Homework 6 - 10

劉韋成 蘇俊杰

Menu

HW#6 Problem 1

(Exercise 2.2; 20 points)

(a) Use the languages $A = \{a^n b^n c^m \mid m, n \geq 0\}$ and $B = \{a^m b^n c^n \mid m, n \geq 0\}$, together with the fact that $\{a^n b^n c^n \mid m, n \geq 0\}$ is not context free, to show that the class of context-free languages is not closed under intersection.

(b) Use the preceding part and DeMorgan's law to show that the class of context-free languages is not closed under complementation.

HW#6 Problem 1 (a)

Transform languages A and B into the new forms:
$A = \{a^i b^j c^k \mid (i = j) \wedge (i, j, k \geq 0)\}$, and
$B = \{a^i b^j c^k \mid (j = k) \wedge (i, j, k \geq 0)\}$

The intersection of A and B
$= \{a^i b^j c^k \mid (i = j) \wedge (j = k) \wedge (i, j, k \geq 0)\}$, which is equal to
$\{a^n b^n c^n \mid m, n \geq 0\}$

We've known that A and B are context-free languages, but the intersection of A and $B = \{a^n b^n c^n \mid m, n \geq 0\}$ is not context free, so the class of context-free languages is not closed under intersection.

HW#6 Problem 1 (b)

DeMorgan's law: $A \cap B = \overline{\overline{A} \cup \overline{B}}$

We've known that the class of context-free languages is closed under union. Now suppose that the class of context-free languages is closed under complementation and A and B are two context-free languages:

A and B are context free.
$\Rightarrow$ $\overline{A}$ and $\overline{B}$ are context free.
$\Rightarrow$ $\overline{A} \cup \overline{B}$ is context free.
$\Rightarrow$ $\overline{\overline{A} \cup \overline{B}}$ is context free.
$\Rightarrow$ $A \cap B$ is context free.
$\Rightarrow$ **false**

HW#6 Problem 1 (b)

We've known that the class of context-free languages is not closed under intersection in problem1 (a), contradiction.

So the class of context-free languages is not closed under complementation.

HW#6 Problem 2

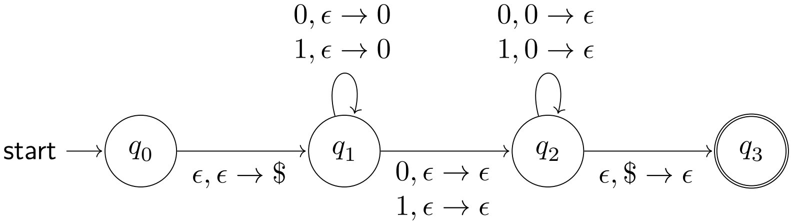
(Exercise 2.5; 20 points) Give informal descriptions and state diagrams of pushdown automata for the following languages. In all parts the alphabet Σ is $\{0, 1\}$.

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

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

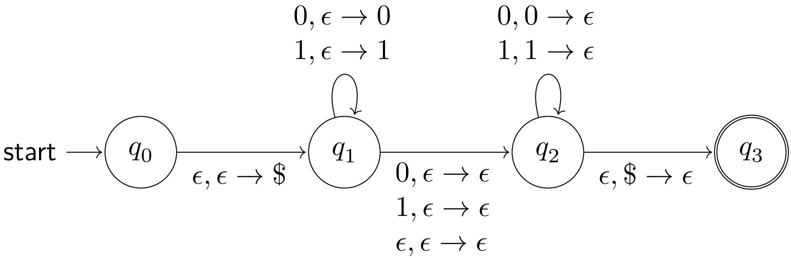
HW#6 Problem 2 (a)

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



HW#6 Problem 2 (b)

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

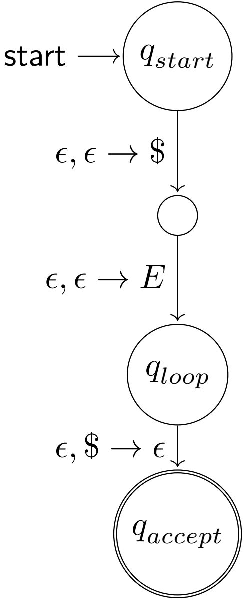


HW#6 Problem 3

(Exercise 2.12; 10 points) Convert the following CFG to an equivalent PDA, using the procedure given in Theorem 2.20.

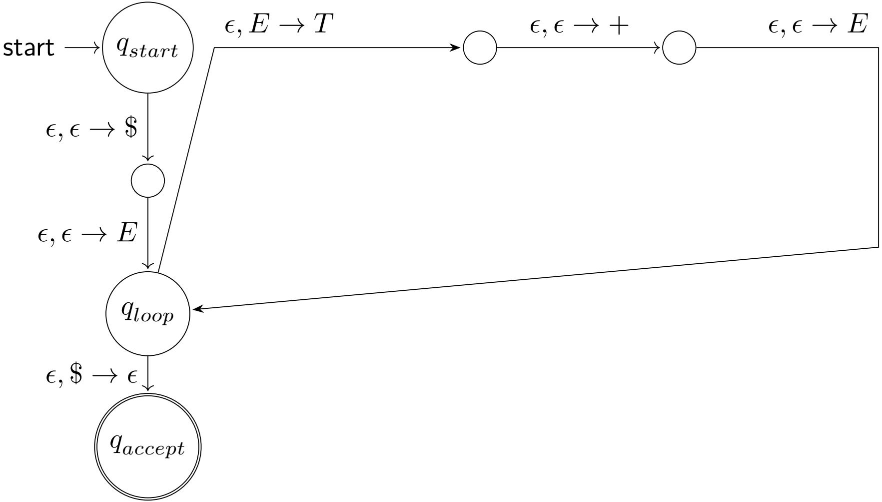
$$\begin{aligned} E &\rightarrow E + T \mid T \\ T &\rightarrow T \times F \mid F \\ F &\rightarrow (\ E\ ) \mid a \end{aligned}$$

HW#6 Problem 3



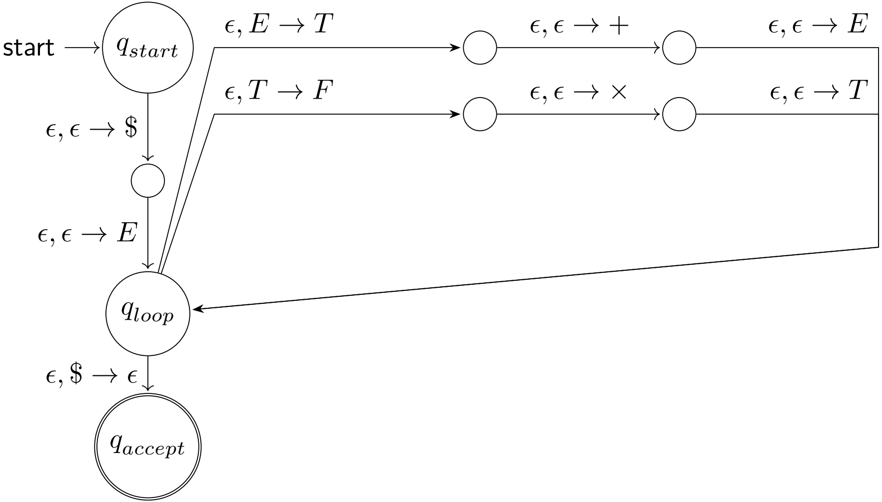
HW#6 Problem 3

$E \to E + T$



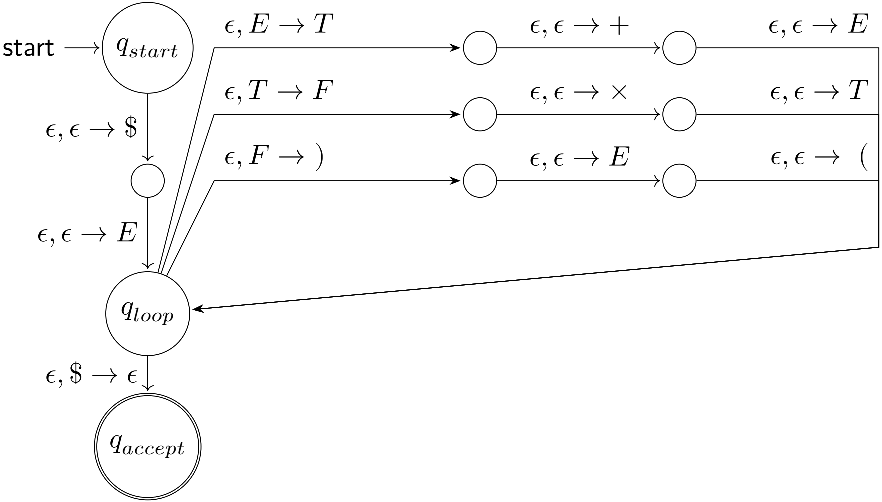
HW#6 Problem 3

$T \to T \times F$



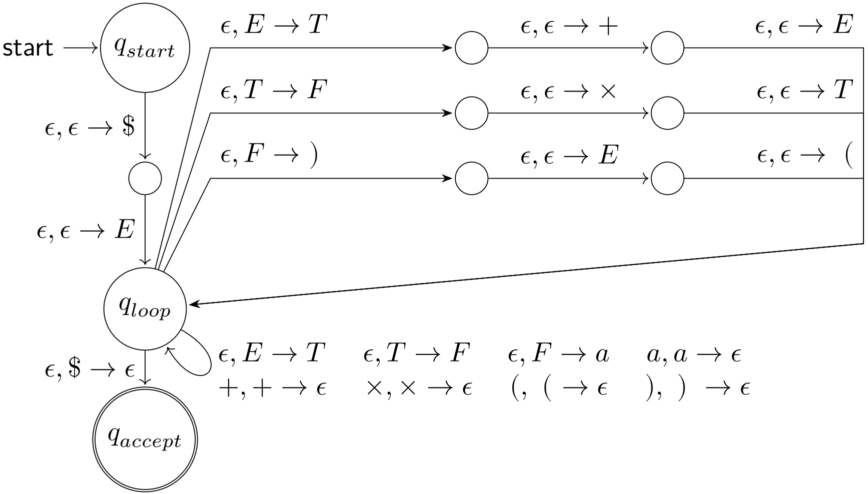
HW#6 Problem 3

$F \to (E)$



HW#6 Problem 3

Remaining grammar



HW#6 Problem 4

4. (Problem 2.39; 20 points) Let $G = (V, \Sigma, R, \langle\text{STMT}\rangle)$ be the following grammar.

$$\begin{aligned}
\langle\text{STMT}\rangle &\rightarrow \langle\text{ASSIGN}\rangle \mid \langle\text{IF-THEN}\rangle \mid \langle\text{IF-THEN-ELSE}\rangle \\
\langle\text{IF-THEN}\rangle &\rightarrow \texttt{if condition then } \langle\text{STMT}\rangle \\
\langle\text{IF-THEN-ELSE}\rangle &\rightarrow \texttt{if condition then } \langle\text{STMT}\rangle \texttt{ else } \langle\text{STMT}\rangle \\
\langle\text{ASSIG}\rangle &\rightarrow \texttt{a} := \texttt{1}
\end{aligned}$$

$$\Sigma = \{\texttt{if}, \texttt{condition}, \texttt{then}, \texttt{else}, \texttt{a} := \texttt{1}\}$$

$$V = \{\langle\text{STMT}\rangle, \langle\text{IF-THEN}\rangle, \langle\text{IF-THEN-ELSE}\rangle, \langle\text{ASSIG}\rangle\}$$

G is a *natural-looking* grammar for a fragment of a programming language, but G is ambiguous.

(a) Show that G is ambiguous.

(b) Give a new unambiguous grammar for the same language.

HW#6 Problem 4 (a)

Counterexample:
`if condition then if condition then a:=1 else a:=1`

There are two way to obtain this language:

1.
$\langle$STMT$\rangle$
$\Rightarrow$ $\langle$IF-THEN$\rangle$
$\Rightarrow$ `if condition then` $\langle$STMT$\rangle$
$\Rightarrow$ `if condition then` $\langle$IF-THEN-ELSE$\rangle$
$\Rightarrow$ `if condition then if condition then` $\langle$STMT$\rangle$ `else` $\langle$STMT$\rangle$
$\Rightarrow$ `if condition then if condition then a:=1 else a:=1`

HW#6 Problem 4 (a)

2.
$\langle$STMT$\rangle$
$\Rightarrow$ $\langle$IF-THEN-ELSE$\rangle$
$\Rightarrow$ `if condition then` $\langle$STMT$\rangle$ `else` $\langle$STMT$\rangle$
$\Rightarrow$ `if condition then` $\langle$IF-THEN$\rangle$ `else` $\langle$STMT$\rangle$
$\Rightarrow$ `if condition then if condition then` $\langle$STMT$\rangle$ `else` $\langle$STMT$\rangle$
$\Rightarrow$ `if condition then if condition then a:=1 else a:=1`

So G is ambiguous.

HW#6 Problem 4 (b)

$\langle$STMT$\rangle \rightarrow \langle$ASSIGN$\rangle$ | $\langle$IF-THEN$\rangle$ | $\langle$IF-THEN-ELSE$\rangle$
$\langle$IF-THEN$\rangle \rightarrow$ `if condition then` $\langle$STMT$\rangle$
$\langle$IF-THEN-ELSE$\rangle \rightarrow$ `if condition then` $\langle$STMT$\rangle$ `else` $\langle$STMT$\rangle$
$\langle$ASSIGN$\rangle \rightarrow$ `a:=1`

The problem of the original grammar G is that when $\langle$IF-THEN-ELSE$\rangle$ appears, we expect that the `if` and `else` in it should be matched, but the $\langle$STMT$\rangle$ in front of the `else` may have a unmatched `if` which may wrongly match the `else`.

To solve the problem, we need to guarantee that all `if` and `else` between the `if` and `else` in $\langle$IF-THEN-ELSE$\rangle$ should already be matched.

HW#6 Problem 4 (b)

A new unambiguous grammar G':

$\langle$STMT$\rangle$ $\rightarrow$ $\langle$ASSIGN$\rangle$ | $\langle$IF-THEN$\rangle$ | $\langle$IF-THEN-ELSE$\rangle$

$\langle$IF-THEN$\rangle$ $\rightarrow$ `if condition then` $\langle$STMT$\rangle$

$\langle$IF-THEN-ELSE$\rangle$ $\rightarrow$ `if condition then` $\langle$STMT-M$\rangle$ `else` $\langle$STMT$\rangle$

$\langle$STMT-M$\rangle$ $\rightarrow$ $\langle$ASSIGN$\rangle$ | $\langle$IF-THEN-ELSE-M$\rangle$

$\langle$IF-THEN-ELSE-M$\rangle$ $\rightarrow$ `if condition then` $\langle$STMT-M$\rangle$ `else` $\langle$STMT-M$\rangle$

$\langle$ASSIGN$\rangle$ $\rightarrow$ `a:=1`

We guarantee that all `if` and `else` in -M variables have already been matched.

HW#6 Problem 5

5. (Problem 2.32; 20 points) Let $A/B = \{w \mid wx \in A \text{ for some } x \in B\}$. Show that, if A is context free and B is regular, then A/B is context free.

HW#6 Problem 5

建構 A/B 對應的 PDA
令 PDA $M_A = (Q_A, \Sigma, \Gamma, \delta_A, q_{0A}, F_A)$
DFA $M_B = (Q_B, \Sigma, \delta_B, q_{0B}, F_B)$
建構 $M_{A/B}$ 的思路是這樣子：
第一階段：先把 w 餵給 M_A
第二階段：再「假裝」輸入一些 symbols，同時跑在 M_A 與 M_B 上
可以想像，第一階段並不需要理會 M_B，所以只需要記錄 M_A 的狀態
而第二階段就需要同時記錄兩台機器的狀態
當兩台機器都待在 accepting state 時就行了

HW#6 Problem 5

$M_{A/B} = (Q, \Sigma, \Gamma, \delta, q_0, F)$

$Q = Q_A \cup Q_A \times Q_B$
左邊的 Q_A 是第一步使用的 state，右邊的 $Q_A \times Q_B$ 是第二步用的

$q_0 = q_{0A}$
$F = F_A \times F_B$

HW#6 Problem 5

第一階段：模擬 M_A
對於 $q_A \in Q_A, b \in \Gamma$
若 $a \in \Sigma$，直接模擬 M_A 的動作
$\delta(q_A, a, b) = \delta_A(q_A, a, b)$

若是 $a = \epsilon$，有兩種路：模擬 M_A 的動作，或者跳轉到第二步驟
跳轉瞬間不該消耗 stack 裡頭的符號，因為這並不是 M_A 的動作，所以會把消耗掉的 b 還回去
跳轉後 M_A 應該待在本來的狀態，而 M_B 從 q_{0B} 開始
$\delta(q_A, \epsilon, b) = \delta_A(q_A, \epsilon, b) \cup \{((q_A, q_{0B}), b)\}$

HW#6 Problem 5

第二階段：同步模擬 M_A 與 M_B

對於 $q_A \in Q_A, q_B \in Q_B, b \in \Gamma$

若是 $a \in \Sigma$

實際輸入 symbols 的程序應該在第一階段完結，第二階段不能吃

$\delta((q_A, q_B), a, b) = \{\}$

若是 $a = \epsilon$

雖然實際上這機器不會再接收輸入，但我們還是得「假裝」有 symbols 打進去

要怎麼模擬？把此時輸入每個 symbol 或 ϵ 的可能性彙總起來

$\delta((q_A, q_B), \epsilon, b) =$

$$\{((q'_A, q'_B), c) \mid \exists a \in \Sigma.(q'_A, c) \in \delta_A(q_A, a, b) \wedge q'_B \in \delta_B(q_B, a)\}$$
$$\cup$$
$$\{((q'_A, q'_B), c) \mid (q'_A, c) \in \delta_A(q_A, \epsilon, b) \wedge q'_B = q_B\}$$

也就是，模擬某個字 a 同時輸入給 M_A 與 M_B 時的反應

HW#6 Problem 5

於是我們得出了 A/B 是 context-free language 的結論

HW#6 Problem 6

6. (10 points) Prove, using the pumping lemma, that $\{a^m b^n c^{m\times n} \mid m, n \geq 1\}$ is not context free.

HW#6 Problem 6

Let $A = \{a^m b^n c^{m \times n} \mid m, n \geq 1\}$.

Prove that A is not regular.

Use the pumping lemma:

Let s be $a^p b^p c^{p \times p}$, where p is the pumping length for A.

HW#6 Problem 6

Cases of dividing s as $uvxyz$ (where $|vy| > 0$ and $|vy| \leq p$):

- Both v and y contain only one type of symbol, e.g.,
  $\overbrace{a \cdot \underbrace{\cdots}_{v} \underbrace{\cdot\, a\, b \cdot}_{x}}^{p}\!\!\overbrace{\underbrace{\cdots}_{y} \cdot b}^{p} \overbrace{c \cdots c}^{p \times p}$, in which case, uv^2xy^2z will have wrong number of c's which is not equal to :
  the number of a's $\times$ the number of b's,
  $a^{p+i}b^{p+j}c^{p\times p}$
  or
  $\overbrace{a \cdots a}^{p}\, \overbrace{b \cdot \underbrace{\cdots}_{v} \underbrace{\cdot\, b\, c \cdot}_{x}}^{p}\!\!\overbrace{\underbrace{\cdots}_{y} \cdot c}^{p \times p}$, in which case, uv^2xy^2z will also have wrong number of c's which is not equal to :
  the number of a's $\times$ the number of b's,
  $a^{p}b^{p+i}c^{p\times p+j}$

HW#6 Problem 6

Cases of dividing s as $uvxyz$ (where $|vy| > 0$ and $|vy| \leq p$):

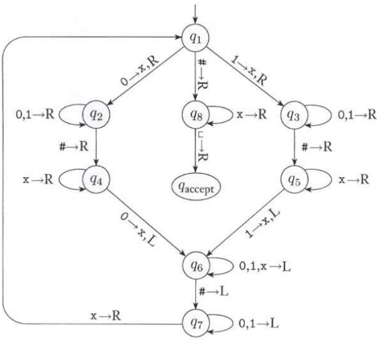
- Either v or y contains more than one type of symbol, e.g., $a \cdot \underbrace{\cdots}_{v} \underbrace{\cdots}_{x} \underbrace{\cdot ab}_{y} \cdot \cdots \cdot bc \cdots c$, in which case, uv^2xy^2z will have some a's and b's out of order and so is not in A.

HW#7 Problem 1

(Exercise 3.2; 10 points) Consider the Turing machine for $\{w\#w \mid w \in \{0,1\}^*\}$ discussed in class. Give the sequence of configurations (using the notation uqv for a configuration) that the machine goes through when started on the input 01#01.

HW#7 Problem 1

$q_1 01\#01$	$xq_1 1\#x1$	$xxq_1\#xx$
$xq_2 1\#01$	$xxq_3\#x1$	$xx\#q_8 xx$
$x1q_2\#01$	$xx\#q_5 x1$	$xx\#xq_8 x$
$x1\#q_4 01$	$xx\#xq_5 1$	$xx\#xxq_8$
$x1q_6\#x1$	$xx\#q_6 xx$	$xx\#xxq_{accept}$
$xq_7 1\#x1$	$xxq_6\#xx$	
$q_7 x1\#x1$	$xq_7 x\#xx$	



HW#7 Problem 2

(20 points) Give a *formal* description (with a state diagram) of a Turing machine that decides the language $\{w \in \{0,1\}^* \mid w \text{ is nonempty and contains twice as many 1s as 0s}\}$.

HW#7 Problem 2

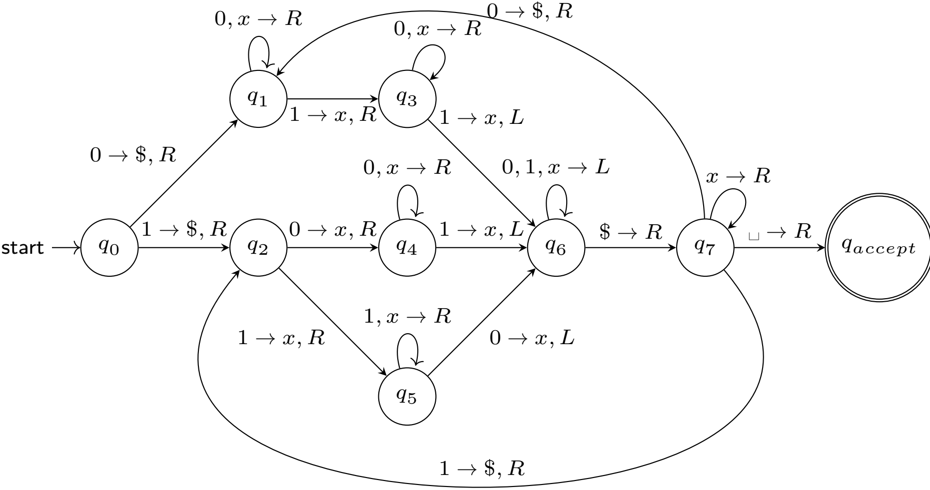
$L = \{\omega \in \{0,1\}^* \mid \omega$ is nonempty and contains twice as many as 1s as 0s $\}$.

TM_L is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where

- $Q = \{q_0, q_1, \cdots, q_7, q_{accept}, q_{reject}\}$,
- $\Sigma = \{0, 1\}$,
- $\Gamma = \{0, 1, \sqcup, \$\}$,
- q_0 is the initial state,
- q_{accept} is the accept state,
- q_{reject} is the reject state, and

HW#7 Problem 2

- $\delta =$ (all undescribed transitions lead to q_{reject})



HW#7 Problem 3

(Exercise 3.7; 10 points) Explain why the following is not a description of a legitimate Turing machine.

M_{bad} = "The input is a polynomial p over variables $x_1, \ldots, x_k$:

(a) Try all possible settings of $x_1, \ldots, x_k$ to integer values.

(b) Evaluate p on all of these settings.

(c) If any of these settings evaluates to 0, *accept*; otherwise, *reject*."

HW#7 Problem 3

為何這台機器不是合法的圖靈機？
這台機器會試圖嘗試所有可能性
但「所有」是多少？
要將係數是任何整數的可能性都考慮進去
所以會有無窮多種可能性
如果係數只有一個，那就是 0 1 2 ... 嘗試下去
但是如果有兩個呢？
00 10 20...？還是 00 10 01 20 11 02...？
這台機器並沒有說明它的執行順序

HW#7 Problem 4

(Problem 3.16; 10 points) Show that the collection of decidable languages is closed under *concatenation*.

HW#7 Problem 4

做出一台圖靈機 decide 兩個 decidable language
假設兩個 decidable language A B 對應到的 Decider M_A M_B
做出 M = "On input w,
1. Divide w into xy ($|w| + 1$ different division)
2. Input x to M_A and y to M_B (try any possible with $|w| + 1$ division)
3. Repeat Step 1 and 2, if both M_A M_B accept on some x y, accept, otherwise, reject."
由於 w 是有限長度字串，它的分割法只有 $|w| + 1$ 種
而且 Decider M_A 與 M_B 都會停機
所以 M 也一定會在有限時間內停機，M decides the concatenation of A and B

HW#7 Problem 5

5. (Problem 3.18; 10 points) A ***Turing machine with a doubly infinite tape*** is similar to an ordinary Turing machine, but its tape is infinite to the left as well as to the right. The tape is initially filled with blanks except for the portion that contains the input. Computation is defined as usual except that the head never encounters an end to the tape, as it moves left. Show that this type of Turing machine recognizes the class of Turing-recognizable languages.

HW#7 Problem 5

要說明兩端無限長的圖靈機，辨識效果和一般圖靈機相同

首先我們用兩端無限長的圖靈機來模擬一般圖靈機
概念上很簡單，就是封印左邊的紙帶不用
實作上就是先在輸入的左方放一個標記，之後只要踩到這個標記就往右退回去

HW#7 Problem 5

接下來要說明一般圖靈機有辦法模擬兩端無限長的圖靈機
依照課堂上講過的定理，對兩條紙帶的圖靈機都有一個一般圖靈機與之等價
所以這邊我們利用一台有兩條紙帶的圖靈機去模擬
效力等同於拿一個一般圖靈機去模擬它

HW#7 Problem 5

兩條紙帶的第一條代表兩端無限長圖靈機的右半邊紙帶，第二條代表左半邊

輸入的字串都是放在第一條紙帶上

第一條用來模擬右半邊的運算，第二條則模擬左半邊

HW#7 Problem 5

為方便理解，補充實作上的細節
首先將第一條紙帶的輸入往右移並在開頭加上標記
第二條紙帶也在開頭加上標記
比如這樣的輸入（空格加上編號以方便看出位置上的關聯）

			↓					
⋯	⊔4	⊔3	1	2	3	⊔1	⊔2	⋯

在兩條紙帶的圖靈機上會先轉換成這樣再操作

	↓					
\$	1	2	3	⊔1	⊔2	⋯

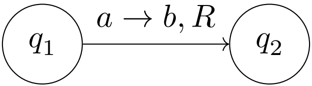
↑			
\$	⊔3	⊔4	⋯

HW#7 Problem 5

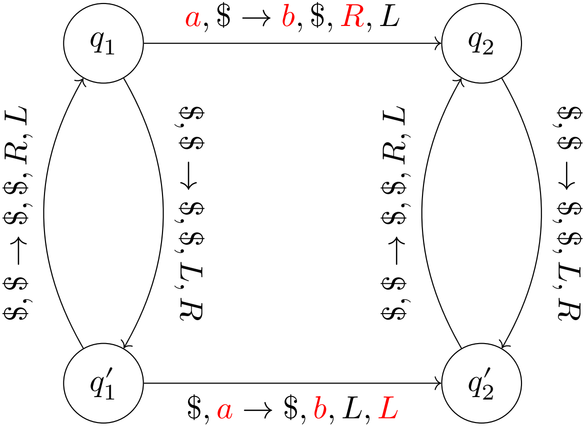
將 state diagram 複製出一份並將 LR 顛倒
原本的 diagram 給第一條紙帶用，以模擬右半邊紙帶，另一個以此類推
在左半右半切換時（對應到碰上 $ 時）移動到另一個 state diagram

HW#7 Problem 5

也就是說原本這樣的 transition



會變成



HW#7 Problem 6

6. (Problem 3.20; 20 points) A ***Turing machine with stay put instead of left*** is similar to an ordinary Turing machine, but the transition function has the form

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, S\}$$

At each point the machine can move instead its head right or let it stay in the same position. Show that this Turing machine variant is *not* equivalent to the usual version. What class of languages do these machines recognize?

HW#7 Problem 6

一台只能往右與停在原地的圖靈機，它的辨識能力究竟為何？

有一個關鍵是在與這個圖靈機沒辦法往回讀取它在左邊所寫過的玩意
這個永遠不回頭的特性，和某種東西好像有點像...
PDA 能夠把前面的內容塞到 stack，所以似乎不是 PDA
NFA/DFA 呢？

HW#7 Problem 6

我們先用這種圖靈機去模擬一台 DFA
很簡單，把 DFA 的 transition 加上指針不寫入且往右移
並在原本的 accepting states 加入一條讀取空格則跑到 q_{accept} 的 transition 即可

HW#7 Problem 6

那麼要如何用 NFA 模擬這種圖靈機？
這個地方稍微有點難懂

當這台圖靈機往右走的時候，其實我們不需要理會它在原本那格寫了什麼
因為它永遠不回頭
但若這台圖靈機在某格一直待著，我們勢必要記錄現在這格到底內容是什麼
我們會利用 **states** 去記錄
也就是說，除了原本圖靈機的狀態集 Q 之外，我們還需要 $Q \times \Gamma$，包含同時存著圖靈機狀態與紙帶當前字元的 pairs

HW#7 Problem 6

那麼要怎麼把輸入字串餵給 NFA
當圖靈機第一次來到某一格，因為永遠不回頭，這格若不是空格就是輸入字元
如果是輸入字元，就對應到 NFA 輸入字元的動作
其餘地方都是 ϵ-transition，利用 state 本身記錄紙帶內容

那若是往右遇到空格了呢？
我們把一開始在 q 狀態遇到空格記為一個 pair $[q, \sqcup]$
Q 與 Γ 都是有限集合，持續走 $|Q| \times |\Gamma|$ 步之後必然會遇到相同的 pair（遇到相同 pair 時的環境都一樣：右邊都是無盡的空格)
就可以確認這機器不會停機，也就是這機器不接受這個字串
反過來如果從 q 持續走若干步之後到達 q_{accept}，則將 q 狀態當成 accepting state
若 NFA 輸入完字串後停在這裡，就相當於接受了這個字串
我們把符合條件的這種 q 收集起來做成集合 F

HW#7 Problem 6

那麼 NFA 裡頭包含 q_{accept} 的狀態要怎麼處理？
因為圖靈機是碰到 q_{accept} 就直接停機
所以 NFA 的這些狀態應該會有一條 ϵ-transition 連到一個永遠待在原地的 accepting state
令這個 accepting state 為 q'_{accept}
那麼 NFA 的 accepting states 就是 $F \cup \{q'_{accept}\}$

HW#7 Problem 6

假設原本圖靈機的 transition function 為 δ，而 NFA 的 transition relation 是 $\delta'(q' \in \delta'(q_a))$
若是圖靈機在 q 狀態接收到一個 $a \in \Sigma$，而下述的 $X \in \Gamma$
$\delta(q, a) = (q', X, R)$，因為往右走所以不需要理會 X，所以
$(q, a, q') \in \delta'$
$\delta(q, a) = (q', X, S)$，因為停下來了，需要記錄 X，所以
$(q, a, (q', X)) \in \delta'$
這邊會實際吃掉輸入字元

若是圖靈機在 q 狀態接收到一個 $X \in \Gamma$，而下述的 $Y \in \Gamma$
$\delta(q, X) = (q', Y, R)$，因為往右走所以不需要理會 Y，所以
$((q, X), \epsilon, q') \in \delta'$
$\delta(q, X) = (q', Y, S)$，因為停下來了，需要記錄 Y，所以
$((q, X), \epsilon, (q', Y)) \in \delta'$
這邊會用 ϵ-transition 去模擬紙帶的運作

HW#7 Problem 6

NFA 模擬 TM 在 q 狀態接收一個 $a \in \Sigma$，而下述的 $X \in \Gamma$：
$(q, a, q') \in \delta'$
$(q, a, (q', X)) \in \delta'$
NFA 模擬 TM 在 q 狀態接收到一個 $X \in \Gamma$，而下述的 $Y \in \Gamma$
$((q, X), \epsilon, q') \in \delta'$
$((q, X), \epsilon, (q', Y)) \in \delta'$

NFA 模擬 TM 處理 accepting state:
$(q_{accept}, \epsilon, q'_{accept}) \in \delta'$
$((q_{accept}, X), \epsilon, q'_{accept}) \in \delta'$ for all $X \in \Gamma$
$(q'_{accept}, a, q'_{accept}) \in \delta'$ for all $a \in \Sigma$

HW#7 Problem 6

弄了這麼長，結論就是我們能用這種圖靈機模擬 DFA，也能用 NFA 模擬這種圖靈機

因為 DFA 與 NFA 的辨識能力是相同的，所以這種圖靈機的辨識能力也和它們相同

所以這種圖靈機能辨識的語言就局限於 regular languages

HW#7 Problem 7

7. (Problem 3.22; 20 points) Let a k-PDA be a pushdown automaton that has k stacks. Thus a 0-PDA is an NFA and a 1-PDA is a conventional PDA. You already know that 1-PDAs are more powerful (recognizing a larger class of languages) than 0-PDAs.

   (a) Show that 2-PDAs are more powerful than 1-PDAs.

   (b) Show that 3-PDAs are *not* more powerful than 2-PDAs. (Hint: simulate a Turing machine tape with two stacks.)

HW#7 Problem 7

第一小題要說明 2-PDA 比 1-PDA 強
很明顯 2-PDA 當然能夠模擬 1-PDA
那要如何證明 1-PDA 無法模擬 2-PDA？
那我們就找一個 language，可以被一個 2-PDA 給辨識，但卻不是 context-free
我們挑 $0^n1^n0^n1^n$，已知它不是 context-free

HW#7 Problem 7

流程大致是這樣：
在兩個 stacks 當中塞入一個識別符號 $
吃若干個 0 放入第一個 stack
吃 1 並把 0 從第一個 stack pop 掉並把 1 塞入第二個 stack
吃 0 並把 1 從第二個 stack pop 掉並把 0 塞入第一個 stack
吃 1 並把 0 從第一個 stack pop 掉
當兩個 stacks 的頂端都是 $ 則跳到 accepting state（若還有字沒輸入完成，輸進去就會爆掉）
這樣我們就建構出一個能辨識這個語言的 2-PDA

HW#7 Problem 7

注意，兩個方向都要給出說明 2-PDA 可以模擬 1-PDA，但 1-PDA 沒辦法辨識某些 2-PDA 能辨識的語言
這樣才能說明 2-PDA 能辨識的語言集合嚴格大於 1-PDA 的

HW#7 Problem 7

第二小題，說明 3-PDA 的辨識能力與 2-PDA 一樣
這邊往另外一個方向，證明這兩種機器都與另外一種機器具有一樣的能力

HW#7 Problem 7

首先我們要用圖靈機去模擬 2-PDA
用 3-tape TM 來模擬
一號紙帶代表 PDA 的輸入，在有實際輸入時才向右
二號三號分別代表兩個 stacks
指針指到的字代表 stack 頂的內容，一開始先填充識別符號代表 stack 為空
PDA 有 pop 代表會偵測指針指到的字
如果有 pop 且沒有塞字進去，則代表填入空格且向左
有 pop 也有塞入，則代表填入塞入的字且停在原地
沒有 pop 也沒有塞字，則停在原地
沒有 pop 而有塞字，則向右並在右邊這格寫入（一個 R 再一個 S）

HW#7 Problem 7

那麼如何用 2-PDA 去模擬 TM？
首先先在 stack 底部填上識別符號
再來吃入所有輸出到一個 stack 裡頭
此時 stack 頂會是最尾巴的字，我們看不到開頭是什麼
所以就把所有字倒到另一個 stack
基於 stack 的性質，現在另一個 stack 的頂端就會是第一個字元了
我們把這個 stack 的頂端當成圖靈機的指針指向的位置
這個 stack（暫時稱為 1 號 stack）代表指針與其右方，另外一個（2 號 stack）代表左方，距離頂端越遠，代表離指針越遠

HW#7 Problem 7

圖靈機指針向右，就是從 1 號 stack pop 掉並把圖靈機寫入的字元塞到 2 號 stack
當 1 號 stack 看到識別符號，代表圖靈機指針初次跑到一個很右的地方
因為在那之前圖靈機還沒到過，所以這裡自然會是空格，所以就先把空格塞入 1 號 stack 再算下去

圖靈機指針向左，就是先對 1 號 stack 做 pop，塞入 TM 所寫入的字，再從 2 號 stack pop 字元塞入 1 號 stack
若是 2 號 stack 已經到底了，代表圖靈機跑到左邊的端點，上述的 pop 動作就不會執行

HW#7 Problem 7

想像一台圖靈機現在處於這樣的狀態

L	O	V	E	␣	␣	···
	↑					

那麼 2-PDA 就可能（依照處理過程不同，1 號 stack 的底部可能有更多東西）是這樣：

1 號 stack	2 號 stack
↓	↓
V	O
E	L
$	$

現在的指針正指在 O 的位置
假設我們想要做一個 $O \rightarrow I, R$ 的操作
在 TM 上的結果呈現會是：

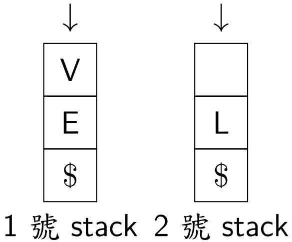
L	I	V	E	␣	␣	⋯

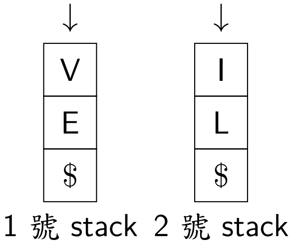
$\uparrow$ (under V)

而想要用 2-PDA 呈現，其步驟會是：

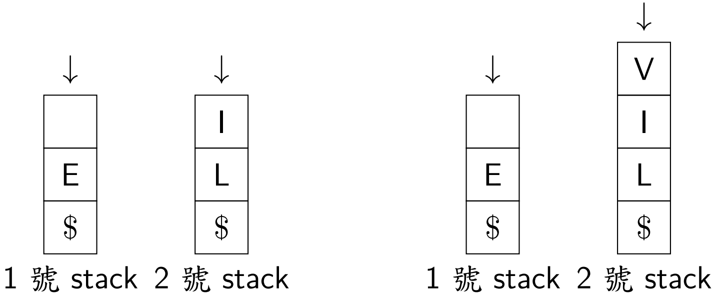
1. 把 O 從 stack 2 pop 出來
2. 把 I push 進 stack 2
3. 對指針做操作：把 V 從 stack 1 pop 出來
4. 把 V push 進 stack 2

HW#7 Problem 7





HW#7 Problem 7



HW#7 Problem 7

因為 3-tape TM 可以模擬 2-PDA，2-PDA 可以模擬 TM，而 3-tape TM 與 TM 的能力一樣，結論就是 2-PDA 與圖靈機的辨識能力一樣

而這些事情代入到 3-PDA 也能成立（4-tape TM 模擬 3-PDA、3-PDA 用其中兩個 stack 就能模擬 TM）

所以 3-PDA 與 2-PDA 的辨識能力都與圖靈機相同，兩者能力相等

HW#8 Problem 1

(10 points) Give a formal definition (with a state diagram) of a Turing machine that shifts the input string one tape cell to the right and put a $\sqcup$ (blank symbol) in front of the input. The input alphabet is $\{0, 1\}$.

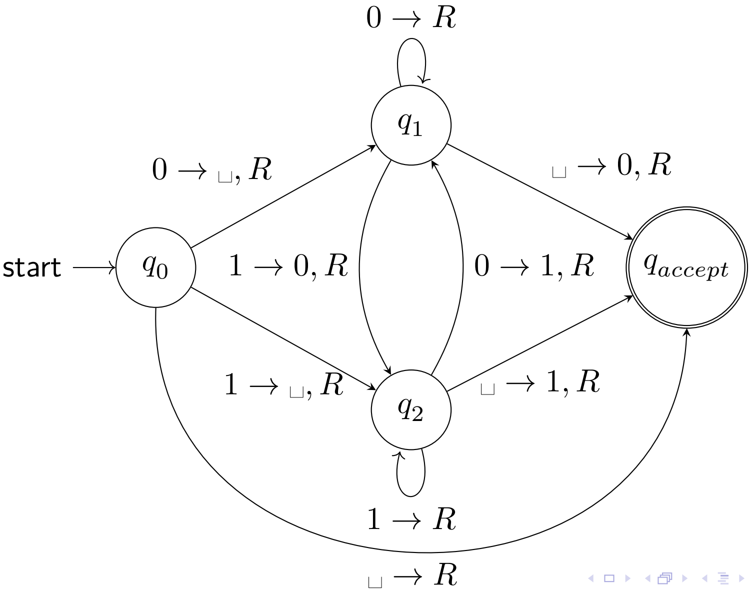
HW#8 Problem 1

The Turing machine TM for the problem is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where

- $Q = \{q_0, q_1, q_2, q_{accept}, q_{reject}\}$,
- $\Sigma = \{0, 1\}$,
- $\Gamma = \{0, 1, \sqcup\}$,
- q_0 is the initial state,
- q_{accept} is the accept state,
- q_{reject} is the reject state, and

HW#8 Problem 1

- $\delta =$



HW#8 Problem 2

(Exercise 3.4; 10 points) Give a formal definition of an enumerator (like that of an NFA, PDA, or Turing machine). Consider it to be a type of two-tape Turing machine that uses its second tape as the printer. Include a definition of the enumerated language.

HW#8 Problem 2

An enumerator is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{print}, q_{reject})$, where Q, Σ, Γ are all finite sets and

- Q is the set of states,
- Σ is the output alphabet, where the *blank* symbol $\sqcup \notin \Sigma$,
- Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\} \times (\Sigma \cup \{\epsilon, \#\})$ is the transition function,
- $q_0 \in Q$ is the initial state,
- $q_{print} \in Q$ is the print state, and
- $q_{reject} \in Q$ is the reject state.

HW#8 Problem 2

When we are in state q_{print}, and the content of second tape (printer) is $\omega = \omega_1 \# \omega_2 \# \cdots \omega_n \# \sqcup \cdots$, where $\omega_i \in \Sigma^*$ for $1 \leq i \leq n$, the collection of all possible occurrences of ω_i in ω is called enumerated language, namely the language enumerated by the enumerator.

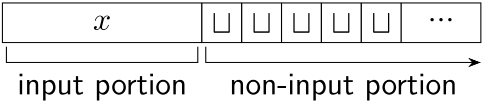
HW#8 Problem 3

(Problem 3.11; 20 points) Show that single-tape TMs that cannot write on the portion of the tape containing the input string recognize only regular languages.

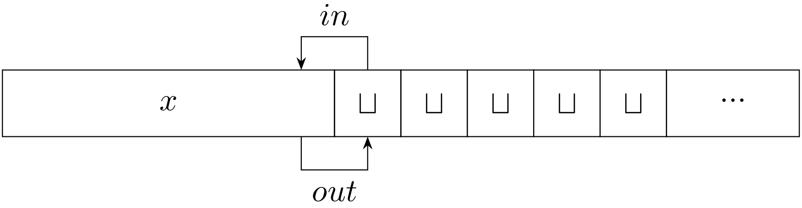
HW#8 Problem 3

Let $M = (Q, \Sigma, \Gamma, q_0, q_{accept}, q_{reject})$ be a single-tape TM that cannot write on the input portion of the tap. A typical case when M works on an input string x is as follows:

the tape head will stay in the input portion for some time, and then enter the non-input portion (i.e., the portion of the tape on the right of the $|x|^{th}$ cells) and stay there for some time, then go back to the input portion, and stay there for some time, and then enter the non-input portion, and so on.



HW#8 Problem 3



We call the event that the tape head switches from input portion to non-input portion an out event, and the event that the tape head switches from non-input portion to input-portion an in event.

HW#8 Problem 3

Let $first_x$ denote the state that M is in just after its first "out" event (i.e., the state of M when it first enters the non-input portion).

In case M never enters the non-input portion, we assign $first_x = q_{accept}$ if M accepts x, and assign $first_x = q_{reject}$ if M does not accept x.

HW#8 Problem 3

Next, we define a characteristic function f_x such that for any $q \in Q$, $f_x(q) = q'$ implies that if M is at state q just after its "in" event, M will move to state q' after its next "out" event.

In case M never enters the non-input portion again, we assign $f_x(q) = q_{accept}$ if M enters the accept state inside the input portion, and q_{reject} otherwise.

HW#8 Problem 3

Now we can define the binary relation R_L over Σ^* for the language L of TM M as follows:

$x\ R_L\ y$ iff

- $first_x = first_y$, and
- for all q, $f_x(q) = f_y(q)$.

We can observe the following property (requirements for Myhill-Nerode Theorem):

$x\ R_L\ y$ iff x and y are indistinguishable by L
(namely, $x\ R_L\ y$ iff $\forall z \in \Sigma^*(xz \in L \leftrightarrow yz \in L)$)

$Why?$

HW#8 Problem 3

x	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

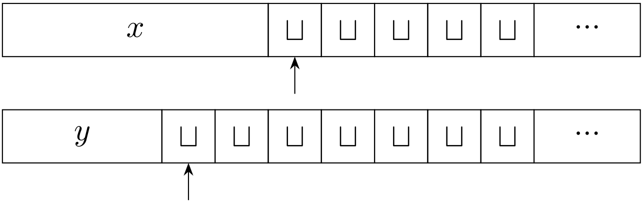
y	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Let we consider two strings x and y with the same $first$ and f:

$Situation\ 1$:

If $first_x = first_y = (q_{accept}$ or $q_{reject})$, x and y will both be accepted or rejected at the same time before "out" event happens.

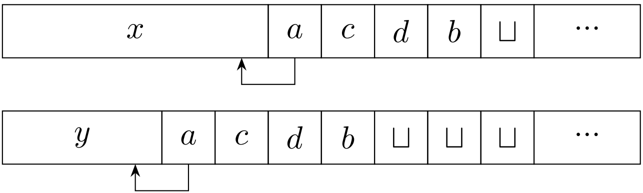
HW#8 Problem 3



$Situation$ 2:

If $first_x = first_y = q \neq (q_{accept}$ or $q_{reject})$, M_x and M_y will stay in the same state q and the heads of them stay in the same position of empty portion of two tapes ,which means that M_x and M_y will take the same actions in this portion (write the same symbol and move to the same state, i.e. if M_x accepts, M_y accepts at the same time).

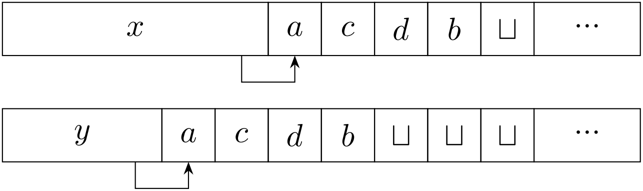
HW#8 Problem 3



Situation 2 (*cont.*):
How about "in" event happens?

Situation 2-1:
Because for all q, $f_x(q) = f_y(q)$, and M_x and M_y stay at the same state q when they are about to perform the "in" event, if $f_x(q) = f_y(q) = (q_{accept}$ or $q_{reject})$, similarly, x and y will both be accepted or rejected at the same time inside the input portion.

HW#8 Problem 3



Situation 2-2:
If $f_x(q) = f_y(q) = q' \neq (q_{accept}$ or $q_{reject})$, M_x and M_y will stay in the same state q' and the heads of them stay in the same position of non-input portion of two tapes (not empty now, but with the same string). Similarly, M_x and M_y will take the same actions in this portion.

If "in" event happens again, *Situation* 2 will happen repeatedly until M_x and M_y accept or reject.

HW#8 Problem 3

x	z	$\sqcup$	$\cdots$

y	z	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Now consider the strings xz and yz, you may notice that it is similar to $Situation$ 2-2, the non-input portion is not empty doesn't affect M_x and M_y to take the same actions in this portion.

So, M accepts xz if and only if M accepts yz, i.e. x and y are indistinguishable by M.

HW#8 Problem 3

In this situation, we say that x and y are in the same equivalence class (all strings in an equivalence class are indistinguishable to each other).

How many possibilities are there at most for the equivalence classes of M?

- $first_x$ has $|Q|$ possibilities.
- $f_x(q)$ has $|Q|$ possibilities for each $q \in Q$, i.e. $|Q|^{|Q|}$ possibilities totally.

So, there are at most $|Q|^{|Q|+1}$ equivalence classes, that is, the number of distinguishable strings are finite (R_L is of finite index). By Myhill-Nerode theorem, the language L is regular.

HW#8 Problem 4

(Problem 3.13; 20 points) Show that a language is decidable iff some enumerator enumerates the language in the standard string order (the usual lexicographical order, except that shorter strings precede longer strings) .

HW#8 Problem 4

$Proof$: if a language is decidable, there's an enumerator enumerates the language in the standard string order.

Let D be the decider that decides the language A and Σ is the alphabet of A, we can construct an enumerator E as follows:

Because Σ^* is countable, E can pick string s from Σ^* in a specific order and run D on s. If D has accepted, print s out and pick the next string; otherwise, do nothing and pick the next string directly.

HW#8 Problem 4

$Proof$: if there's an enumerator enumerates a language in the standard string order, the language is decidable.

Let E be the enumerator that enumerates the language A in the standard string order, we can construct a decider D on input string s as follows:

Run E, when E's turn to print s (will be in finite turns), if E prints s, $accept$; otherwise, $reject$.(即判斷當順序走到要印 s 的時候是否有印出 s，如果印出了 s 順序後面的字串卻沒有印出 s，代表 s 被跳過了！)

HW#8 Problem 5

(Exercise 4.3; 10 points) Let $ALL_{\text{DFA}} = \{\langle A \rangle \mid A \text{ is a DFA and } L(A) = \Sigma^*\}$. Show that ALL_{DFA} is decidable.

HW#8 Problem 5

We can construct a decider D as follows:

$D =$ "On input $\langle A \rangle$, where A is a DFA:

1. Mark the initial state of A.
2. Mark the states of A that can be arrived from any marked states.
3. Repeat step 2 until no state can be marked.
4. If there is any non-accepting state marked, $reject$; otherwise, $accept$."

HW#8 Problem 5

Reduction method:

Let TM T decides E_{DFA}, we can construct a decider D as follows:

$D =$ "On input $\langle A \rangle$, where A is a DFA:
1. Construct the complement $\overline{A}$ of A.
2. Run T on input $\langle \overline{A} \rangle$.
3. If T accepts, *accept*; otherwise, *reject*."

HW#8 Problem 6

(20 points) Let $A = \{\langle M, N\rangle \mid M \text{ is a PDA and } N \text{ is a DFA such that } L(M) \subseteq L(N)\}$. Show that A is decidable.

HW#8 Problem 6

Use the property: $A \subseteq B \Leftrightarrow A \cap \overline{B} = \emptyset$.
Let TM R decides E_{CFG}, we can construct a decider D as follows:

$D =$ "On input $\langle M, N \rangle$, where M is a PDA and N is a DFA:
1. Construct the complement $\overline{N}$ of N.
2. Construct a PDA P that recognizes the intersection of M and $\overline{N}$ (the intersection of a context-free language and a regular language is context free).
3. Let G_P be the context-free grammar that recognized by P, run R on input $\langle G_P \rangle$.
4. If R accepts, $accept$; otherwise, $reject$."

HW#8 Problem 7

(Problem 4.4; 10 points) Let $A\varepsilon_{\text{CFG}} = \{\langle G \rangle \mid G \text{ is a CFG that generates } \varepsilon\}$. Show that $A\varepsilon_{\text{CFG}}$ is decidable.

HW#8 Problem 7

We can construct a decider D as follows:

$D =$ "On input $\langle G \rangle$, where G is a CFG:
1. Convert G to an equivalent grammar in Chomsky normal form G'.
2. If $(S_0 \to \epsilon) \in G'$, *accept* (in Chomsky normal form, only S_0 can generate ϵ); otherwise, *reject*."

HW#8 Problem 7

Reduction method:

Let TM S decides A_{CFG}, we can construct a decider D as follows:

$D =$ "On input $\langle G \rangle$, where G is a CFG:
1. Run S on input $\langle G, \epsilon \rangle$.
2. If S accepts, $accept$; otherwise, $reject$."

HW#9 Problem 1

(Exercise 4.9; 10 points) Review the way that we define sets to be of the same size in Definition 4.12. Show that "are of the same size" is an equivalence relation.

HW#9 Problem 1

We have the definition that if there is a *one-one correspondence* between two sets A and B, A and B are considered to have the same size.

To prove that "A and B are of the same size" is an equivalence relation, we need to prove the following properties:

- Reflexive
- Symmetric
- Transitive

HW#9 Problem 1

Reflexive:

Trivial. We can construct a $function\ f_A$ according to the following rule: $f_A(a) = a$, where $a \in A$.
Obviously, f_A is a correspondence.

HW#9 Problem 1

Symmetric:

Let $f_{AB} : A \to B$ be a $function$, which is also a $correspondence$. We can construct a $relation$ $f_{BA} : B \to A$ defined by the following rule: $f_{BA}(b) = a$ if $f_{AB}(a) = b$. We can prove that f_{BA} is a $function$ and is also a $correspondence$:

- f_{BA} is a $function$: for all $b \in B$, $f_{BA}(b)$ has at least one output $a \in A$ (f_{AB} is *onto*) and at most one output $a \in A$ (f_{AB} is *one-to-one*). Hence for all $b \in B$, $f_{BA}(b)$ has exactly one corresponding output $a \in A$.
- f_{BA} is *one-to-one*: if f_{BA} is not *one-to-one*, $f_{AB}(a)$ may have two or more possible outputs, then f_{AB} would not be a $function$.
- f_{BA} is *onto*: because f_{AB} is a $function$, all $a \in A$ have one corresponding $f(a) \in B$.

HW#9 Problem 1

Transitive:

Let $f_{AB} : A \to B, f_{BC} : B \to C$ be two $functions$, which are also $correspondence$s. We can construct a $relation$ $f_{AC} : A \to C$ defined by the following rule: $f_{AC}(a) = f_{BC}(f_{AB}(a))$. We can prove that f_{AC} is a $function$ and is also a $correspondence$:

- f_{AC} is a $function$: for all input $a \in A$ of f_{AC}, we can obtain a fixed output $b \in B$ through $f_{AB}(a)$ and a fixed output $c \in C$ through $f_{BC}(b)$. Hence, for all input $a \in A, f_{AC}(a)$ has a fixed output $c \in C$.
- f_{AC} is *one-to-one*: if $x \neq y$, $f_{AB}(x) \neq f_{AB}(y)$ because f_{AB} is *one-to-one*, and $f_{BC}(f_{AB}(x)) \neq f_{BC}(f_{AB}(y))$ because f_{BC} is *one-to-one*.
- f_{AC} is *onto*: for all $c \in C$, there is an $b \in B$ such that $f_{BC}(b) = c$ (f_{BC} is *onto*), and for all $b \in B$, there is an $a \in A$ such that $f_{AB}(a) = b$ (f_{AB} is *onto*). So for all $c \in C$, there is an $a \in A$ such that $f_{AC}(a) = c$.

HW#9 Problem 2

(Problem 4.12; 10 points) Let A be a Turing-recognizable language consisting of descriptions of Turing machines, $\{\langle M_1\rangle, \langle M_2\rangle, \ldots\}$, where every M_i is a decider. Prove that some decidable language D is not decided by any decider M_i whose description appears in A. (Hint: you may find it helpful to consider an enumerator for A.)

HW#9 Problem 2

A 是 Turing-recognizable language，包含了某些 Deciders
說明必然存在一個 decidable language D，它不能被 A 裡頭的任何 Decider 給 decide

HW#9 Problem 2

使用對角論證法：

題目提示告訴我們，既然 A 是 Turing-recognizable，就表示有一個 Enumerator E 可以生成 A
將 E 生成的第 i 個 TM 標記為 M_i
而因為 Σ^* 是可數集，存在一種排序法使對於任一個字串 $s \in \Sigma^*$ 而言，都能標記它出現的順序
於是可以做出一張表

HW#9 Problem 2

	s_1	s_2	...	s_i	...
M_1	accept	accept	...	reject	...
M_2	accept	reject	...	accept	...
$\vdots$	...	...	...	...	...
M_i	reject	accept	...	reject	...
$\vdots$	...	...	...	...	...

依照這張表，建構一個 TM M_D recognize D

M_D = "On input s:

1. 計算出 s 在 Σ^* 當中的順位 i
2. 將 s 丟入 M_i 當中計算
3. If M_i accepts, reject; otherwise, accept."

這樣就能建構出一台與 A 當中的任何圖靈機都不一樣的機器

而且 M_i 本身是 Decider，這台機器一定會停機，所以 M_D 是 Decider，D 是 decidable language

得證，存在一個 D 不能被 A 當中的任何 Decider 給判定

HW#9 Problem 2

這題能告訴我們什麼
一個存著「所有」Deciders 的語言
$D_{ALL} = \{\langle D \rangle \mid \text{D decides a language over } \Sigma^*\}$ 不可能是 Turing-recognizable

HW#9 Problem 3

(Problem 4.14; 20 points) Let $C = \{\langle G, x\rangle \mid G$ is a CFG and x is a substring of some $y \in L(G)\}$. Show that C is decidable. (Hint: an elegant solution to this problem uses the decider for E_{CFG}.)

HW#9 Problem 3

存在一個 decider，可以判斷 CFG G 是否會生成某個字串 y 使得 x 是它的子字串

那麼就是要把 L(G) 與 $\Sigma^* x \Sigma^*$ 這兩個 language 取交集
一個 CFL 與 RL 的交集也是 CFL（將 PDA 與 DFA 的狀態合在一起做成新的 PDA）
再把交集出來的語言丟到 E_{CFG} 的 Decider 裡頭即可

HW#9 Problem 3

M = "On input $\langle G, x\rangle$ where G is a CFG:

1. Construct a CFG G' s.t. $L(G') = L(G) \cap \Sigma^* x \Sigma^*$
2. Run $M_{E_{CFG}}$ on input $\langle G' \rangle$
3. If $M_{E_{CFG}}$ accept, reject; otherwise, accept."

上面每個步驟都能在有限時間完成，所以得 C 是 decidable

HW#9 Problem 4

4. (Problem 4.18; 20 points) A *useless state* in a pushdown automaton is never entered on any input string. Consider the problem of determining whether a pushdown automaton has any useless states. Formulate this problem as a language and show that it is decidable.

HW#9 Problem 4

如何偵測 PDA 裡頭的 useless state

將所有狀態都變成 nonaccepting，然後再單獨把一個 state 畫成 accepting

如果此時 PDA recognize 的語言為空，則代表這個 state 完全不會被抵達

HW#9 Problem 4

M = "On input $\langle P \rangle$, P is a PDA:

1. 將 PDA 所有狀態都變成 nonaccepting
2. Choose one state to be accepting
3. Convert this PDA into CFG G
4. Run $M_{E_{CFG}}$ on input $\langle G \rangle$
5. Repeat step 2 to 4
6. If $M_{E_{CFG}}$ has ever accepted, accept; otherwise, reject."

HW#9 Problem 5

5. (Problem 4.31; 20 points) Let $INFINITE_{\text{PDA}} = \{\langle M\rangle \mid M \text{ is a PDA and } L(M) \text{ is infinite}\}$. Show that $INFINITE_{\text{PDA}}$ is decidable.

HW#9 Problem 5

判定 PDA 辨識的字串是否有無限多個
由 pumping lemma 可以知道，只要 CFL 內有個字串 s 長度有 pumping length p 以上，就可以生成無限多個字串也在 CFL 內
而且此時一定會有一個長度介於 p 與 2p 之間的字串：
如果 $p \leq |s| \leq 2p$ 那就有了

HW#9 Problem 5

Y = "On input $\langle M \rangle$ where M is a PDA:

1. Convert M to a CFG G and compute G's pumping length p.
2. Construct a regular expression E that contains all strings of length p or more.
3. Construct a CFG H such that $L(H) = L(G) \cap L(E)$
4. Test $L(H) = \emptyset$, using the E_{CFG} decider R.
5. If R accepts, reject; if R rejects, accepts. "

HW#9 Problem 6

6. (Exercise 5.4; 20 points) If A is reducible to B and B is a regular language, does that imply that A is a regular language? Why or why not?

HW#9 Problem 6

若 A 能 reduce 成一個正規語言 B，那 A 是不是正規的呢？

假設 A 是一個 CFL，而 B = {1}
試著找到 f 使得 $w \in A \iff f(w) \in B$
假設 A 對應的 CFG 為 G
F = “On input w:
1. Run $M_{A_{CFG}}$ on input $\langle G, w \rangle$
2. If $M_{A_{CFG}}$ accepts, output 1; otherwise, output 0”
因為 A_{CFG} 是 decidable，所以 f 是 computable
如此可知，雖然 A 可以 reduce 成正規語言 B，但 A 並不見得是正規的

HW#10 Problem 1

(Problem 5.9; 10 points) Let $AMBIG_{\mathrm{CFG}} = \{\langle G\rangle \mid G \text{ is an ambiguous CFG}\}$. Show that $AMBIG_{\mathrm{CFG}}$ is undecidable. (Hint: use a reduction from PCP. Given an instance

$$P = \left\{ \left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \ldots, \left[\frac{t_k}{b_k}\right] \right\}$$

of PCP, construct a CFG G with the rules:

$$\begin{aligned} S &\rightarrow T \mid B \\ T &\rightarrow t_1 T a_1 \mid \cdots \mid t_k T a_k \mid t_1 a_1 \mid \cdots \mid t_k a_k \\ B &\rightarrow t_1 B a_1 \mid \cdots \mid t_k B a_k \mid t_1 a_1 \mid \cdots \mid t_k a_k, \end{aligned}$$

where $a_1, \ldots, a_k$ are new terminal symbols. Prove that this reduction works.)

HW#10 Problem 1

Assume that a TM D_{AMBIG} decides $AMBIG_{\text{CFG}}$, we can construct a decider D that decides PCP as follows:

$D =$ "On input $\langle P \rangle$, where $P = \left\{ \left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \cdots, \left[\frac{t_k}{b_k}\right] \right\}$:

1. Construct a CFG G with the rules:

$$S \to T \mid B$$
$$T \to t_1 T a_1 \mid \cdots \mid t_k T a_k \mid t_1 a_1 \mid \cdots \mid t_k a_k$$
$$B \to b_1 B a_1 \mid \cdots \mid b_k B a_k \mid b_1 a_1 \mid \cdots \mid b_k a_k$$

2. Run D_{AMBIG} on input $\langle G \rangle$.
3. If D_{AMBIG} accepts, *accept*; otherwise, *reject*."

But we've known that PCP is undecidable, so $AMBIG_{\text{CFG}}$ is undecidable.

HW#10 Problem 2

(Problem 5.14(b); 20 points) Define a *two-headed finite automaton* (2DFA) to be a deterministic finite automaton that has two read-only, bidirectional heads that start at the left-hand end of the input tape and can be independently controlled to move in either direction. The tape of a 2DFA is finite and is just large enough to contain the input plus two additional blank tape cells, one on the left-end and one on the right-hand end, that serve as delimiters. A 2DFA accepts its input by entering a special accept state. For example, a 2DFA can recognize the language $\{a^n b^n c^n \mid n \geq 0\}$.

Let $E_{\text{2DFA}} = \{\langle M \rangle \mid M \text{ is a 2DFA and } L(M) = \emptyset\}$. Show that E_{2DFA} is undecidable.

HW#10 Problem 2

We can reduce E_{TM} to $E_{2\text{DFA}}$.

The idea is to construct a 2DFA that recognizes the accept computational history of a TM M.

To do so, the 2DFA needs to check if the first and the last configurations are the starting configuration and the accepting configuration and then check for each transition whether it is valid in M.

It is able to do this task because with the two heads we can compare the configurations without writing anything (just like how it recognizes the language $\{a^n b^n c^n \mid n \geq 0\}$).

HW#10 Problem 2

Assume that a TM $D_{2\text{DFA}}$ decides $E_{2\text{DFA}}$, we can construct a decider D that decides E_{TM} as follows:

$D =$ "On input $\langle M \rangle$, where M is a TM:
1. Construct a 2DFA N from M as described in previous slide.
2. Run $D_{2\text{DFA}}$ on input $\langle N \rangle$.
3. If $D_{2\text{DFA}}$ accepts, *accept*; otherwise, *reject*."

But we've known that E_{TM} is undecidable, so $E_{2\text{DFA}}$ is undecidable.

HW#10 Problem 3

(Problem 5.18(b); 10 points) Use Rice's theorem to prove the undecidability of the language $\{\langle M \rangle \mid M \text{ is a TM and } 101 \in L(M)\}$. (Note: you should show that Rice's theorem is applicable for the problem/language.)

HW#10 Problem 3

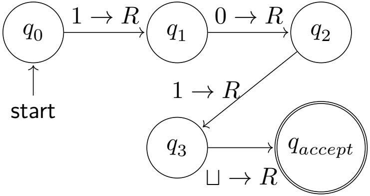
When we use Rice's Theorem to prove the decidability of a language, we need to confirm if the property is nontrivial.

$\{M$ is a $\text{TM}\}$ is the input constraint (we do not need to discuss), so we only need to consider the property $\{101 \in L(M)\}$.

HW#10 Problem 3

$\{101 \in L(M)\}$ is obviously an nontrivial property because there must exist some TM that recognizes the string 101 and some do not.

e.q. $\Sigma = \{0, 1\}$, the left TM recognizes the string 101 but the right one does not:



So, by Rice's Theorem we can prove that the language $\{\langle M \rangle \mid M$ is a TM and $101 \in L(M)\}$ is undecidable.

HW#10 Problem 4

(Problem 5.22; 20 points) Let $X = \{\langle M, w\rangle \mid M$ is a single-tape TM that never modifies the portion of the tape that contains the input $w\}$. Is X decidable? Prove your answer.

HW#10 Problem 4

We can try to reduce A_{TM} to X.

Assume that a TM D_X decides X, we can construct a decider D that decides A_{TM} as follows:

$D =$ "On input $\langle M, w\rangle$, where M is a TM and w is a string:

1. Construct $M' =$ "On input u:
    1. Move to the right of u and put \$.
    2. Copy w after \$.
    3. Simulate M on the portion of w.
    4. If M accepts and u is not empty, modify any character of u and *accept*; otherwise, *reject*."
2. Run D_X on input $\langle M', u\rangle$ for any non-empty string u.
3. If D_X accepts, *reject*; otherwise, *accepts*."

But we've known that A_{TM} is undecidable, so X is undecidable.

HW#10 Problem 5

(20 points) Prove that $HALT_{\mathrm{TM}} \leq_m \overline{E_{\mathrm{TM}}}$, where $HALT_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on } w\}$ and $E_{\mathrm{TM}} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$.

HW#10 Problem 5

We will construct a computable function f (as defined by F below) such that

$$\langle M, w\rangle \in HALT_{\mathrm{TM}} \Leftrightarrow f(\langle M, w\rangle) \in \overline{E_{\mathrm{TM}}}.$$

F = "On input $\langle M, w\rangle$:

1. Construct the following machine M'.

   M' = "On input x:

   1. If $x \neq w$, *reject*.
   2. If $x = w$, run M on input x.
   3. If M halts, *accepts*; otherwise, *reject*."

2. Output $\langle M'\rangle$."

HW#10 Problem 6

(10 points) Let $ALL_{\text{DFA}} = \{\langle A \rangle \mid A \text{ is a DFA and } L(A) = \Sigma^*\}$. Prove that $ALL_{\text{DFA}} \in \text{P}$.

HW#10 Problem 6

We can construct a deterministic single-tape decider D that decides ALL_{DFA} in polynomial time as follows:

$D =$ "On input $\langle A \rangle$, where A is a DFA with n states:

$(O(1))$ 1. Mark the initial state of A.

$(O(|Q|^n))$ 2. Mark the states of A that can be arrived from any marked states until no state can be marked.

$(O(|Q|))$ 3. If there is any non-accepting state marked, $reject$; otherwise, $accepts$."

The decider D will decide ALL_{DFA} in $(O(|Q|^n))$, so $ALL_{\text{DFA}} \in P$.

HW#10 Problem 7

(10 points) Two graphs G and H are said to be *isomorphic* if the nodes of G may be renamed so that it becomes identical to H. Let $ISO = \{\langle G, H\rangle \mid G \text{ and } H \text{ are isomorphic}\}$. Prove that $ISO \in \text{NP}$, using the definition $\text{NP} = \bigcup_k \text{NTIME}(n^k)$.

HW#10 Problem 7

We can construct an nondeterministic polynomial time decider N decides ISO as follows:

$N =$ "On input $\langle G, H \rangle$ where G and H are undirected graphs:
1. Let m be the number of nodes of G and H. If they don' t have the same number of nodes, $reject$.
2. Nondeterministically select a permutation π of m elements.
3. For each pair of nodes x and y of G check that (x, y) is an edge of G iff $(\pi(x), \pi(y))$ is an edge of H. If all agree, $accepts$. If any differ, $reject$.

Stage 2 can be implemented in polynomial time nondeterministically. Stages 1 and 3 takes polynomial time. Hence ISO $\in$ NP.