$C_4 \to 0B$

$C_5 \to 0B$

HW#5 Problem 7

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