Homework 6 - 10

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

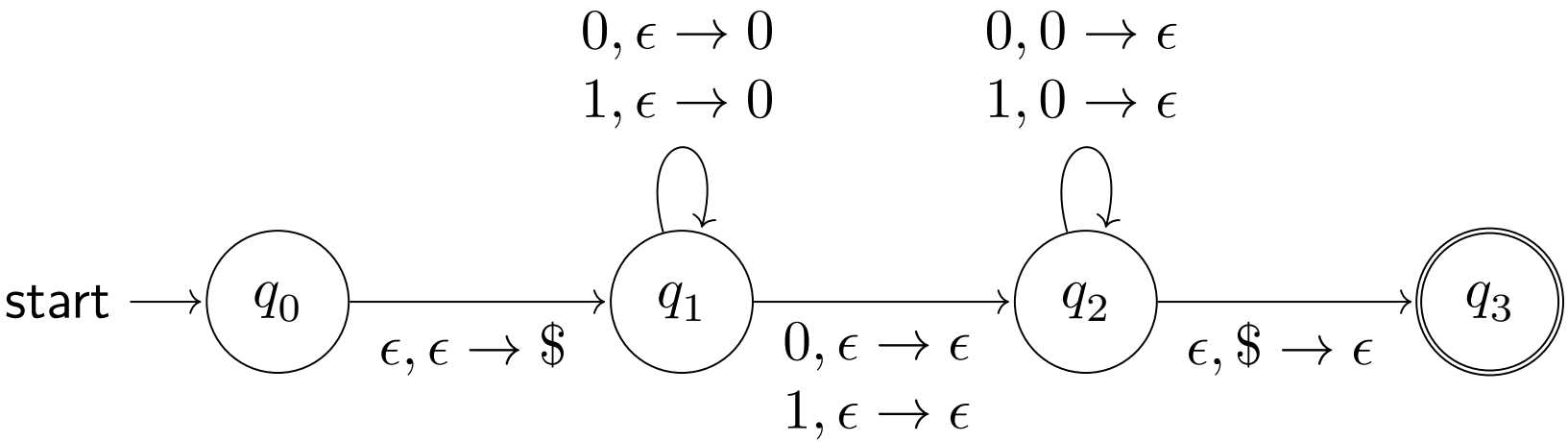
(Exercise 2.5; 20 points) Give informal descriptions and state diagrams of pushdown automata for the following languages. In all parts the alphabet Σ is $\{0, 1\}$.

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

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

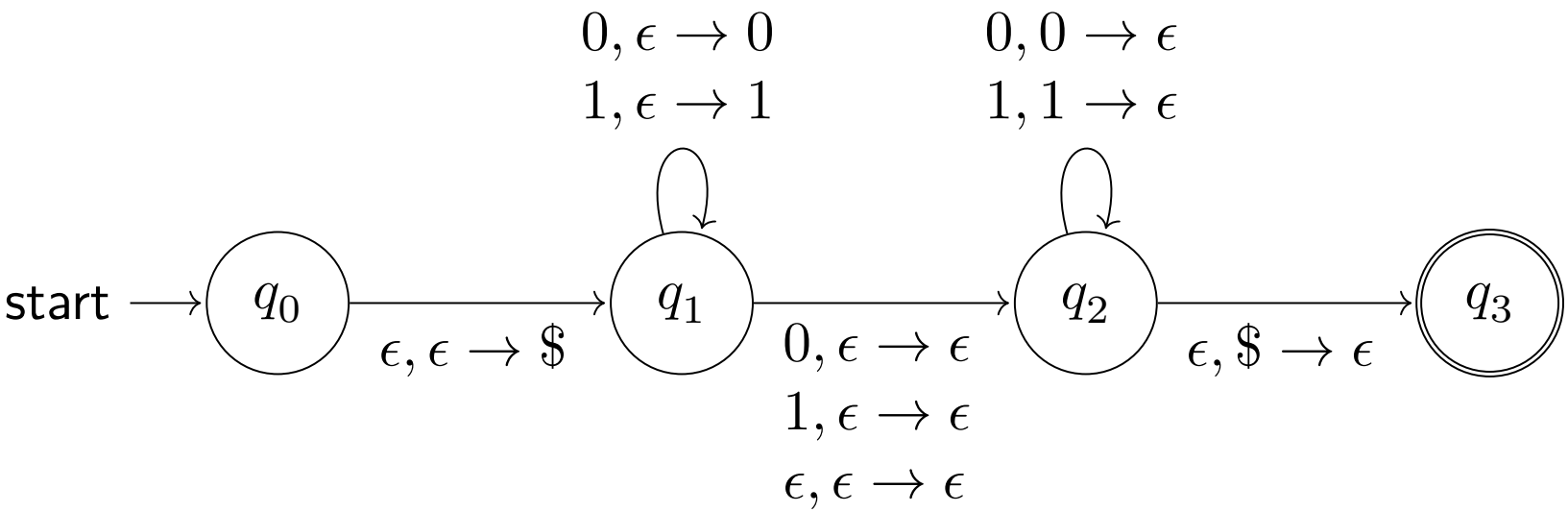
HW#6 Problem 2 (a)

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



HW#6 Problem 2 (b)

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

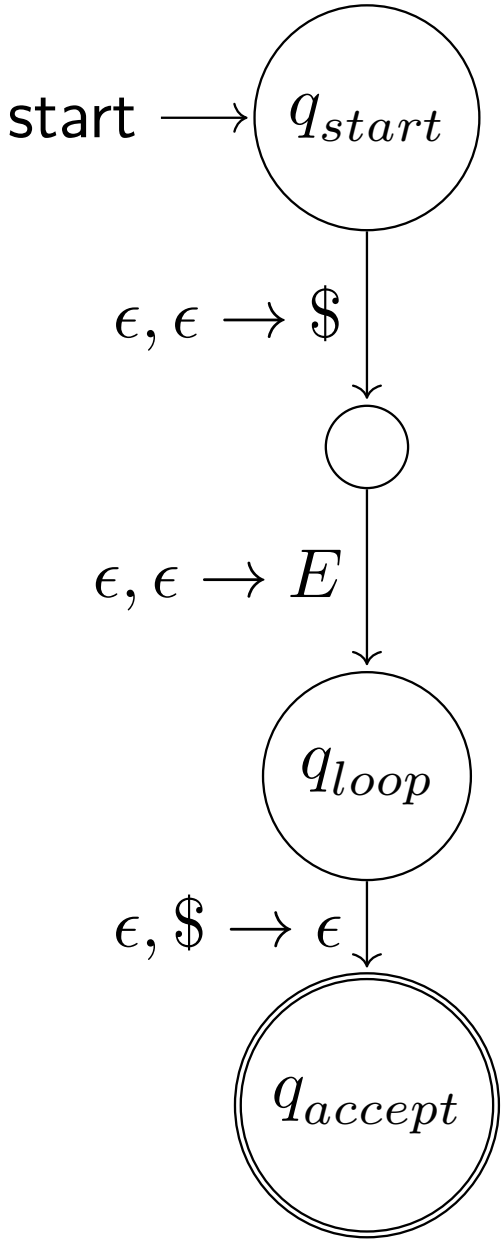


HW#6 Problem 3

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

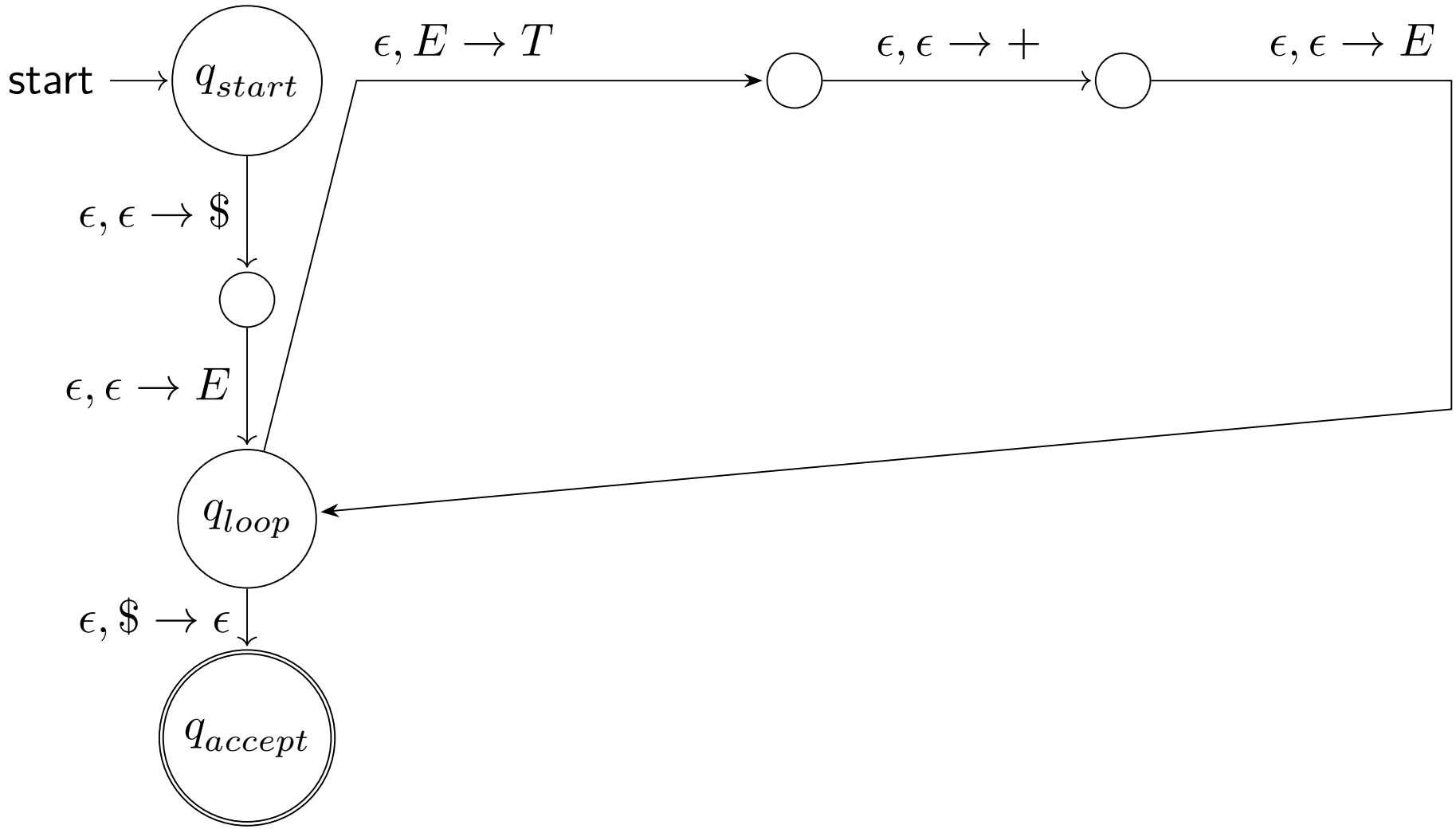
$$\begin{aligned} E &\rightarrow E + T \mid T \\ T &\rightarrow T \times F \mid F \\ F &\rightarrow (E) \mid a \end{aligned}$$

HW#6 Problem 3



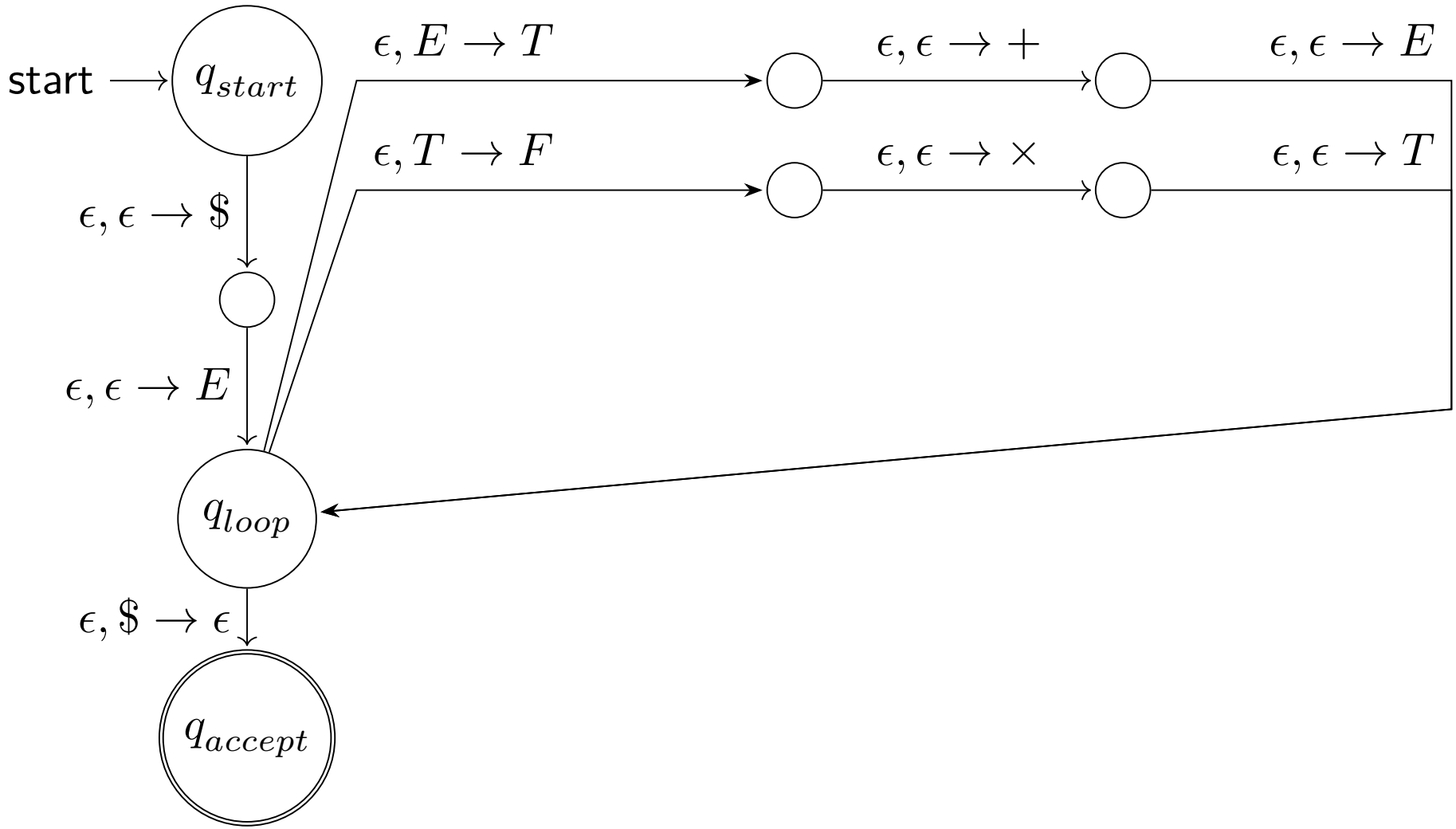
HW#6 Problem 3

$E \to E + T$



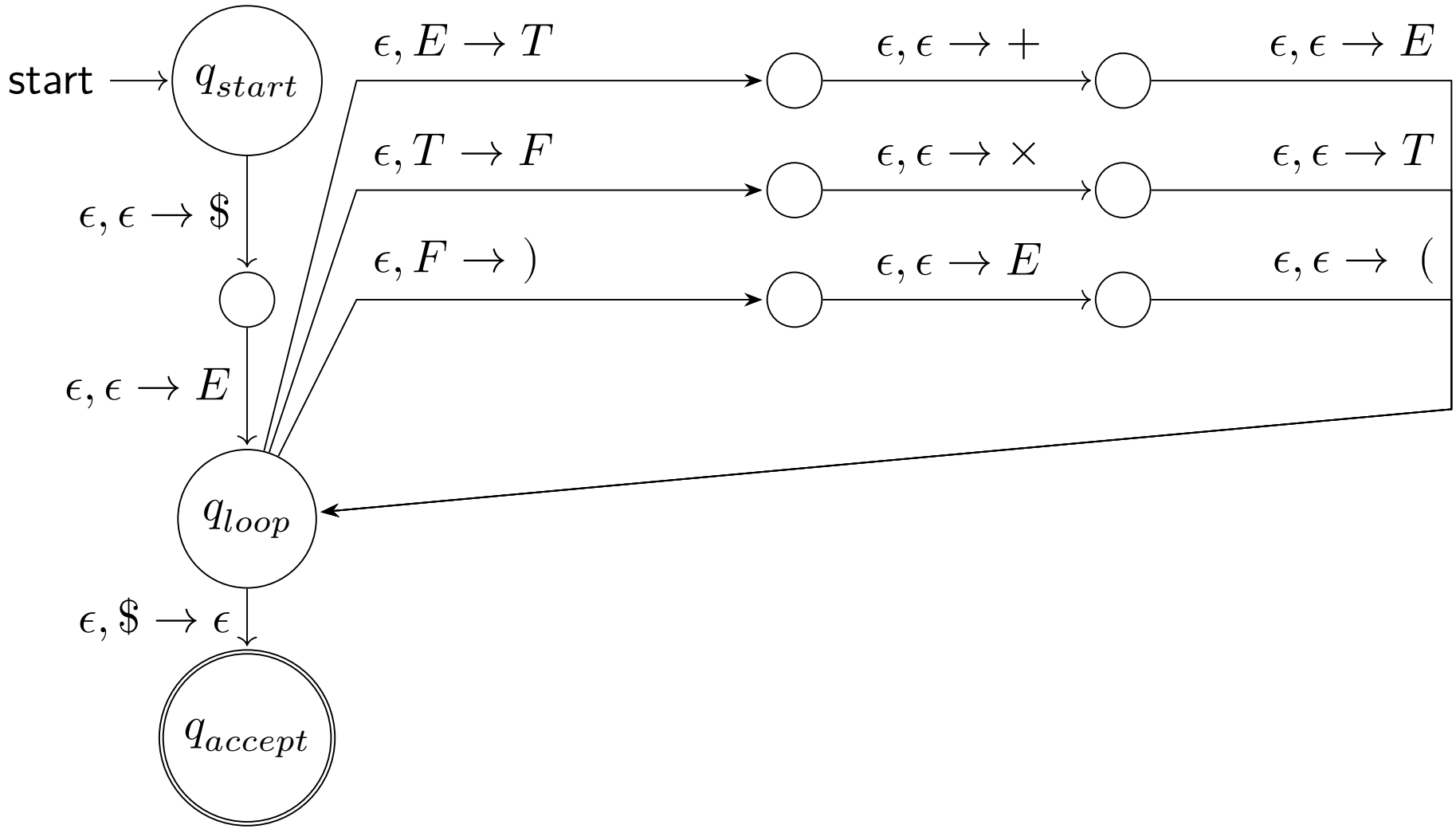
HW#6 Problem 3

$T \to T \times F$



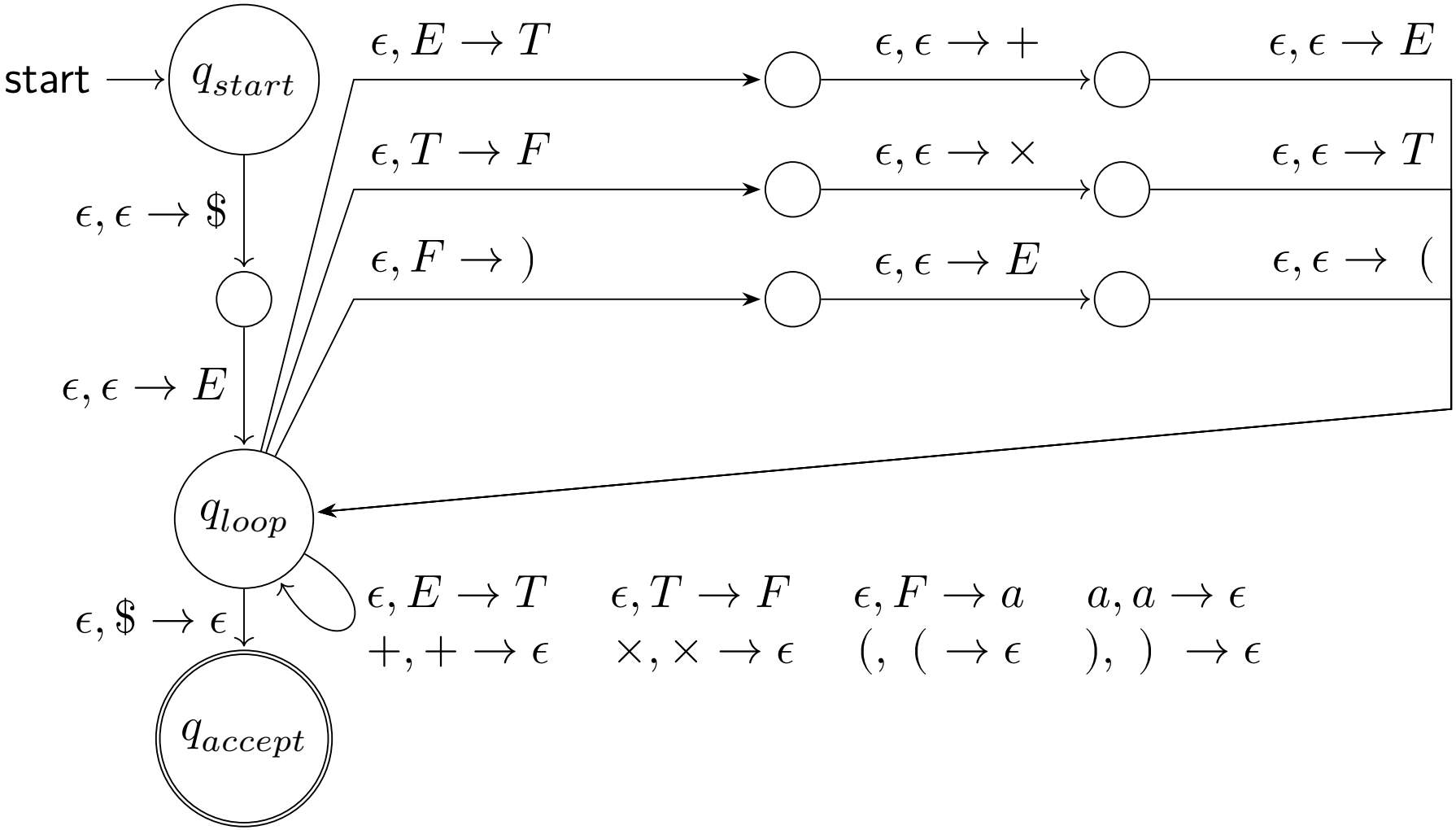
HW#6 Problem 3

$F \rightarrow \ (\ E \ )$



HW#6 Problem 3

Remaining grammar



HW#6 Problem 4

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

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

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

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

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 ways to obtain this language:

1.
⟨STMT⟩
⇒ ⟨IF-THEN⟩
⇒ `if condition then` ⟨STMT⟩
⇒ `if condition then` ⟨IF-THEN-ELSE⟩
⇒ `if condition then if condition then` ⟨STMT⟩ `else` ⟨STMT⟩
⇒ `if condition then if condition then a:=1 else a:=1`

HW#6 Problem 4 (a)

2.
⟨STMT⟩
⇒ ⟨IF-THEN-ELSE⟩
⇒ `if condition then` ⟨STMT⟩ `else` ⟨STMT⟩
⇒ `if condition then` ⟨IF-THEN⟩ `else` ⟨STMT⟩
⇒ `if condition then if condition then` ⟨STMT⟩ `else`
⟨STMT⟩
⇒ `if condition then if condition then a:=1 else a:=1`

So G is ambiguous.

HW#6 Problem 4 (b)

$\langle$STMT$\rangle \to \langle$ASSIGN$\rangle \mid \langle$IF-THEN$\rangle \mid \langle$IF-THEN-ELSE$\rangle$
$\langle$IF-THEN$\rangle \to$ `if condition then` $\langle$STMT$\rangle$
$\langle$IF-THEN-ELSE$\rangle \to$ `if condition then` $\langle$STMT$\rangle$ `else` $\langle$STMT$\rangle$
$\langle$ASSIGN$\rangle \to$ `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':

⟨STMT⟩ → ⟨ASSIGN⟩ | ⟨IF-THEN⟩ | ⟨IF-THEN-ELSE⟩

⟨IF-THEN⟩ → `if condition then` ⟨STMT⟩

⟨IF-THEN-ELSE⟩ → `if condition then` ⟨STMT-M⟩ `else` ⟨STMT⟩

⟨STMT-M⟩ → ⟨ASSIGN⟩ | ⟨IF-THEN-ELSE-M⟩

⟨IF-THEN-ELSE-M⟩ → `if condition then` ⟨STMT-M⟩ `else` ⟨STMT-M⟩

⟨ASSIGN⟩ → `a:=1`

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

HW#6 Problem 5

(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.

HW#6 Problem 5

Let $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$ and $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$ be two NFAs that recognize A and B, respectively. We can construct a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ that recognizes $A \diamond B$:

HW#6 Problem 5

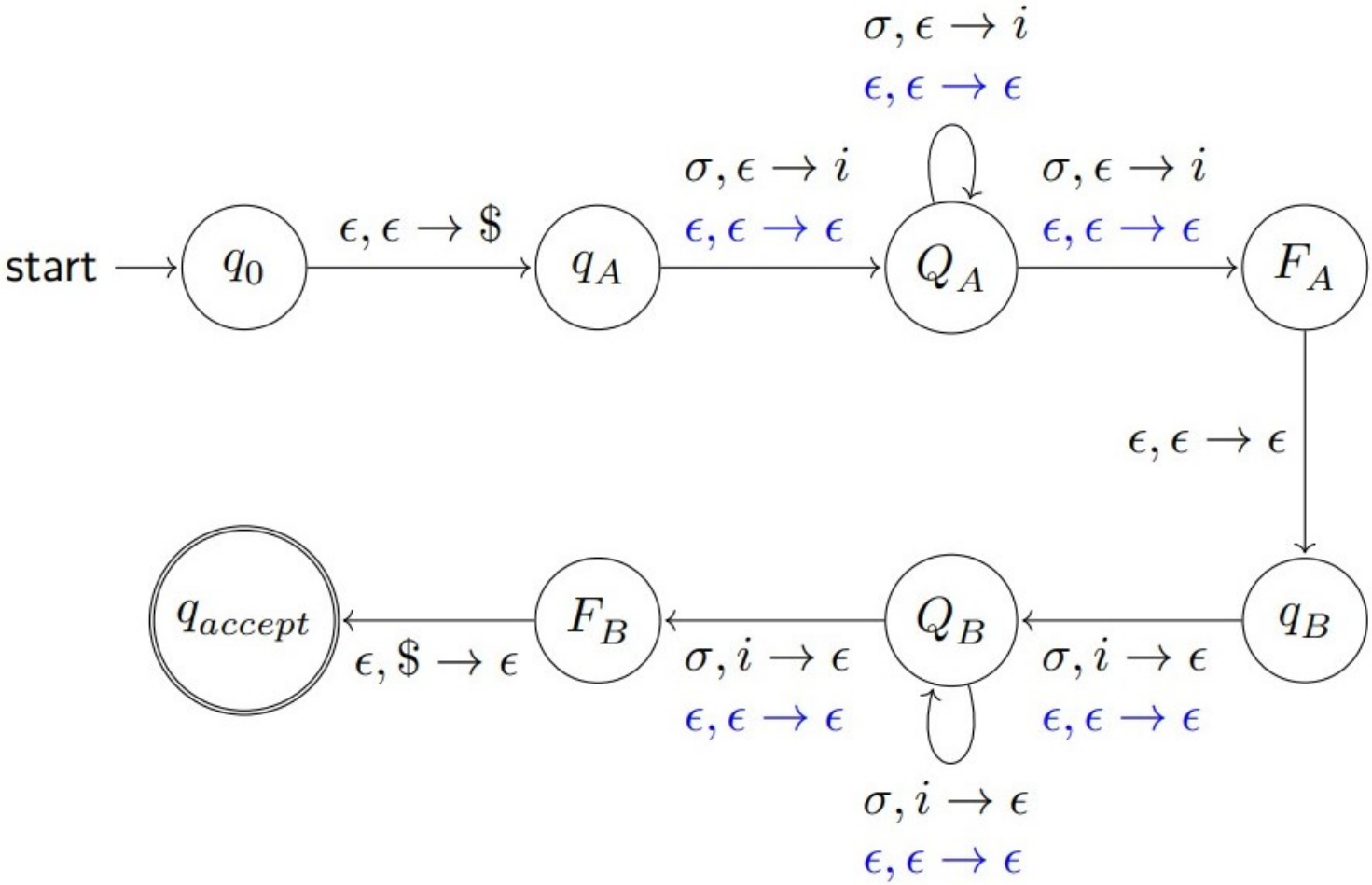
- $Q = Q_A \cup Q_B \cup \{q_0, q_{accept}\}$,
- $\Gamma = \{\$, i\}$,
- $\delta((q, \gamma), \sigma) =$

$$\begin{cases} (q_A, \$) & \text{if } q = q_0 \text{ and } \gamma = \epsilon \text{ and } \sigma = \epsilon \\ (\delta_A(q, \sigma), i) & \text{if } q \in Q_A \text{ and } \gamma = \epsilon \text{ and } \sigma \neq \epsilon \\ (\delta_A(q, \sigma), \epsilon) & \text{if } q \in Q_A \text{ and } \gamma = \epsilon \text{ and } \sigma = \epsilon \\ (q_B, \epsilon) & \text{if } q \in F_A \text{ and } \gamma = \epsilon \text{ and } \sigma = \epsilon \\ (\delta_B(q, \sigma), \epsilon) & \text{if } q =\in Q_B \text{ and } \gamma = i \text{ and } \sigma \neq \epsilon \\ (\delta_B(q, \sigma), \epsilon) & \text{if } q =\in Q_B \text{ and } \gamma = \epsilon \text{ and } \sigma = \epsilon \\ (q_accept, \epsilon) & \text{if } q \in F_B \text{ and } \gamma = \$ \text{ and } \sigma = \epsilon \end{cases}$$

- q_0 is the start state,
- $F = \{q_accept\}$ is the set of accept states.

HW#6 Problem 5

Schematic state diagram (not real PDA):



HW#6 Problem 6

(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.

HW#6 Problem 6

Let s be $0^p 1^{2p} 0^p$, 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 $0^i 1^j$ and y are all 1s, uv^2xy^2z will not be a palindrome.

if v are all 1s and y are $1^i 0^j$, uv^2xy^2z will not be a palindrome.
So A is not context free.

HW#6 Problem 7

7. (Problem 2.45; 10 points) Let $F = \{a^i b^j \mid i = kj \text{ for some positive integer } k\}$. Prove that F is not context free.

HW#6 Problem 7

Let s be $a^{r(p+1)}b^{p+1}$, where p is the pumping length and $r > p$.
Cases of dividing s as $uvxyz$ (where $|vy| > 0$ and $|vxy| \leq p$):

- if vxy are all b's, then uv^lxy^lz is not in F for all $l > r$.
- if vxy are all a's, then uv^0xy^0z is not in F $((r-1)(p+1) < r(p+1) - |vy| < r(p+1))$.
- if v are all a's and y are all b's. Suppose uv^lxy^lz is in F, then $r(p+1) + (l-1)|v| = k((p+1) + (l-1)|y|)$.
  (1) If $k \geq r$, $(k-r)(p+1) = (l-1)(|v| - k|y|)$, which leads to a contradiction since LHS is positive but $|v| - k|y| < 0$.
  (2) If $k < r$, $(r-k)(p+1) = (l-1)(k|y| - |v|)$, for sufficient large l, the equation does not hold.
  Thus we prove by contradiction that uv^lxy^lz is not in F.
- if v are a^ib^j and y are all b's, uv^2xy^2z is not in F.
- if v are all a's and y are a^ib^j, uv^2xy^2z is not in F.

So F is not context free.

HW#7 Problem 1

1. (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.

HW#7 Problem 1

$q_1 0000$

$\sqcup q_2 000$
$\sqcup x q_3 00$
$\sqcup x 0 q_4 0$
$\sqcup x 0 x q_3$
$\sqcup x 0 q_5 x$
$\sqcup x q_5 0 x$
$\sqcup q_5 x 0 x$

$\sqcup q_2 x 0 x$
$\sqcup x q_2 0 x$
$\sqcup x x q_3 x$
$\sqcup x x x q_3$
$\sqcup x x q_5 x$
$\sqcup x q_5 x x$
$\sqcup q_5 x x x$

$q_5 \sqcup x x x$
$\sqcup q_2 x x x$
$\sqcup x q_2 x x$
$\sqcup x x q_2 x$
$\sqcup x x x q_2$
$\sqcup x x x \sqcup q_{accept}$

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

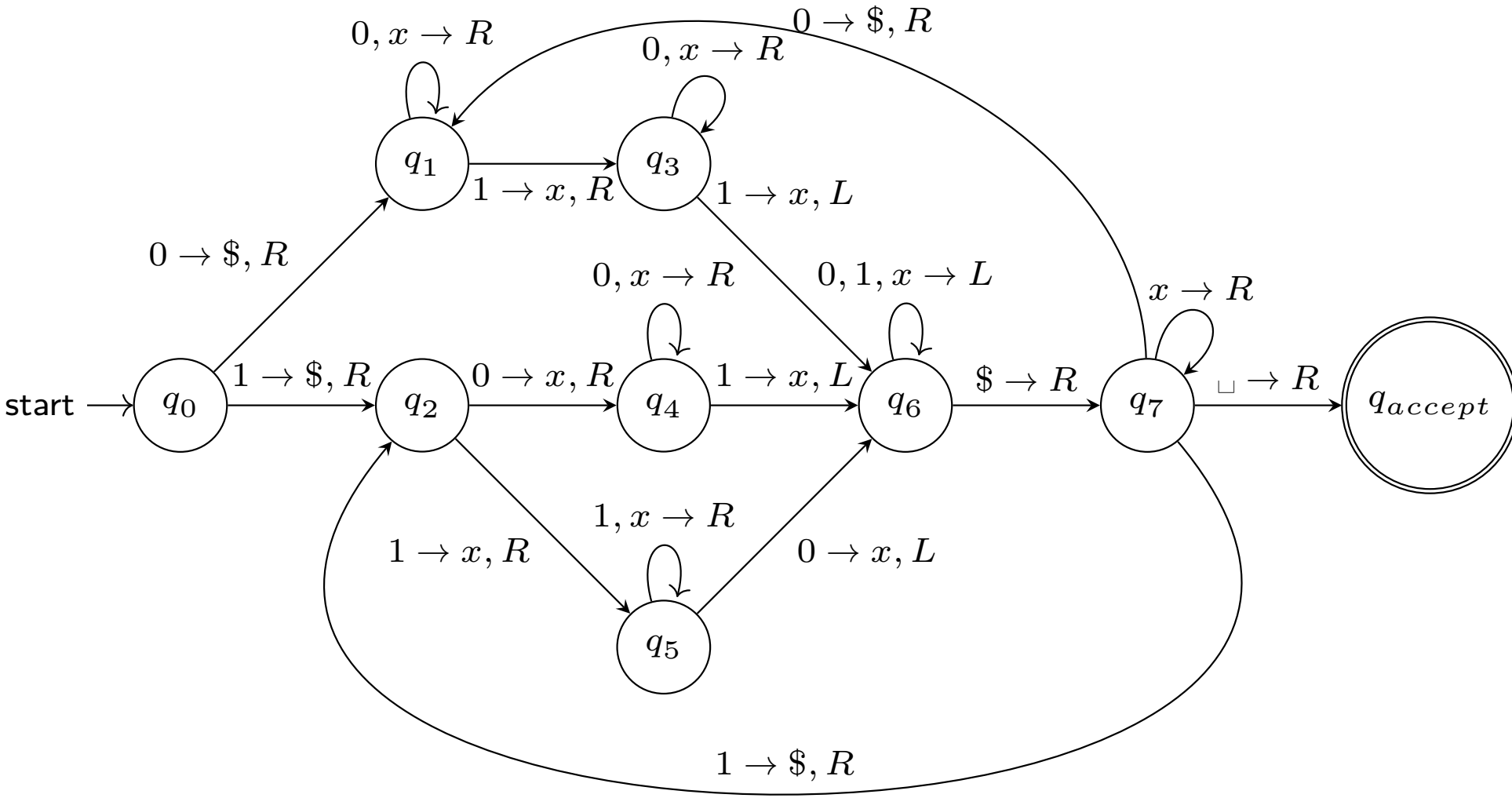
$L = \{\omega \in \{0,1\}^* \mid \omega$ is nonempty and contains twice as many 1s as 0s $\}$.

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

不是合法的 Turing machine，原因是並沒有寫出 (a) 步驟列舉所有 $x_1, ..., x_k$ 的方法
$=>$ 不同的列舉方法可能產生不同的結果!

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停機，M decides the concatenation of A and B

HW#7 Problem 5

(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

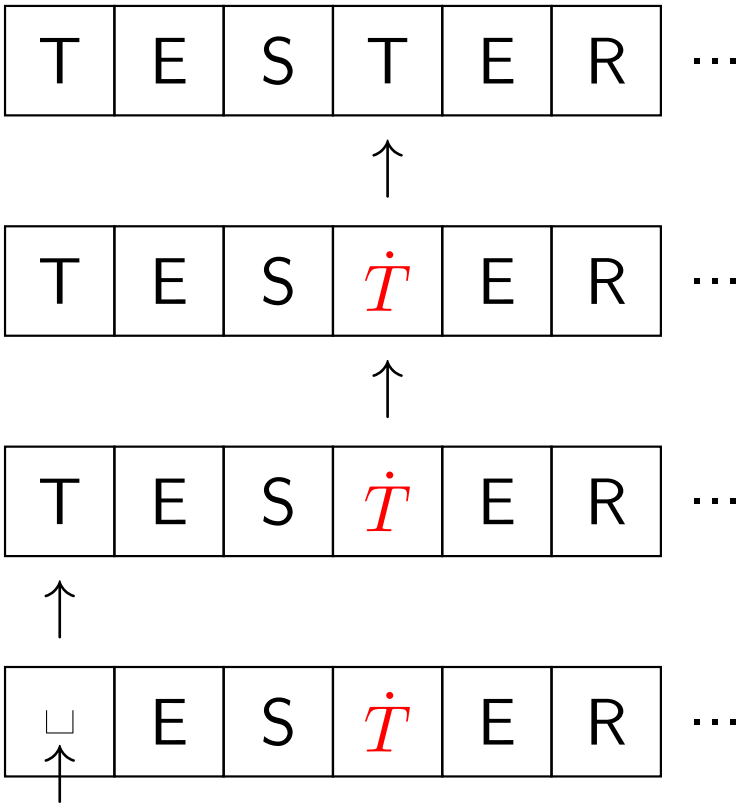
$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, RESET\}$$

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.

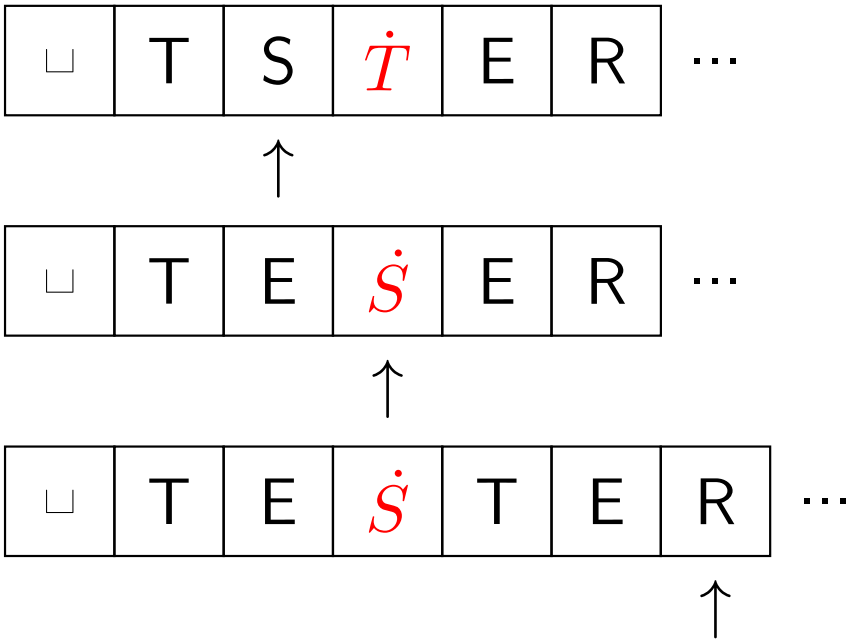
HW#7 Problem 5

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
想法是：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
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，如此一來可以實現所有的 symbol 都往右移了一格
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！

HW#7 Problem 5

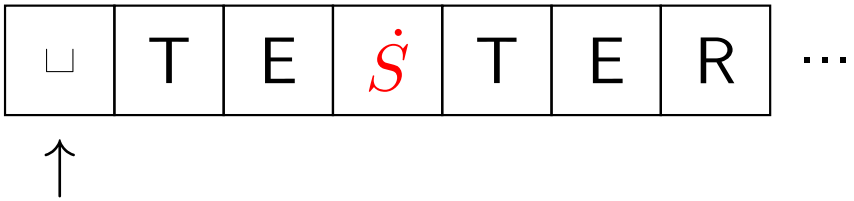


HW#7 Problem 5

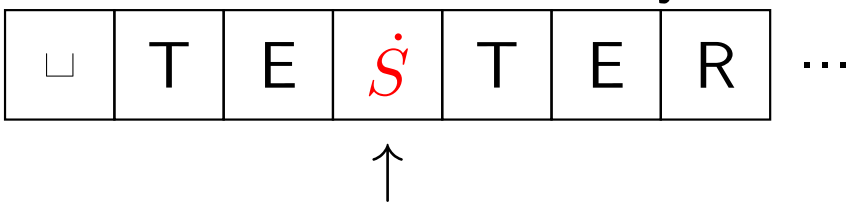


接下來再 reset 一次，一直讓指標往右跑，

HW#7 Problem 5



直到指標指到被標記的 symbol，就是目標了



HW#7 Problem 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，它的辨識能力究竟爲何？

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，因爲它永遠不回頭
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，因爲永遠不回頭，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 狀態遇到空格記爲一個 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？
因爲圖靈機是碰到 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 爲 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 爲 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 $X \in \Gamma$
$\delta(q, a) = (q', X, R)$，因爲往右走所以不需要理會 X，所以
$(q, a, q') \in \delta'$
$\delta(q, a) = (q', X, S)$，因爲停下來了，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 $Y \in \Gamma$
$\delta(q, X) = (q', Y, R)$，因爲往右走所以不需要理會 Y，所以
$((q, X), \epsilon, q') \in \delta'$
$\delta(q, X) = (q', Y, S)$，因爲停下來了，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
因爲 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

(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 爲空
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 (暫時稱爲 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

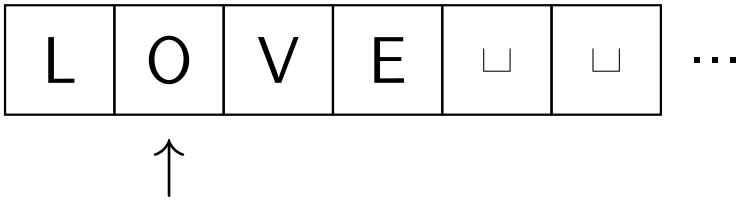
HW#7 Problem 7

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
因爲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

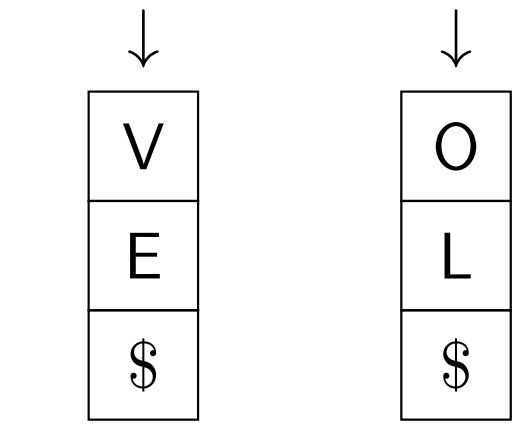
圖靈機指針向左，就是先對 1 號 stack 做 pop，塞入 TM 所寫入的字，再從 2 號 stack pop 字元塞入 1 號 stack
若是 2 號 stack 已經到底了，代表圖靈機跑到左邊的端點，上述的 pop 動作就不會執行

HW#7 Problem 7

想像一台圖靈機現在處於這樣的狀態

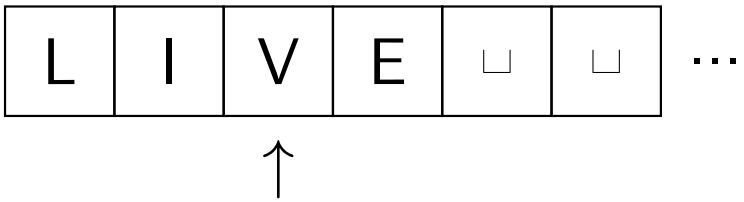


那麼 2-PDA 就可能 (依照處理過程不同，1 號 stack 的底部可能有更多東西) 是這樣：



1 號 stack　2 號 stack

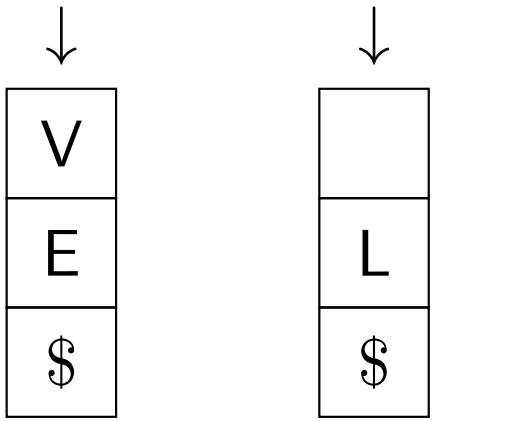
現在的指針正指在 O 的位置
假設我們想要做一個 $O \to I, R$ 的操作
在 TM 上的結果呈現會是：

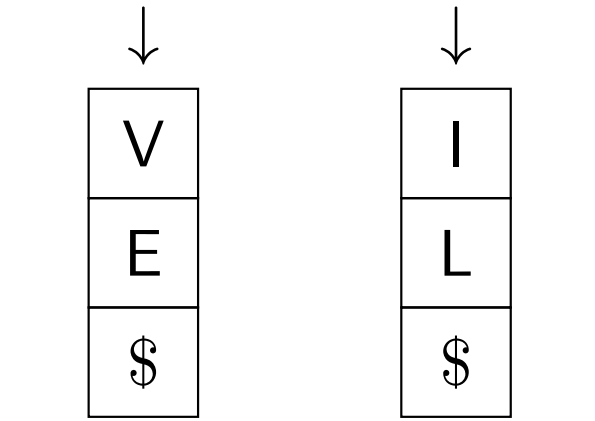


而想要用 2-PDA 呈現，其步驟會是：

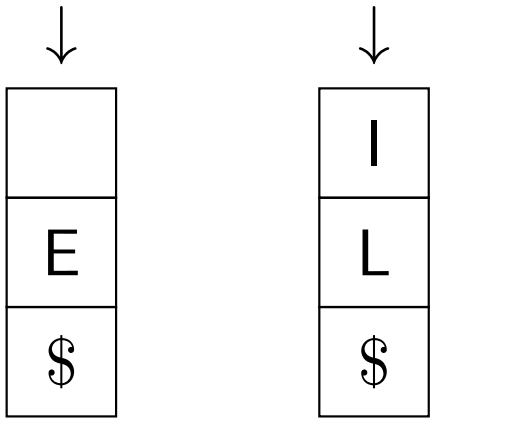
1. 把 O 從 stack 2 pop 出來
2. 把 I push 進 stack 2
3. 對指針做操作：把 V 從 stack 1 pop 出來
4. 把 V push 進 stack 2

HW#7 Problem 7

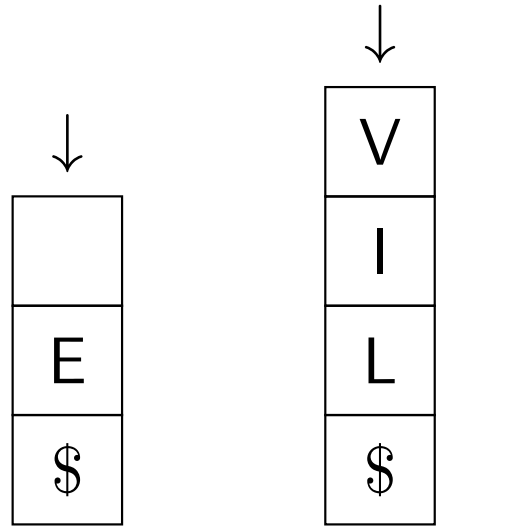




HW#7 Problem 7



1 號 stack 2 號 stack



1 號 stack 2 號 stack

HW#7 Problem 7

因爲 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

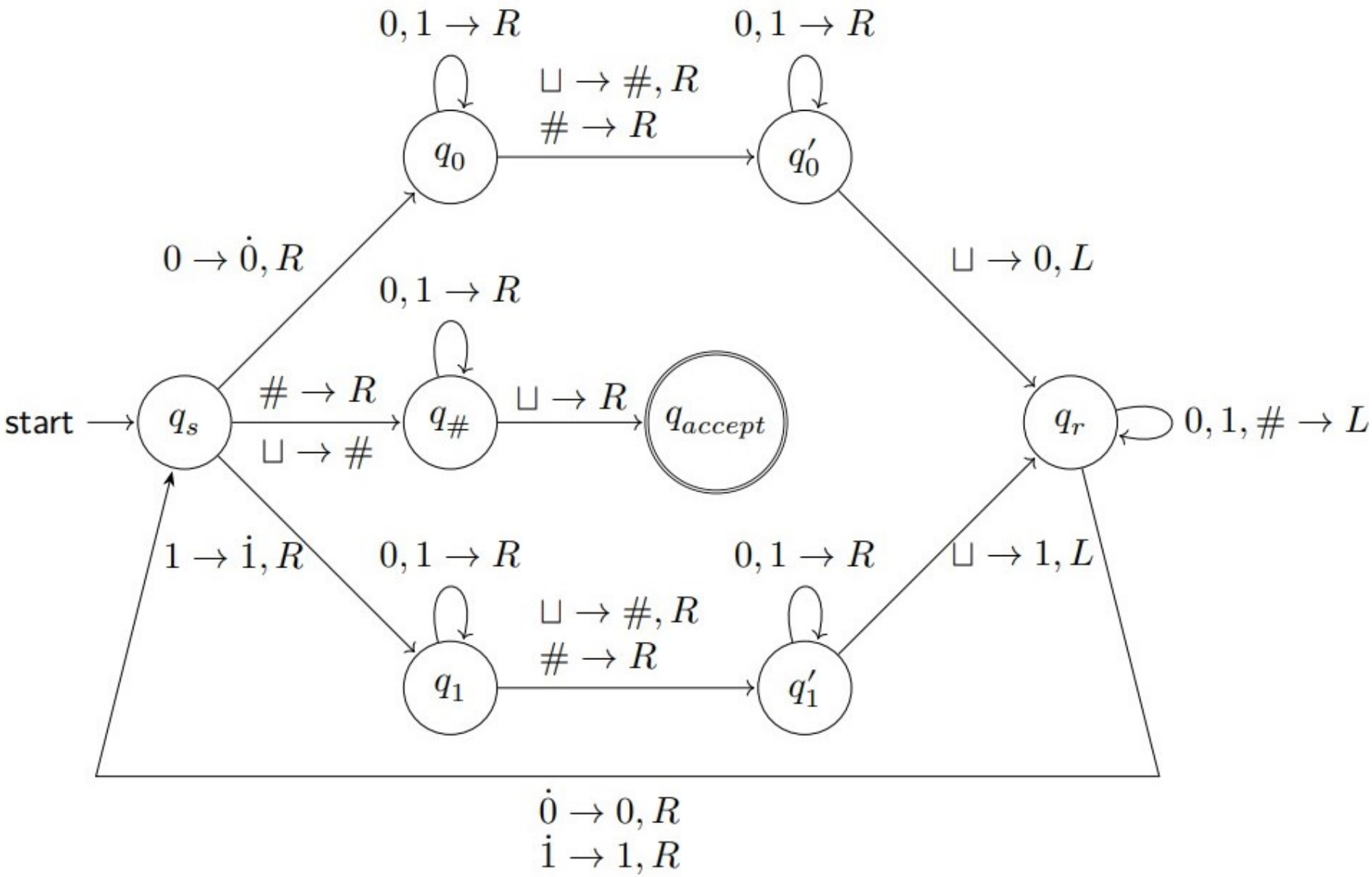
1. (10 points) Give a formal definition (with a state diagram) of a Turing machine that appends a # at the end of the input string and then copies and appends the original input after the #. The input alphabet is $\{0, 1\}$.

HW#8 Problem 1

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

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

HW#8 Problem 1



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 E 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 $\Sigma \subseteq \Gamma$,
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\} \times (\Sigma \cup \{\epsilon, \#\})$ 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, where $q_{print} \neq q_{reject}$.

HW#8 Problem 2

$\delta(q_1, a) = (q_2, b, R, c)$ means that when in state q_1, reading input a, E enters state q_2, writes b on the first tape, moves the first tape head right, and prints c on the second tape, then moves the first tape head right.
The enumerator halts when it enters q_{print}.
$L(E) = \{w | w \text{is on the second tape if } q_{print} \text{ is entered}\}$

HW#8 Problem 3

3. (Problem 3.12; 20 points) Show that every infinite Turing-recognizable language has an infinite decidable subset.

HW#8 Problem 3

Let A be an infinite Turing-recognizable language, then there exists an enumerator E that enumerates all strings in A.
We can construct an enumerator E' that prints a subset of A in lexicographic order:

1. Simulate E, when E prints its first string w_1, print w_1 and let $w_r = w_1$.
2. Continue simulating E.
3. When E is ready to print a new string w, check if w is longer than w_r (this ensures w occurs after w_r in lexicographic order). If so, then print w and let $w_r = w$, otherwise do not print w.
4. Go to 2.

HW#8 Problem 3

The language of E' is infinite since A is infinite, there exist strings in A longer than the current w_r , which means E will eventually print one of these and so will E'.
The language of E' language is decidable since it prints strings in lexicographic order(which will be proved in the next problem).
Thus, the language of E' is an infinite decidable subset of A.

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 lexicographic order, we can construct a decider D on input string s as follows:

Run E, when E is about to print s, check if s is the next string in lexicographic order of last printed string, if so, $accept$; otherwise, $reject$.(即判斷當順序走到要印 s 的時候是否有印出 s，如果印出了順序在 s 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 $correspondence$ 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 $one\text{-}to\text{-}one$). Hence for all $b \in B$, $f_{BA}(b)$ has exactly one corresponding output $a \in A$.
- f_{BA} is $one\text{-}to\text{-}one$: if f_{BA} is not $one\text{-}to\text{-}one$, $f_{AB}(a)$ 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 $correspondences$. 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 $one\text{-}to\text{-}one$: if $x \neq y$, $f_{AB}(x) \neq f_{AB}(y)$ because f_{AB} is $one\text{-}to\text{-}one$, and $f_{BC}(f_{AB}(x)) \neq f_{BC}(f_{AB}(y))$ because f_{BC} is $one\text{-}to\text{-}one$.
- 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 標記爲 M_i
而因爲 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	...
⋮	...	...	...	...	...
M_i	reject	accept	...	reject	...
⋮	...	...	...	...	...

依照這張表，建構一個 TM M_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，並且 M_D 在 input s_i 下會得到和 M_i 不同的結果

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

3. (Problem 4.16; 20 points) Let $PAL_{\text{DFA}} = \{\langle M \rangle \mid M \text{ is a DFA that accepts some palindrome}\}$. Show that PAL_{DFA} is decidable. (Hint: Theorems about CFLs are helpful here.)

HW#9 Problem 3

Suppose TM R decides E_{CFG}, and P is a PDA which $L(P) = \{w|w$ is a palindrome$\}$.
We can construct a decider D that decides PAL_{DFA},

D = " On input $\langle M \rangle$, M is a DFA
1. Construct a PDA P' which $L(P') = L(P) \cap L(M)$ (the intersection of a regular language and a context-free language is context-free).
2. Convert P' into an equivalent CFG G.
3. Run R on $\langle G \rangle$.
4. If R accepts, reject; otherwise, accept."

Since R is a decider and D runs in finite steps, PAL_{DFA} is 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 的語言爲空，則代表這個 state 完全不會被抵達

HW#9 Problem 4

Let the TM $M_{E_{CFG}}$ decides E_{CFG}, we can construct a decider M which $L(M) = \{\langle P \rangle | P \text{ has useless states}\}$,

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."

Since $M_{E_{CFG}}$ is a decider and M runs in finite steps, $L(M)$ is decidable.

HW#9 Problem 5

5. (Problem 4.22; 10 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.

HW#9 Problem 5

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。

HW#9 Problem 5

先令 A 和 B 爲兩 co-Turing recognized languages，這使得 $\bar{A}$ 和 $\bar{B}$ 都爲 Turing recognizable
再令兩 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 $w \in \bar{A} \cup \bar{B} = \Sigma^*$
1. Run both $TM_{\bar{A}}$ and $TM_{\bar{B}}$ on w simultaneously.
2. if $TM_{\bar{A}}$ accepts first, reject; if $TM_{\bar{B}}$ accepts first, accept. “

HW#9 Problem 5

因爲 $w \in \bar{A} \cup \bar{B} = \Sigma^*$，$TM_{\bar{A}}$ 和 $TM_{\bar{B}}$ 其中一個會 recognize w
代表著 $L(TM_C)$ decidable
如果 $w \in A$ 那麼 w 會被 $TM_{\bar{B}}$ accept
如果 $w \in B$ 那麼 w 會被 $TM_{\bar{A}}$ accept
因此 $A \subseteq C$ 且 $B \subseteq \bar{C}$

HW#9 Problem 6

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

HW#9 Problem 6

The idea is to reduce ALL_{CFG} to EQ_{CFG}.
Assume that a TM R decides EQ_{CFG}.
Construct a CFG G' which $L(G') = \Sigma^*$.
We could then construct a decider S for ALL_{CFG} as follows:

$S =$ " On input $\langle G \rangle$, G is a CFG:
1. Run TM R on input $\langle G, G' \rangle$.
2. If R rejects, reject; if R accepts, accepts. "

But we've known that ALL_{CFG} is undecidable, so EQ_{CFG} is undecidable.

HW#9 Problem 7

7. (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 7

利用反例說明 A 不一定是 regular language

假設 A 是 context-free language, 對應的 CFG 爲 G,
B 是 regular language, $B = \{1\}$,
建構一個 computable function f 使得 $w \in A \iff f(w) \in B$,
令 $M_{A_{CFG}}$ decides A_{CFG},

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"

HW#10 Problem 1

1. (Problem 5.12; 10 points) Let $J = \{w \mid \text{either } w = 0x \text{ for some } x \in A_{\text{TM}}, \text{ or } w = 1y \text{ for some } y \in \overline{A_{\text{TM}}}\}$. Show that neither J nor $\overline{J}$ is Turing-recognizable.

HW#10 Problem 1

Use the fact that $\overline{A_{TM}}$ is not Turing-recognizable.

Since $f(w) = 1w$ is a computable function and
$w \in \overline{A_{TM}} \iff f(w) \in J$, $\overline{A_{TM}} \leq_m J$.
Thus J is not Turing-recognizable.

Since $g(w) = 0w$ is a computable function and
$w \in A_{TM} \iff f(w) \in J$, $A_{TM} \leq_m J$. (i.e. $\overline{A_{TM}} \leq_m \overline{J}$)
Thus $\overline{J}$ is not Turing-recognizable.

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_{2DFA}.

The idea is to construct a 2DFA that recognizes the accept computational history $c_1 \# c_2 \# \ldots \# c_n$ of a TM M.

To do so, the 2DFA checks if c_1 consists q_{start} and a symbol in Σ, and then checks if c_n consists q_{accept} and symbols in Σ.

For middle transitions, let one head on c_i and the other on c_{i+1} and check each states and symbols.

HW#10 Problem 2

Assume that a TM D_{2DFA} decides E_{2DFA}, 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_{2DFA} on input $\langle N \rangle$.
3. If D_{2DFA} accepts, *accept*; otherwise, *reject*."

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

HW#10 Problem 3

3. (Problem 5.18 adapted; 20 points) Please discuss briefly the applicability of Rice's theorem to proving the undecidability of each of the following languages.

   (a) $HALT_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on } w\}$.

   (b) $REGULAR_{\text{TM}} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) \text{ is regular}\}$.

   (c) $E_{\text{LBA}} = \{\langle M\rangle \mid M \text{ is an LBA where } L(M) = \emptyset\}$.

   (d) $ALL_{\text{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \Sigma^*\}$.

HW#10 Problem 3

To show if Rice's theorem is applicable, check two things:

(1) The property is of a language recognized by a TM.

(2) The property is non-trivial.

The property is non-trivial if there exists a TM satifies it and one does not.

(a) Not applicable. M is a TM but "a TM halts" is not a property of a language.

(b) Applicable. M is a TM and "$L(M)$ is regular "is a non-trivial property of a language.

(c) Not applicable. M is not a TM

(d) Not applicable. G is not a TM.

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

5. (Problem 5.23(b); 10 points) A variable A in CFG G is said to be *necessary* if it appears in every derivation of some string $w \in L(G)$. Let $NECESSARY_{\text{CFG}} = \{\langle G, A\rangle \mid A \text{ is a necessary variable in } G\}$. Prove that $NECESSARY_{\text{CFG}}$ is undecidable.

HW#10 Problem 5

Suppose TM R decides $NECESSARY_{CFG}$,
we can construct a decider D that decides ALL_{CFG} as follows:

$D =$ "On input $\langle G \rangle$, where G is a CFG:
1. Map $\langle G \rangle$ to $\langle G', A \rangle$ which A is a new variable that does not appear in G, and G' is G adding these new rules:
   (1) S(start variable of G) $\rightarrow A$
   (2) $A \rightarrow aA$ for all $a \in \Sigma$
   (3) $A \rightarrow \epsilon$
2. Run R on $\langle G', A \rangle$, if R accepts, reject; otherwise, accept.

If A is not a necessary variable of G', then G' can generate Σ^* using only rules from G, thus $G \in ALL_{CFG}$; otherwise G' cannot generate Σ^* using only rules from G.
But we've known that ALL_{CFG} is undecidable, so $NECESSARY_{CFG}$ is undecidable.

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 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.

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

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

The decider D will decide ALL_{DFA} in $(O(n^2))$, 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 $G(V, E)$ and $H(V', E')$ are undirected graphs:
1. If $|V| \neq |V'|$ or $|E| \neq |E'|$, $reject$.
2. Nondeterministically select a permutation π of m elements.
3. For all $\{(x, y)|x, y \in V\}$, check whether "$(x, y) \in E$ iff $(\pi(x), \pi(y)) \in E'$" is satisfied . If all agree, $accepts$. If any differ, $reject$.

Stage 2 can be implemented in polynomial time nondeterministically. (arbitrary pop a node x from V and repeat until V is empty)
Stages 1 and 3 takes polynomial time. Hence ISO $\in$ NP.