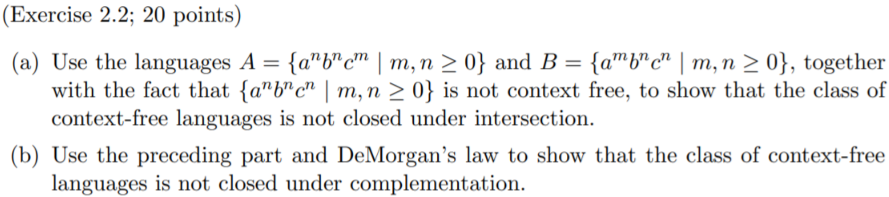
Homework 6 - 10

劉韋成 蘇俊杰

Menu

1. Hw6
   - 1
   - 2
   - 3
   - 4
   - 5
   - 6
2. Hw7
   - 1
   - 2
   - 3
   - 4
   - 5
   - 6
   - 7

Hw6 Problem1



(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.

Hw6 Problem1 (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.

Hw6 Problem1 (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:

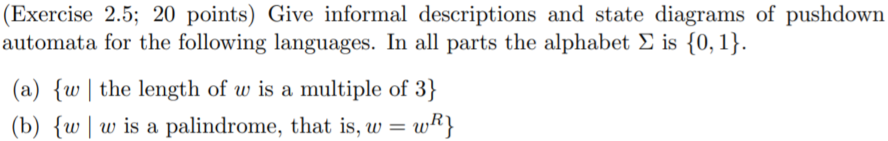
A and B are context free.
$\Rightarrow$ $\overline{A}$ and $\overline{B}$ are context free.
$\Rightarrow$ $\overline{A} \cup \overline{B}$ is context free.
$\Rightarrow$ $\overline{\overline{A} \cup \overline{B}}$ is context free.
$\Rightarrow$ $A \cap B$ is context free.

Hw6 Problem1 (b)

But 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.

Hw6 Problem2



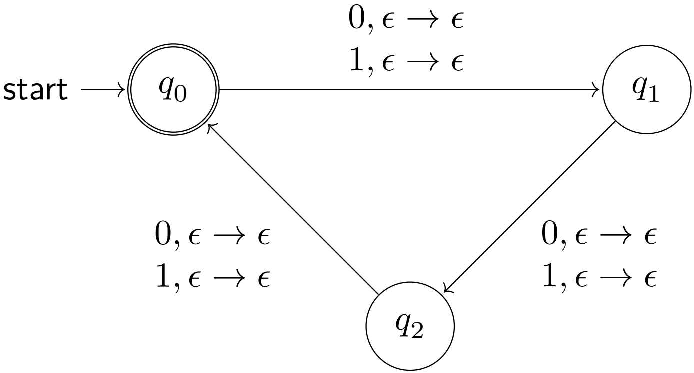
(Exercise 2.5; 20 points) Give informal descriptions and state diagrams of pushdown automata for the following languages. In all parts the alphabet Σ is $\{0, 1\}$.

(a) $\{w \mid \text{the length of } w \text{ is a multiple of } 3\}$

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

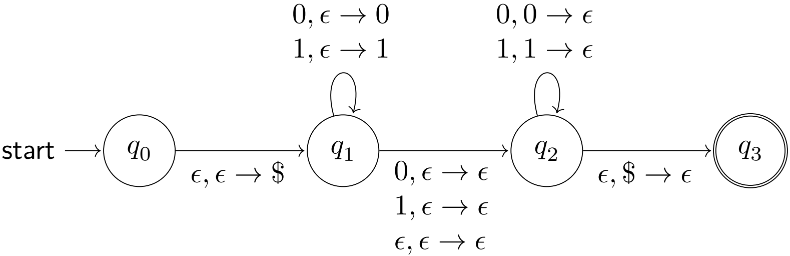
Hw6 Problem2 (a)

$\{w \mid$ the length of w is a multiple of 3$\}$


start
q_0
q_1
q_2
$0, \epsilon \to \epsilon$
$1, \epsilon \to \epsilon$
$0, \epsilon \to \epsilon$
$1, \epsilon \to \epsilon$
$0, \epsilon \to \epsilon$
$1, \epsilon \to \epsilon$

Hw6 Problem2 (b)

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

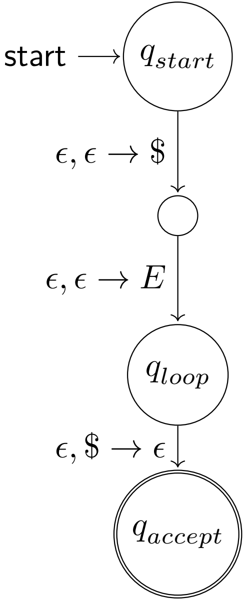

start
q_0
q_1
q_2
q_3
$\epsilon, \epsilon \rightarrow \$$
$0, \epsilon \rightarrow 0$
$1, \epsilon \rightarrow 1$
$0, \epsilon \rightarrow \epsilon$
$1, \epsilon \rightarrow \epsilon$
$\epsilon, \epsilon \rightarrow \epsilon$
$0, 0 \rightarrow \epsilon$
$1, 1 \rightarrow \epsilon$
$\epsilon, \$ \rightarrow \epsilon$

Hw6 Problem3

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

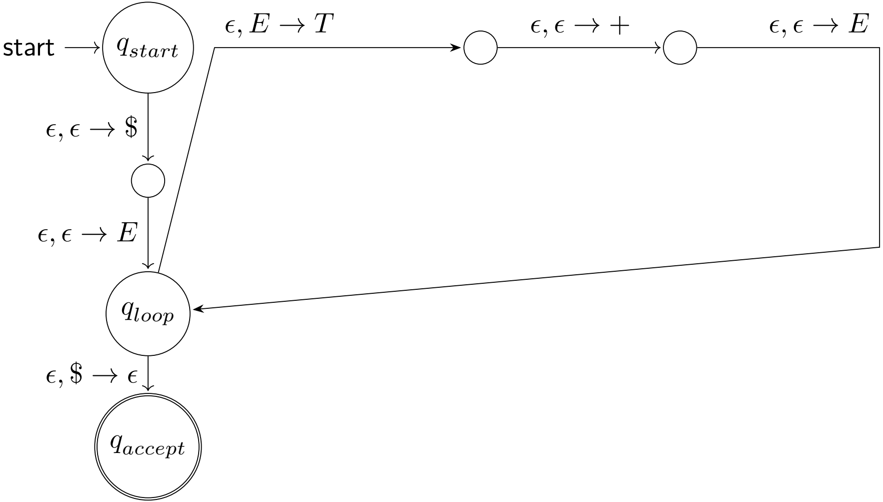
$$\begin{array}{rcl} E & \to & E + T \mid T \\ T & \to & T \times F \mid F \\ F & \to & (\ E\ ) \mid a \end{array}$$

Hw6 Problem3


start
q_{start}
$\epsilon, \epsilon \rightarrow \$$
$\epsilon, \epsilon \rightarrow E$
q_{loop}
$\epsilon, \$ \rightarrow \epsilon$
q_{accept}

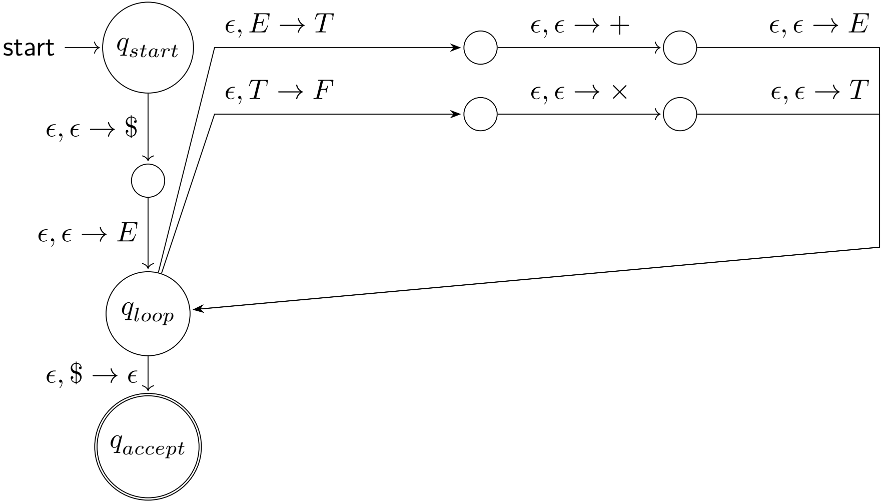
Hw6 Problem3

$E \rightarrow E + T$


start
q_{start}
$\epsilon, \epsilon \rightarrow \$$
$\epsilon, \epsilon \rightarrow E$
q_{loop}
$\epsilon, \$ \rightarrow \epsilon$
q_{accept}
$\epsilon, E \rightarrow T$
$\epsilon, \epsilon \rightarrow +$
$\epsilon, \epsilon \rightarrow E$

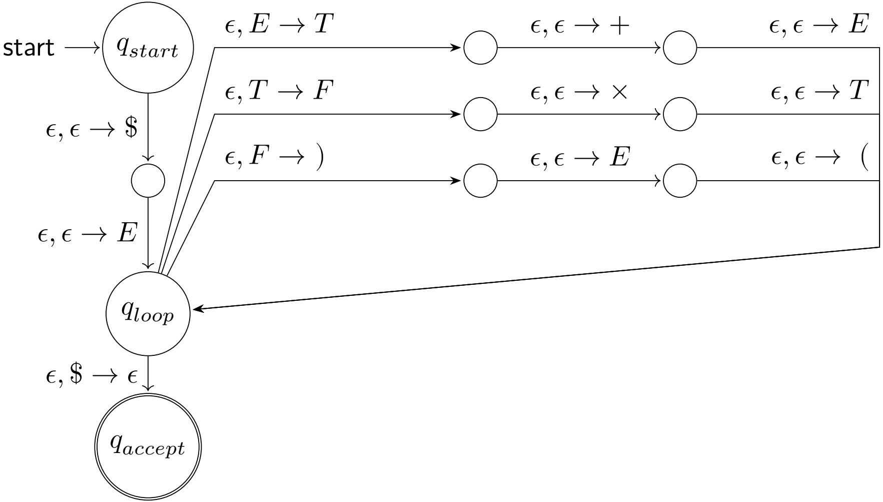
Hw6 Problem3

$T \rightarrow T \times F$


start
q_{start}
$\epsilon, \epsilon \rightarrow \$$
$\epsilon, \epsilon \rightarrow E$
q_{loop}
$\epsilon, \$ \rightarrow \epsilon$
q_{accept}
$\epsilon, E \rightarrow T$
$\epsilon, \epsilon \rightarrow +$
$\epsilon, \epsilon \rightarrow E$
$\epsilon, T \rightarrow F$
$\epsilon, \epsilon \rightarrow \times$
$\epsilon, \epsilon \rightarrow T$

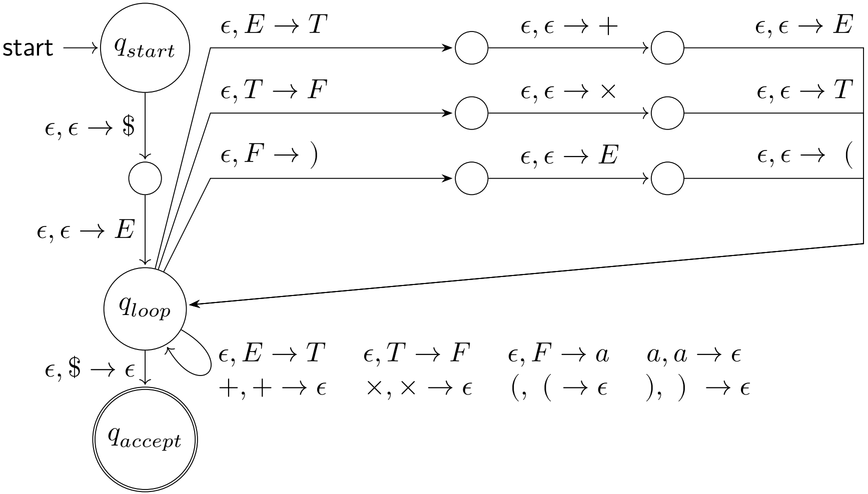
Hw6 Problem3

$F \rightarrow (E)$


start
q_{start}
$\epsilon, \epsilon \rightarrow \$$
$\epsilon, \epsilon \rightarrow E$
q_{loop}
$\epsilon, \$ \rightarrow \epsilon$
q_{accept}
$\epsilon, E \rightarrow T$
$\epsilon, \epsilon \rightarrow +$
$\epsilon, \epsilon \rightarrow E$
$\epsilon, T \rightarrow F$
$\epsilon, \epsilon \rightarrow \times$
$\epsilon, \epsilon \rightarrow T$
$\epsilon, F \rightarrow)$
$\epsilon, \epsilon \rightarrow E$
$\epsilon, \epsilon \rightarrow ($

Hw6 Problem3

Remaining grammar


start
q_{start}
$\epsilon, \epsilon \to \$$
$\epsilon, \epsilon \to E$
q_{loop}
$\epsilon, \$ \to \epsilon$
q_{accept}
$\epsilon, E \to T$
$\epsilon, \epsilon \to +$
$\epsilon, \epsilon \to E$
$\epsilon, T \to F$
$\epsilon, \epsilon \to \times$
$\epsilon, \epsilon \to T$
$\epsilon, F \to)$
$\epsilon, \epsilon \to E$
$\epsilon, \epsilon \to ($
$\epsilon, E \to T$ $\epsilon, T \to F$ $\epsilon, F \to a$ $a, a \to \epsilon$
$+, + \to \epsilon$ $\times, \times \to \epsilon$ $(, (\to \epsilon$ $),) \to \epsilon$

Hw6 Problem4

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

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

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

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

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.

Hw6 Problem4 (a)

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

There are two way to obtain this language:

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

Hw6 Problem4 (a)

2.

$\langle \text{STMT} \rangle$

$\Rightarrow \langle \text{IF-THEN-ELSE} \rangle$

$\Rightarrow$ `if condition then` $\langle \text{STMT} \rangle$ `else` $\langle \text{STMT} \rangle$

$\Rightarrow$ `if condition then` $\langle \text{IF-THEN} \rangle$ `else` $\langle \text{STMT} \rangle$

$\Rightarrow$ `if condition then if condition then` $\langle \text{STMT} \rangle$ `else` $\langle \text{STMT} \rangle$

$\Rightarrow$ `if condition then if condition then a:=1 else a:=1`

So G is ambiguous.

Hw6 Problem4 (b)

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

The problem of the original grammar G is that when $\langle \text{IF-THEN-ELSE} \rangle$ appears, we expect that the `if` and `else` in it should be matched, but the $\langle \text{STMT} \rangle$ 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 $\langle \text{IF-THEN-ELSE} \rangle$ should already be matched.

Hw6 Problem4 (b)

A new unambiguous grammar G':

$\langle\text{STMT}\rangle \rightarrow \langle\text{ASSIGN}\rangle \mid \langle\text{IF-THEN}\rangle \mid \langle\text{IF-THEN-ELSE}\rangle$

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

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

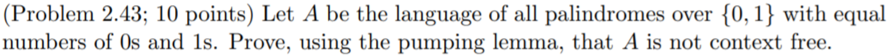
$\langle\text{STMT-M}\rangle \rightarrow \langle\text{ASSIGN}\rangle \mid \langle\text{IF-THEN-ELSE-M}\rangle$

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

$\langle\text{ASSIGN}\rangle \rightarrow$ `a:=1`

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

Hw6 Problem5



(Problem 2.43; 10 points) Let A be the language of all palindromes over $\{0, 1\}$ with equal numbers of 0s and 1s. Prove, using the pumping lemma, that A is not context free.

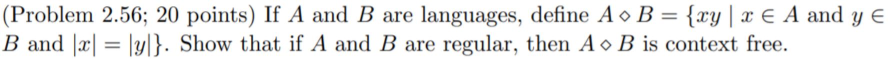
Hw6 Problem5

Let s be $0^p1^{2p}0^p$, where p is the pumping length.
Cases of dividing s as $uvxyz$ (where $|vy| > 0$ and $|vxy| \leq p$):

- if vxy are all 0s or 1s, uv^2xy^2z will make the number of 0s and 1s become unbalanced.
- if v are all 0s and y are all 1s, uv^2xy^2z will not be a palindrome.
- if v are all 1s and y are all 0s, uv^2xy^2z will not be a palindrome.
- if v are 0^i1^j and y are all 1s, uv^2xy^2z will not be a palindrome.
- if v are all 1s and y are 1^i0^j, uv^2xy^2z will not be a palindrome.

So A is not context free.

Hw6 Problem6



(Problem 2.56; 20 points) If A and B are languages, define $A \diamond B = \{xy \mid x \in A \text{ and } y \in B \text{ and } |x| = |y|\}$. Show that if A and B are regular, then $A \diamond B$ is context free.

Hw6 Problem6

Assume that $\mathcal{N}_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$ and $\mathcal{N}_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$ be two NFAs that recognize A and B separately.

We can construct a pushdown automaton $\mathcal{P} = (Q, \Sigma, \Gamma, \delta, q_0, F)$ that recognize $A \diamond B$ as follows:

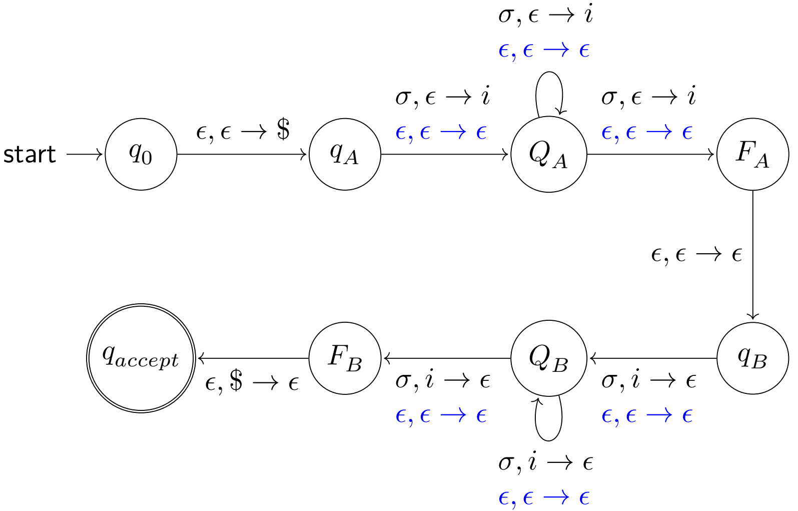
Hw6 Problem6

- $Q = Q_A \cup Q_B \cup \{q_0, q_{accept}\}$,
- $\Gamma = \{\$, i\}$,
- $$\delta((q,\gamma),\sigma) = \begin{cases} (q_A, \$) & \text{if } q = q_0 \text{ and } \gamma = \epsilon \text{ and } \sigma = \epsilon \\ (\delta_A(q,\sigma), i) & \text{if } q \in Q_A \text{ and } \gamma = \epsilon \text{ and } \sigma \neq \epsilon \\ (\delta_A(q,\sigma), \epsilon) & \text{if } q \in Q_A \text{ and } \gamma = \epsilon \text{ and } \sigma = \epsilon \\ (q_B, \epsilon) & \text{if } q \in F_A \text{ and } \gamma = \epsilon \text{ and } \sigma = \epsilon \\ (\delta_B(q,\sigma), \epsilon) & \text{if } q \in Q_B \text{ and } \gamma = i \text{ and } \sigma \neq \epsilon \\ (\delta_B(q,\sigma), \epsilon) & \text{if } q \in Q_B \text{ and } \gamma = \epsilon \text{ and } \sigma = \epsilon \\ (q_{accept}, \epsilon) & \text{if } q \in F_B \text{ and } \gamma = \$ \text{ and } \sigma = \epsilon \end{cases}$$
- q_0 is the start state, and
- $F = \{q_{accept}\}$ is the set of accept states.

(The blue part can be omitted if you assume that $\mathcal{N}_A$ and $\mathcal{N}_B$ are DFAs.)

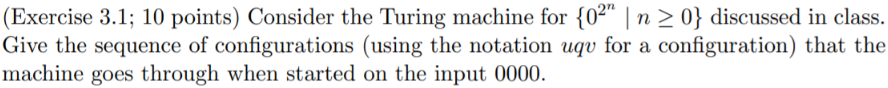
Hw6 Problem6

Schematic state diagram (not real PDA):



start
q_0
$\epsilon, \epsilon \to \$$
q_A
$\sigma, \epsilon \to i$
$\epsilon, \epsilon \to \epsilon$
Q_A
$\sigma, \epsilon \to i$
$\epsilon, \epsilon \to \epsilon$
$\sigma, \epsilon \to i$
$\epsilon, \epsilon \to \epsilon$
F_A
$\epsilon, \epsilon \to \epsilon$
q_B
$\sigma, i \to \epsilon$
$\epsilon, \epsilon \to \epsilon$
Q_B
$\sigma, i \to \epsilon$
$\epsilon, \epsilon \to \epsilon$
$\sigma, i \to \epsilon$
$\epsilon, \epsilon \to \epsilon$
F_B
$\epsilon, \$ \to \epsilon$
q_{accept}

Hw7 Problem1



(Exercise 3.1; 10 points) Consider the Turing machine for $\{0^{2^n} \mid n \geq 0\}$ 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 0000.

Hw7 Problem1

$q_1 0000$

$\sqcup q_2 000$

$\sqcup x q_3 00$

$\sqcup x 0 q_4 0$

$\sqcup x 0 x q_3$

$\sqcup x 0 q_5 x$

$\sqcup x q_5 0 x$

$\sqcup q_5 x 0 x$

$\sqcup q_2 x 0 x$

$\sqcup x q_2 0 x$

$\sqcup x x q_3 x$

$\sqcup x x x q_3$

$\sqcup x x q_5 x$

$\sqcup x q_5 x x$

$\sqcup q_5 x x x$

$q_5 \sqcup x x x$

$\sqcup q_2 x x x$

$\sqcup x q_2 x x$

$\sqcup x x q_2 x$

$\sqcup x x x q_2$

$\sqcup x x x \sqcup q_{accept}$

Hw7 Problem2

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

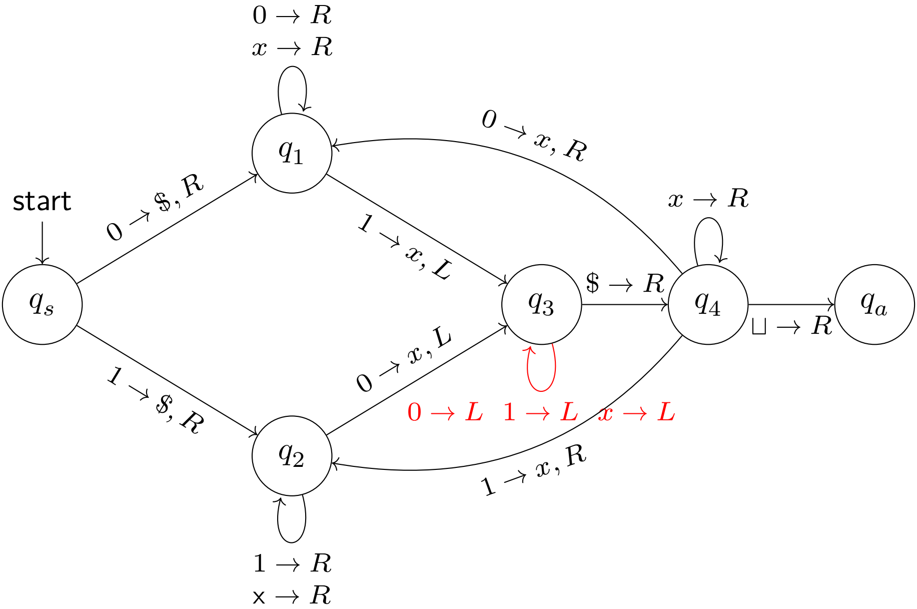
Hw7 Problem2

非空字串，且 1 和 0 的數量為相同
吃到 0 就找後面的一個 1 抵銷
吃到 1 就找後面的一個 0 抵銷
用 x 來標註已經找過的格子
每找完一次就回頭
需要注意的是，我們要有東西標註起點在哪 (這裡用 \$)
所以第一次吃字會有特別處理，順便偵測輸入是否為空

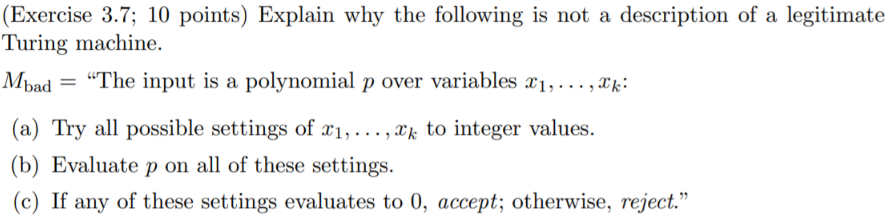
Hw7 Problem2

$M = (Q, \Sigma, \Gamma, \delta, q_{start}, q_{accept})$
$\Sigma = \{0, 1\}$
$\Gamma = \{0, 1, \sqcup, x, \$\}$

Hw7 Problem2



Hw7 Problem3



(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*."

Hw7 Problem3

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

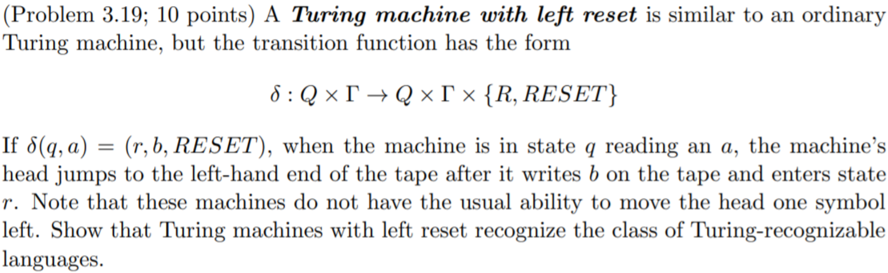
Hw7 Problem4

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

Hw7 Problem4

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

Hw7 Problem5



(Problem 3.19; 10 points) A ***Turing machine with left reset*** is similar to an ordinary Turing machine, but the transition function has the form

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

If $\delta(q, a) = (r, b, RESET)$, when the machine is in state q reading an a, the machine's head jumps to the left-hand end of the tape after it writes b on the tape and enters state r. Note that these machines do not have the usual ability to move the head one symbol left. Show that Turing machines with left reset recognize the class of Turing-recognizable languages.

Hw7 Problem5

要說明一個能夠向左回溯到頭，但無法像一般的圖靈機一樣向左看一格
其辨識效果和一般圖靈機相同

想法很簡單，把他想成：
總目標是想要往左一格的話，先把目前 head 的位置的 square 用一個 dot 標註起來
接下來做一個 reset 的動作

Hw7 Problem5

把 reset 過後的 head 往右移，直到遇到了一個 non-blank symbol
把這一個 symbol 塗成 blank (writes a blank in its square)，並把他記在 state 中，再往右一格
用記在 state 中的 symbol 去覆寫往右一格後的 symbol，直到遇到一開始標記 dot 的 symbol
如此一來可以實現所有的 symbol 都往右移了一格

Hw7 Problem5

遇到了標記了 dot 的 square，代表現在存在 state 中的 symbol 就是我們的目標

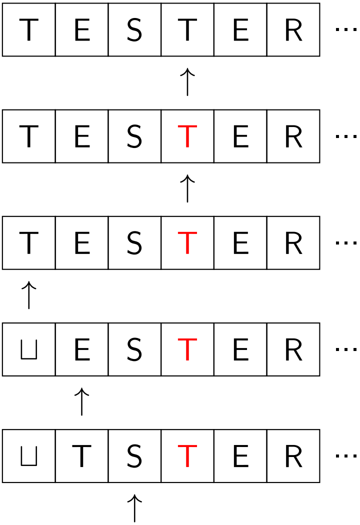
(複習一下我們的目標，用 left reset 實現往左移一格)

用存在 state 中的 symbol 寫入標記 dot 的 square 的同時

head 指針也剛好指在這個 square 上

接下來只需要再 reset 一次，往右移到標註 dot 的 square，就是我們的目標！

Hw7 Problem5



T	E	S	T	E	R	⋯
			↑			

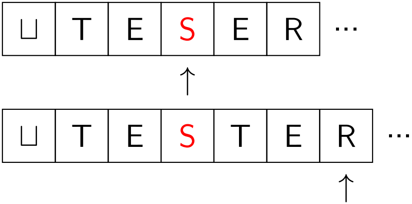
T	E	S	T	E	R	⋯
			↑			

T	E	S	T	E	R	⋯
↑						

⊔	E	S	T	E	R	⋯
	↑					

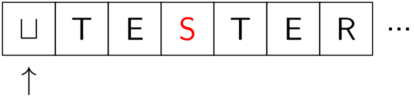
⊔	T	S	T	E	R	⋯
		↑				

Hw7 Problem5



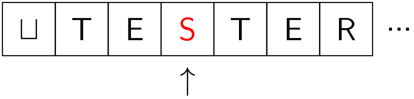
接下來在 reset 一次，一直讓指標往右跑：

Hw7 Problem5



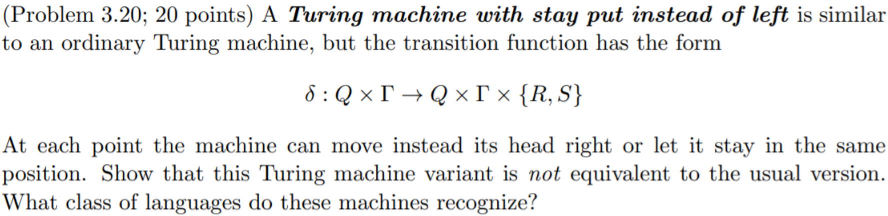
⊔ T E S T E R ⋯

直到指標指到被標記的 symbol，就是目標了！



⊔ T E S T E R ⋯

Hw7 Problem6



(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

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

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?

Hw7 Problem6

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？

Hw7 Problem6

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

Hw7 Problem6

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

Hw7 Problem6

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

Hw7 Problem6

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 $F \cup \{q'_{accept}\}$

Hw7 Problem6

假設原本圖靈機的 transition function 為 δ，而 NFA 的 transition relation 是 δ'
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

Hw7 Problem6

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 $a \in \Sigma$

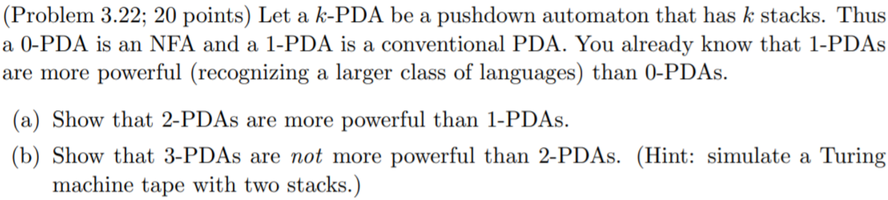
Hw7 Problem6

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

Hw7 Problem7



(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.)

Hw7 Problem7

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

Hw7 Problem7

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

Hw7 Problem7

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

Hw7 Problem7

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

Hw7 Problem7

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）

Hw7 Problem7

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

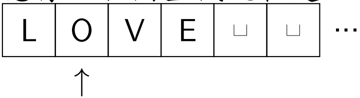
Hw7 Problem7

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

Hw7 Problem7

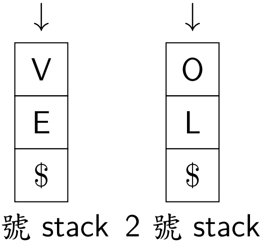
想像一台圖靈機現在處於這樣的狀態



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：



↓ ↓

V O

E L

$ $

1 號 stack 2 號 stack

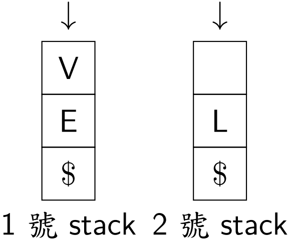
現在的指針正指在 O 的位置
假設我們想要做一個 $O \to I, R$ 的操作
在 TM 上的結果呈現會是：

L	I	V	E	␣	␣	⋯
		↑				

而想要用 2-PDA 呈現，其步驟會是：

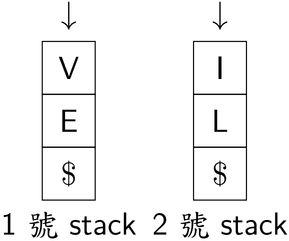
1. 把 O 從 stack 2 pop 出來
2. 把 I push 進 stack 2
3. 對指針做操作：把 V 從 stack 1 pop 出來
4. 把 V push 進 stack 2

Hw7 Problem7



↓ V E $ 1 號 stack

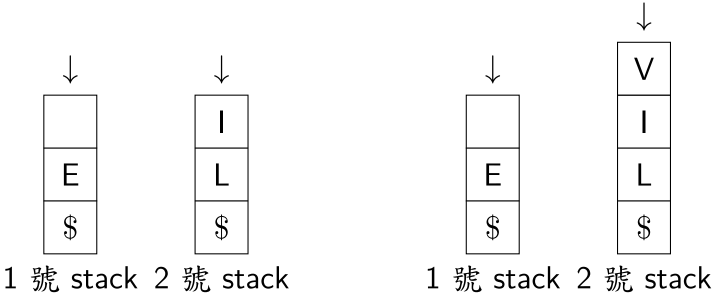
↓ L $ 2 號 stack



↓ V E $ 1 號 stack

↓ I L $ 2 號 stack

Hw7 Problem7



↓ ↓

E I

$ L

1 號 stack $

2 號 stack

↓ ↓

V

E I

$ L

1 號 stack $

2 號 stack

Hw7 Problem7

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

Hw8 Problem1

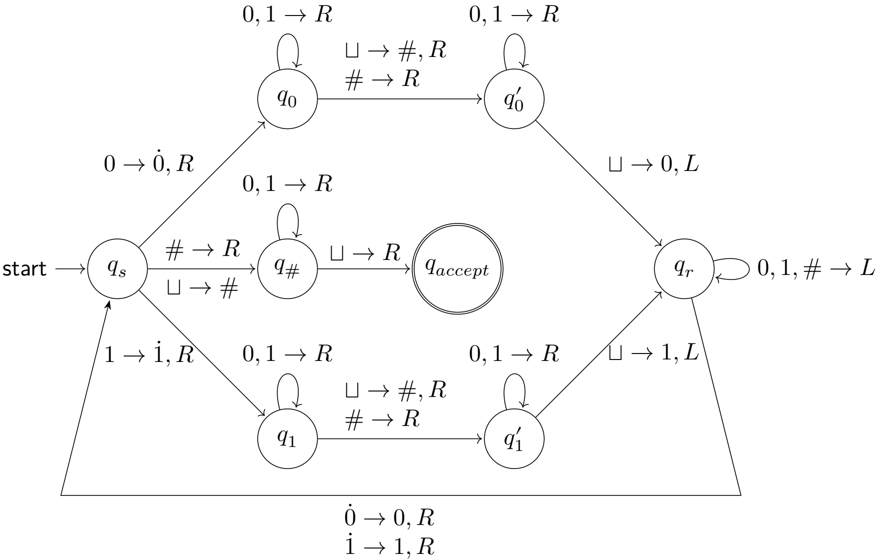
(10 points) Give a formal definition (with a state diagram) of a Turing machine that appends a # at the end of the input string and then copies and appends the original input after the #. The input alphabet is $\{0, 1\}$.

Hw8 Problem1

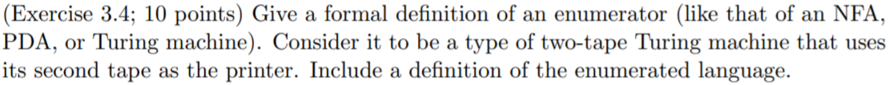
$M = (Q, \Sigma, \Gamma, \delta, q_s, q_{accept}, q_{reject})$, where Q, Σ, Γ are all finite sets and

- Q is the set of states,
- $\Sigma = \{0, 1\}$,
- $\Gamma = \{0, 1, \dot{0}, \dot{1}, \sqcup, \#\}$,
- $q_s \in Q$ is the start state,
- $q_{accept} \in Q$ is the accept state, and
- $q_{reject} \in Q$ is the reject state.

Hw8 Problem1



Hw8 Problem2



(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.

Hw8 Problem2

An enumerator is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{print}, q_{halt})$, 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 start state,
- $q_{print} \in Q$ is the printing state, and
- $q_{halt} \in Q$ is the halting state.

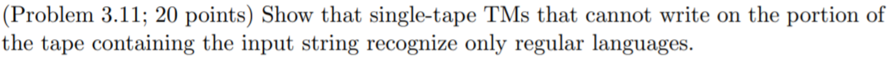
Hw8 Problem2

Definition of configurations and computation of enumerator are similar to corresponding definition for Turing machines:

- configuration is a snapshot of enumerator's state, positions of two tapes, and
- computation is a sequence of configurations, wherein each configuration after first is produced by previous one, according to transition function.

State q_{halt} denotes the end of enumeration, and q_{print} is responsible for printing: when we are in this state and if content of second tape (printer) is $w = w_1\#w_2\#\cdots w_n\# \sqcup \cdots$, where $w_i \in \Sigma^*$ for $1 \leq i \leq n$, we say that w is in language of enumerator.

Hw8 Problem3



(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.

Hw8 Problem3

Direct explanation:

TM's have memory and counting capability, while DFA's don't have. To memorize, TM's use special tape symbols and write on tape containing input. So, already read portion of tape is recognized. Machine can reach back to a marked position. So, we know where the head was last time. When we restrict the TM's to change any thing on tape containing input, no marker can be put on tape. So, no counting or memorization exists.

Hw8 Problem3

What such tapes can do in this situation?

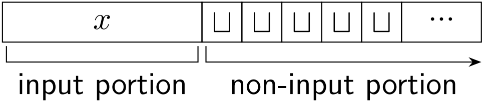
They start reading input but can't remember what they had read before. This is the exact property which DFA's hold. Even PDA's can remember in a limited fashion, but such TM's can't remember any thing. So, they also don't accept CFL's. Hence, they are equivalent to DFA's, and can recognize only regular languages.

Hw8 Problem3

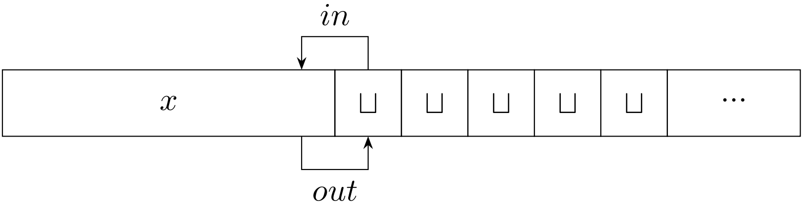
Precise explanation:

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.



Hw8 Problem3



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.

Hw8 Problem3

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.

Hw8 Problem3

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.

Hw8 Problem3

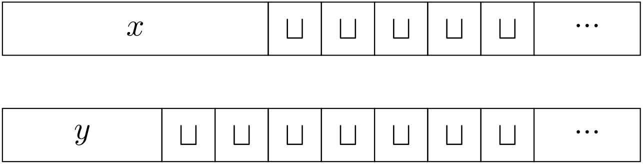
It is easy to check that if for two strings x and y, if:

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

we have x and y are indistinguishable by M (That is, M accepts xz if and only if M accepts yz).

$Why?$

Hw8 Problem3

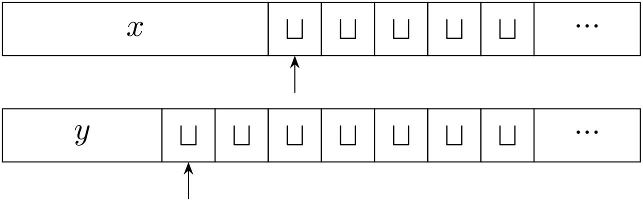

x
$\sqcup$ $\sqcup$ $\sqcup$ $\sqcup$ $\sqcup$ $\cdots$
y
$\sqcup$ $\sqcup$ $\sqcup$ $\sqcup$ $\sqcup$ $\sqcup$ $\sqcup$ $\cdots$

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.

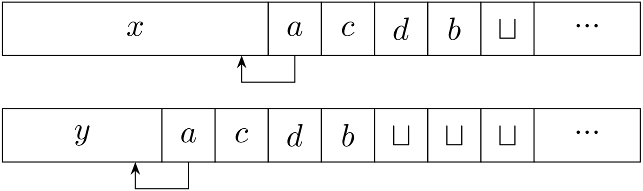
Hw8 Problem3


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

$Situation\ 2$:

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).

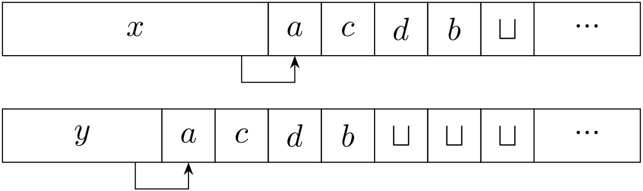
Hw8 Problem3



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.

Hw8 Problem3



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.

Hw8 Problem3

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.

Hw8 Problem3

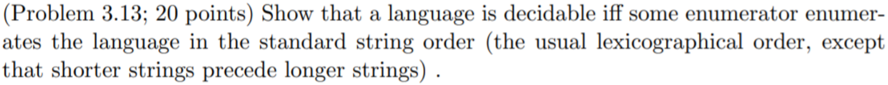
Is this situation, we say that x and y are in the same equivalence class (all strings in an equivalence class are indistinguishable to each other).

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. By Myhill-Nerode theorem, the language recognized by M is regular.

Hw8 Problem4



(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) .

Hw8 Problem4

$Proof$: 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.

Hw8 Problem4

$Proof$: 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, $reject$.

Hw8 Problem5

(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.

Hw8 Problem5

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, $accept$."

Hw8 Problem5

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, $reject$."

Hw8 Problem6

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

Hw8 Problem6

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 L_P be the context-free language that recognized by P, run R on input $\langle L_P \rangle$.
4. If R accepts, *accept*; otherwise, *reject*."

Hw8 Problem7

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

Hw8 Problem7

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 $(S_0 \rightarrow \epsilon) \in G'$, *accept* (in Chomsky normal form, only S_0 can generate ϵ); otherwise, *reject*."

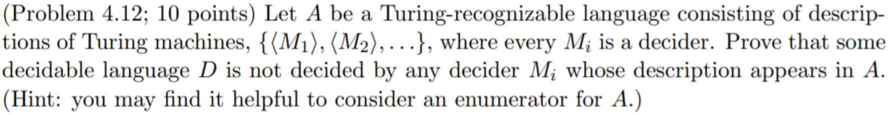
Hw8 Problem7

Reduction method:

Let $\textsc{tm}$ S decides $A_{\textsc{cfg}}$, we can construct a decider D as follows:

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

Hw9 Problem1



(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.)

Hw9 Problem1

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

Hw9 Problem1

學到對角論證法之後，當然要拿來用一用
至於怎麼用呢
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

Hw9 Problem1

	s_1	s_2	...	s_i
M_1	$\underline{\text{accept}}$	accept	...	reject
M_2	accept	$\underline{\text{reject}}$	...	accept
$\vdots$	...	...	...	...
M_i	reject	accept	...	$\underline{\text{reject}}$
$\vdots$	...	...	...	...

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

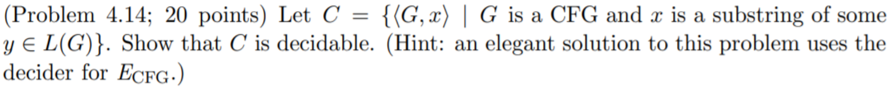
而且 M_i 本身是 Decider，這台機器一定會停機，所以 M_D 是 Decider，D 是 decidable language

得證，存在一個 D 不能被 A 當中的任何 Decider 給判定

Hw9 Problem1

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

Hw9 Problem2



(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}.)

Hw9 Problem2

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

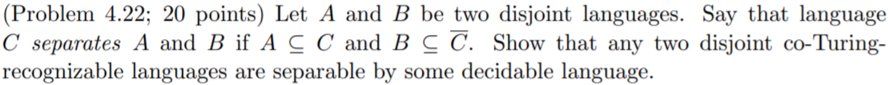
Hw9 Problem2

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

Hw9 Problem3



(Problem 4.22; 20 points) Let A and B be two disjoint languages. Say that language C *separates* A and B if $A \subseteq C$ and $B \subseteq \overline{C}$. Show that any two disjoint co-Turing-recognizable languages are separable by some decidable language.

Hw9 Problem3

我們的目標是：證明任意兩個 disjoint co-Turing recognizable languages
一定存在某個 decidable language 可以將他們 separate 開。

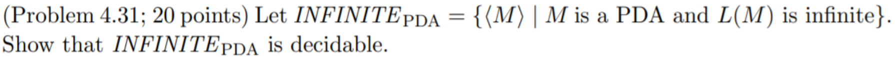
Hw9 Problem3

先令 A 和 B 為兩 co-Turing recognized languages，這可以使得 $\bar{A}$ 和 $\bar{B}$ 都為 Turing recognizable
再假設兩 Turing machine : $TM_{\bar{A}}$，$TM_{\bar{B}}$ 分別對應 $\bar{A}$ 和 $\bar{B}$
建一個 Turing machine TM_C:
$TM_C =$ “On input w where $w \in \bar{A} \cup \bar{B}$
1. Run both $TM_{\bar{A}}$ and $TM_{\bar{B}}$ on w simultaneously.
2. if $TM_{\bar{A}}$ accepts, reject; otherwise, accept.
3. if $TM_{\bar{B}}$ accepts, accept; otherwise, reject.“

Hw9 Problem3

因為 $w \in \bar{A} \cup \bar{B}$, 在把 w 丟到 $TM_{\bar{A}}$ 和 $TM_{\bar{B}}$ 後遲早會停下來
這代表著 $L(TM_C)$ 會是 decidable 的
結論就是: $L(TM_C)$ 會使得 $A \subseteq C$ 和 $B \subseteq \bar{C}$，便是我們的目標
的some language

Hw9 Problem4



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

Hw9 Problem4

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

Hw9 Problem4

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. "

Hw9 Problem5

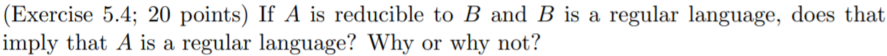
(Exercise 5.1; 10 points) Show that EQ_{CFG} is undecidable.

Hw9 Problem5

EQ_{CFG} is undecidable
這一題在已知 ALL_{CFG} 是 undecidable 的前提下就很好解

如果 EQ_{CFG} 是 decidable
那麼只要做出一個生成 Σ^* 的 CFG G
檢查其他 CFG 與 G 在不在 EQ_{CFG} 裡頭
就能判定這個 CFG 是否能生成 Σ^*
但已知 ALL_{CFG} 是不可判定語言，矛盾

Hw9 Problem6



(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?

Hw9 Problem6

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