# Q1

(a) 如果A是一个正则语言，那么|A|<∞。（错误）

正则语言的定义是它们可以被有限状态自动机（DFA或NFA）识别。正则语言的集合是无限的，因为可以构造出无限多个不同的正则语言。因此，说正则语言的集合大小小于无穷是不正确的。

**If A is a regular language, then |A| < ∞. (False)**

Explanation: A regular language is defined as one that can be recognized by a finite state automaton (DFA or NFA). The set of regular languages is infinite, as it is possible to construct an infinite number of different regular languages. Therefore, stating that the size of the set of regular languages is less than infinite is incorrect.

(b) 如果语言A是非正则的，那么它有一个NFA。（正确）

非正则语言是指那些不能被有限状态自动机识别的语言。尽管如此，对于任何语言，无论是正则还是非正则，都可以构造一个非确定有限状态自动机（NFA）来识别它。因此，这个陈述是正确的。

**If a language A is nonregular, then it has an NFA. (True)**

Explanation: A nonregular language is one that cannot be recognized by a finite state automaton. However, for any language, whether regular or nonregular, it is possible to construct a non-deterministic finite state automaton (NFA) that can recognize it. Therefore, this statement is true.

(c) 一个NFA的转移函数是δ: Q x Σ → Q。（错误）（PPT L3 p10）

一个非确定有限状态自动机（NFA）的转移函数通常定义为δ: Q x (Σ ∪ ε) → P(Q)，其中Q是状态集合，Σ是输入字母表，P(Q)是Q的幂集，ε表示空字符串。因此，给出的转移函数定义是不正确的。

The transition function of an NFA is δ: Q x Σ → Q. (False)

Explanation: The transition function of a non-deterministic finite state automaton (NFA) is typically defined as δ: Q x (Σ ∪ ε) → P(Q), where Q is the set of states, Σ is the input alphabet, P(Q) is the power set of Q, and ε represents the empty string. Therefore, the given definition of the transition function is incorrect.

(d) 正则表达式(01\*0U1)\*0生成由所有包含奇数个0的字符串组成的语言。（错误）

正则表达式(01\*0U1)\*0描述的语言包含0和1的任意组合，但并不保证字符串中有奇数个0。例如，字符串"010"就包含偶数个0，但符合该正则表达式。因此，该陈述是错误的。

**The regular expression (01\*0U1)\*0 generates the language consisting of all strings over Σ={0,1} having an odd number of 0's. (False)**

Explanation: The regular expression (01\*0U1)\*0 describes a language that includes any combination of 0s and 1s, but it does not ensure that the string has an odd number of 0s. For example, the string "010" contains an even number of 0s but still matches the regular expression. Therefore, this statement is false.

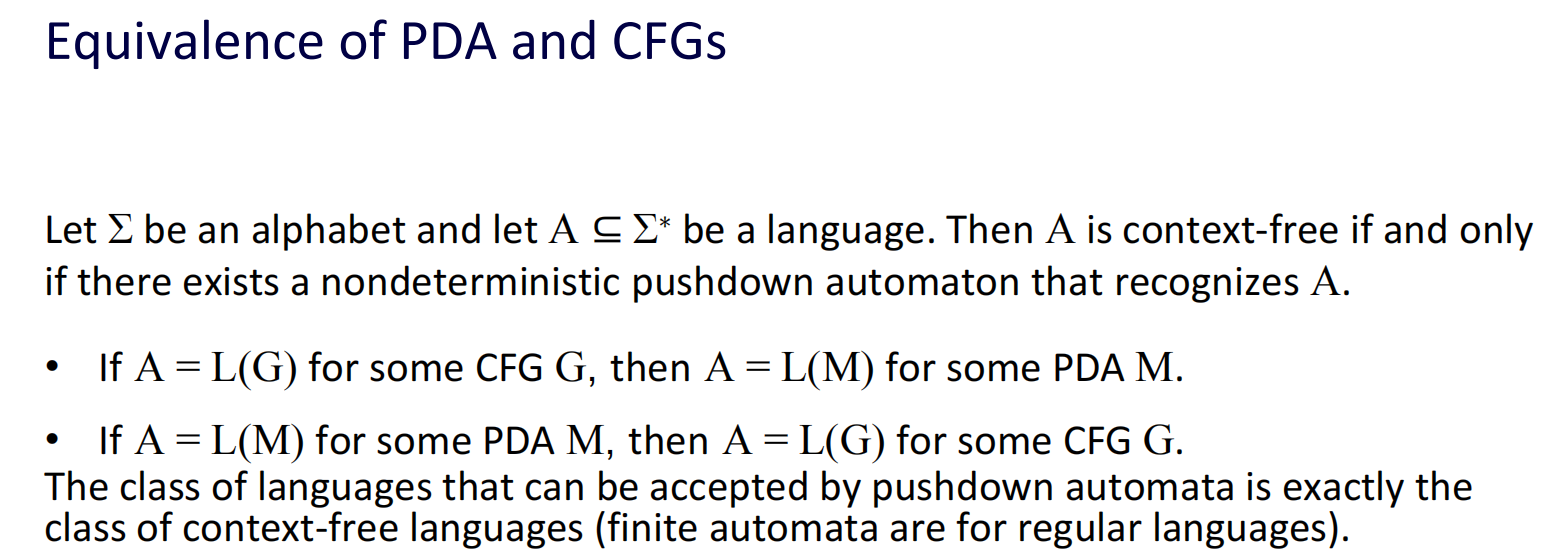
(e) 如果语言A是正则的，那么A有一个Chomsky正常形式的CFG。（正确）

任何正则语言都可以被转换成一个Chomsky正常形式的上下文无关文法（CFG）。因此，如果语言A是正则的，那么它确实可以有一个Chomsky正常形式的CFG。

**If a language A is regular, then A has a CFG in Chomsky normal form. (True)**

Explanation: Any regular language can be converted into a Context-Free Grammar (CFG) in Chomsky normal form. Therefore, if a language A is regular, it indeed can have a CFG in Chomsky normal form.

(f) 语言A是上下文无关的当且仅当存在一个确定的下推自动机D使得A=L(D)。（错误）



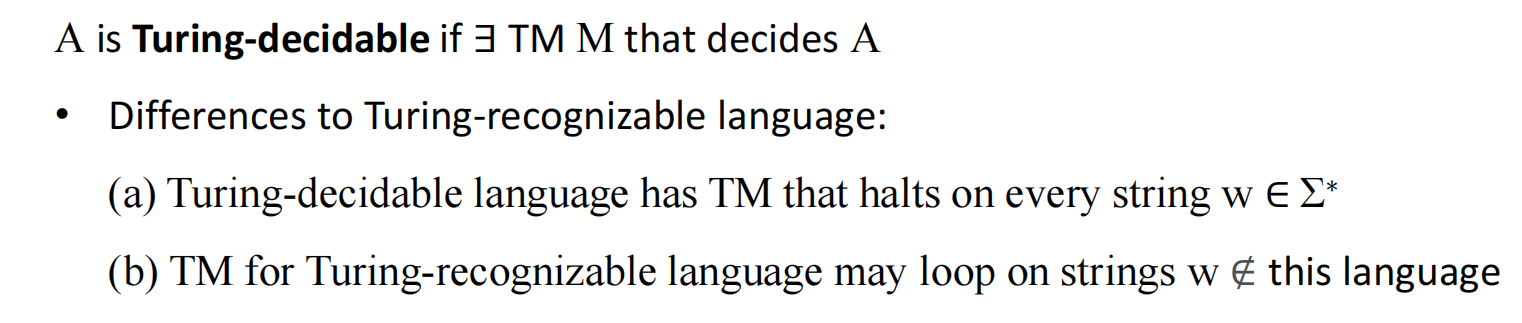
语言A是上下文无关的，这意味着它可以被一个上下文无关文法生成，或者可以被一个非确定的下推自动机（PDA）识别。并不是所有上下文无关语言都可以被一个确定的下推自动机识别，因此这个陈述是错误的。

**A language A is context-free if and only if there exists a deterministic pushdown automaton D such that A=L(D). (False)(PPT L8 p4)**

Explanation: A language A is context-free if it can be generated by a context-free grammar or recognized by a non-deterministic pushdown automaton (PDA). Not all context-free languages can be recognized by a deterministic pushdown automaton, so this statement is false.

(g) 语言A是图灵可判定的如果存在一个图灵机TM使得A=L(TM)。（正确）(PPT L9 p43)

一个语言是图灵可判定的，如果存在一个图灵机可以判断该语言中的所有字符串是否属于该语言。因此，如果存在一个图灵机TM使得A=L(TM)，那么语言A是图灵可判定的。



**A language A is Turing-decidable if there exists a Turing machine TM such that A=L(TM). (True)**

Explanation: A language is Turing-decidable if there exists a Turing machine that can determine whether all strings in the language belong to it. Therefore, if there exists a Turing machine TM such that A=L(TM), then the language A is Turing-decidable.

(h) 如果语言A可以被多带图灵机识别，A是图灵可识别的。（正确）

如果一个语言可以被多带图灵机识别，那么它也可以被单带图灵机识别。因此，如果语言A可以被多带图灵机识别，它也是图灵可识别的。

**If a language A can be recognized by a multi-tape Turing machine, A is TM-recognizable. (True)**

Explanation: If a language can be recognized by a multi-tape Turing machine, it can also be recognized by a single-tape Turing machine. Therefore, if a language A can be recognized by a multi-tape Turing machine, it is also Turing-recognizable.

(i) 所有语言的集合是可数的。（错误）

所有语言的集合是不可数的，因为语言的集合是无穷的，并且对于每个自然数，我们都可以构造出一个不同的语言。因此，所有语言的集合是不可数的。

**The set of all languages is countable. (False)**

Explanation: The set of all languages is uncountable because the set of languages is infinite, and for every natural number, we can construct a different language. Therefore, the set of all languages is uncountable.

(j) If a language A is mapping reducible to a TM-recognizable language B and A is decidable, then B is decidable also.

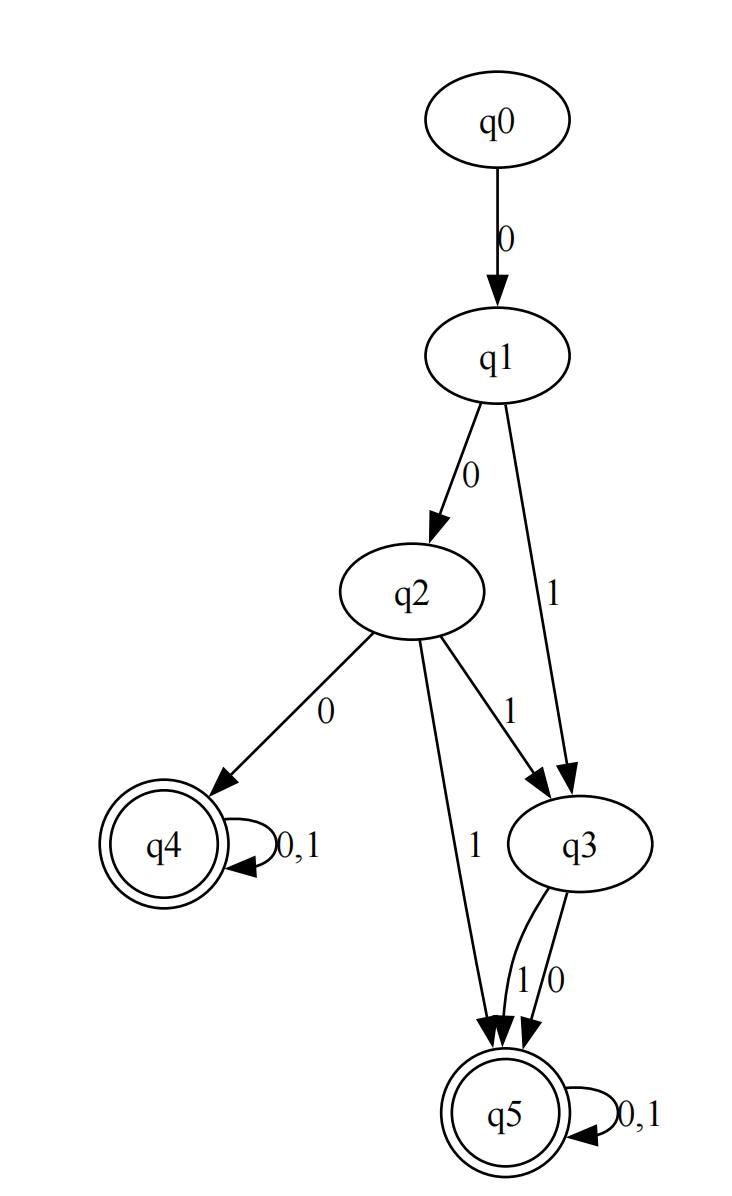
答案：False

Analysis:

A mapping reduction implies that there is a computable function that can transform instances of language A into instances of language B such that the acceptance by A is equivalent to acceptance by B. **However, the decidability of A does not necessarily imply the decidability of B.**

Language A being decidable means there is a Turing machine that accepts or rejects every input in finite time. However, B being TM-recognizable only guarantees that there is a Turing machine that can list all strings of B, not necessarily deciding membership of any string in finite time. Thus, even if A is decidable, B can still be undecidable.

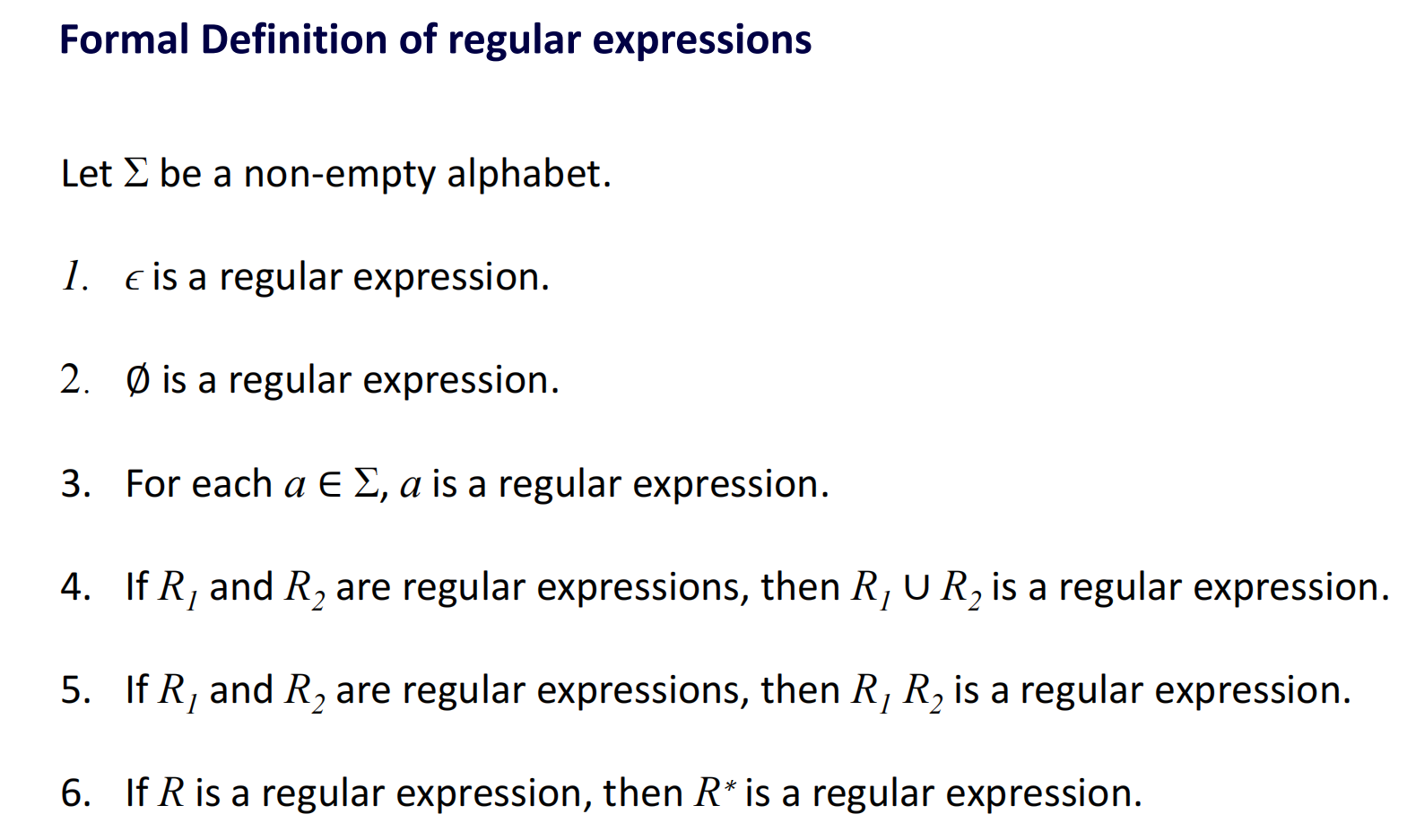
# Q2



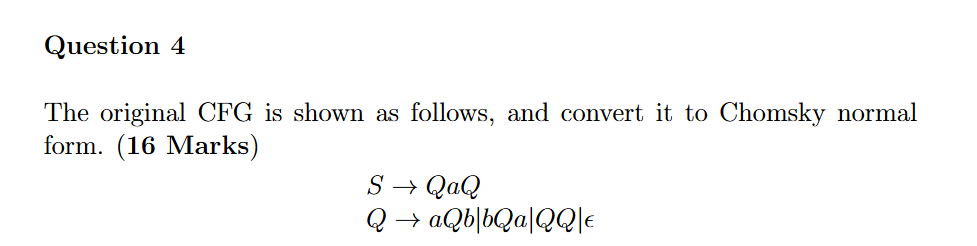
from graphviz import Digraph  
*# 创建一个有向图对象*dot = Digraph(comment=**'NFA for Language {w | w contains at least two 0s, or exactly two 1s}'**)  
*# 添加节点*states = [**'q0'**, **'q1'**, **'q2'**, **'q3'**, **'q4'**, **'q5'**]  
for state in states:  
 dot.node(state, state)  
*# 添加边（转移）*transitions = [  
 (**'q0'**, **'q1'**, **'0'**),  
 (**'q1'**, **'q2'**, **'0'**),  
 (**'q1'**, **'q3'**, **'1'**),  
 (**'q2'**, **'q3'**, **'1'**),  
 (**'q2'**, **'q4'**, **'0'**),  
 (**'q2'**, **'q5'**, **'1'**),  
 (**'q3'**, **'q5'**, **'0'**),  
 (**'q3'**, **'q5'**, **'1'**),  
 (**'q4'**, **'q4'**, **'0,1'**),  
 (**'q5'**, **'q5'**, **'0,1'**),  
]  
for src, dst, label in transitions:  
 dot.edge(src, dst, label=label)  
*# 设置起始和结束节点*dot.edge(**'q0'**, **'q0'**, style=**' invis'**, arrowhead=**'none'**) *# 虚线表示起始节点*dot.node(**'q4'**, **'q4'**, shape=**'doublecircle'**) *# 双圆圈表示接受状态*dot.node(**'q5'**, **'q5'**, shape=**'doublecircle'**) *# 双圆圈表示接受状态  
# 导出图像文件*dot.render(**'nfa\_language.gv'**, view=True)

提示：运行前要先安装Graphviz

# Q3(PPT L4 p21--27)



1. a∪(b(a∪b)\*a)b\*
2. (b∪aa)a\*bb\*



为了将给定的上下文无关文法（CFG）转换为乔姆斯基范式（Chomsky Normal Form, CNF），我们需要按照一系列步骤来处理。乔姆斯基范式要求每个产生式的右侧要么是两个非终结符的串联，要么是一个终结符，要么是ε。

原始CFG

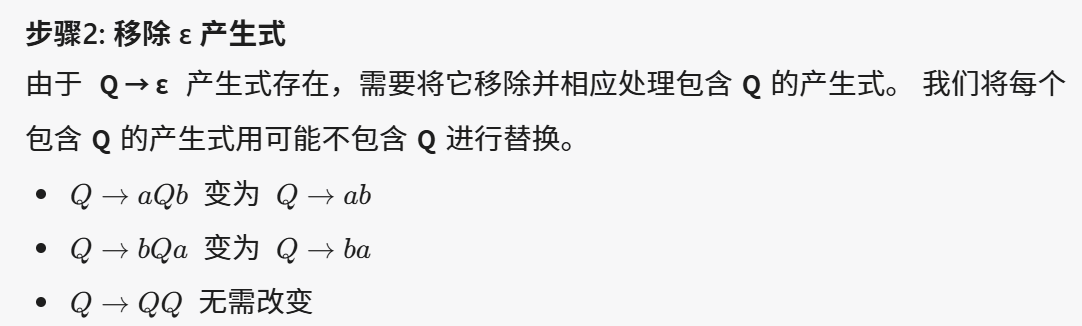
S → QaQ

Q → aQb | bQa | QQ | ε

步骤1: 引入新的初始符号

为了确保初始符号不出现在任何右侧，我们引入新的初始符号S0:

S0 → S



新的产生式：

S → QaQ

Q → aQb | bQa | QQ | ab | ba

步骤3: 移除单一非终结符产生式

`S → QaQ` 和 `Q → QQ` 都是由单一非终结符串联构成，符合CNF要求。

步骤4: 移除左递归和大于2个的非终结符产生式

我们确保每个右侧只有两个符号：

1. 将 `Q → aQb` 变为 `Q → aX1`, `X1 → Qb`

2. 将 `Q → bQa` 变为 `Q → bX2`, `X2 → Qa`

3. 将 `Q → ab` 拆分 `Q → aX3`, `X3 → b`

4. 将 `Q → ba` 拆分 `Q → bX4`, `X4 → a`

新的非终结符 X1, X2, X3, X4 引入：

S0 → S

S → QX1

X1 → aQ

Q → aX2 | bX3 | QQ | aX4 | bX5

X2 → Qb

X3 → Qa

X4 → b

X5 → a

这里我们重新命名，保证形式正确：

S0 → S

S → QX1

X1 → aQ

Q → aX2 | bX3 | QQ | aX4 | bX5

X2 → Qb

X3 → Qa

X4 → b

X5 → a

最终的Chomsky Normal Form (CNF):

S0 → S

S → QX1

X1 → aQ

Q → aX2 | bX3 | QQ | aX4 | bX5

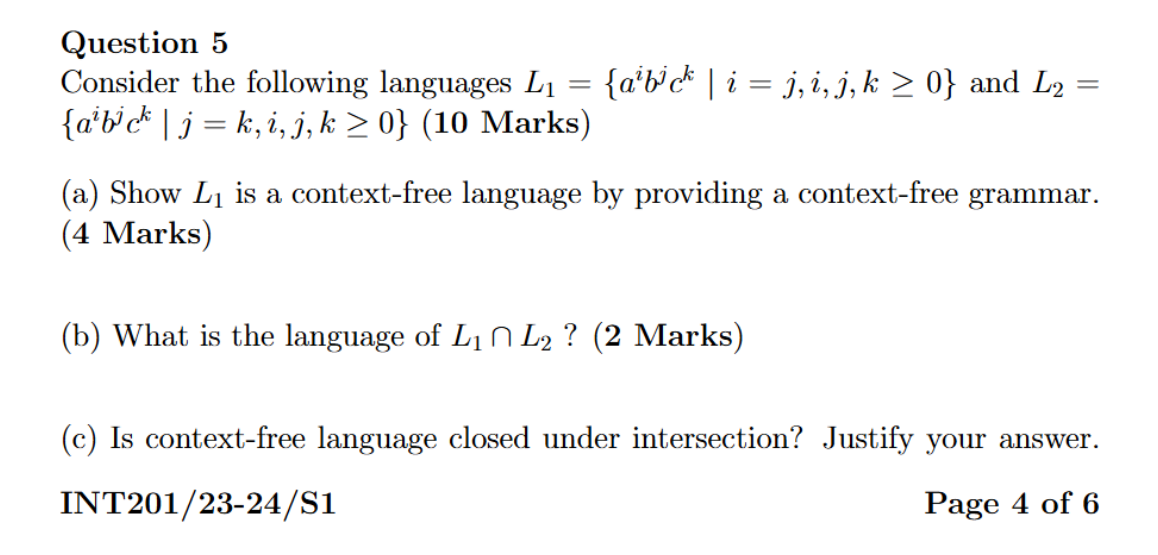
X2 → Qb

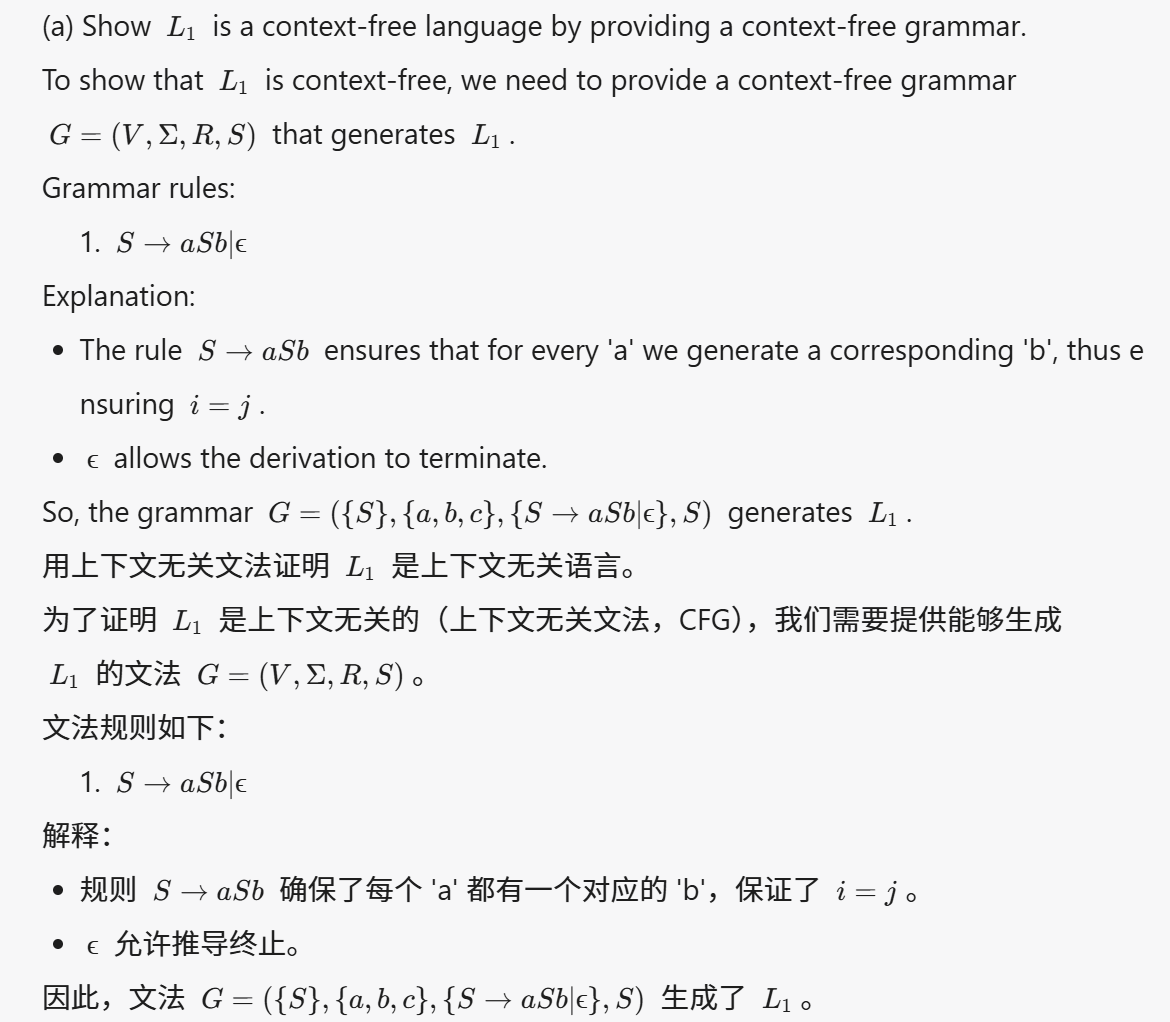
X3 → Qa

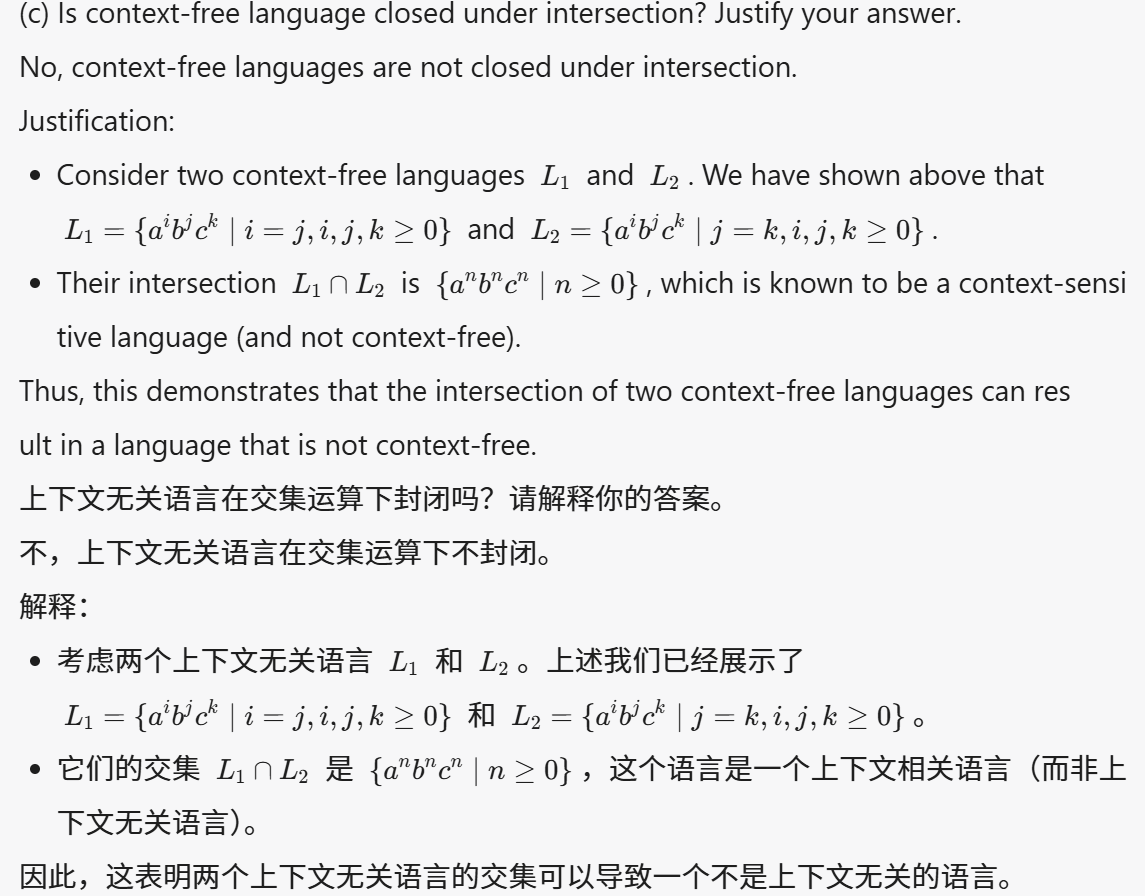
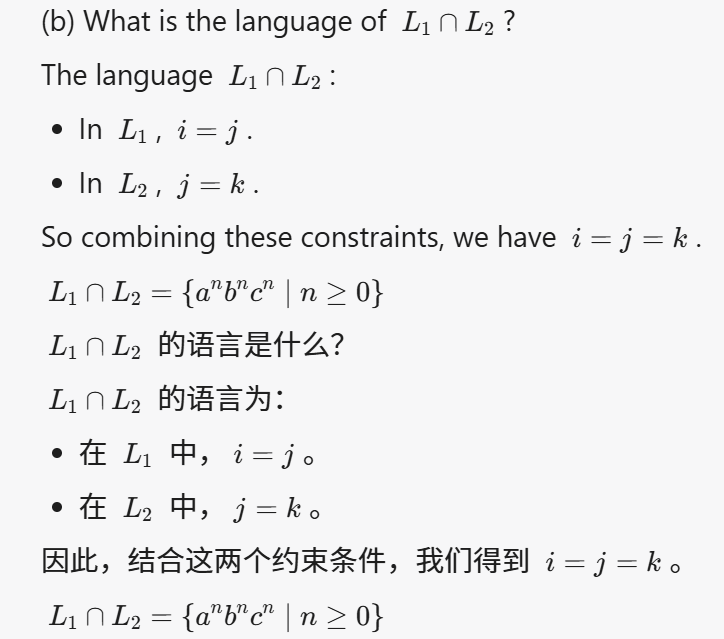
X4 → b

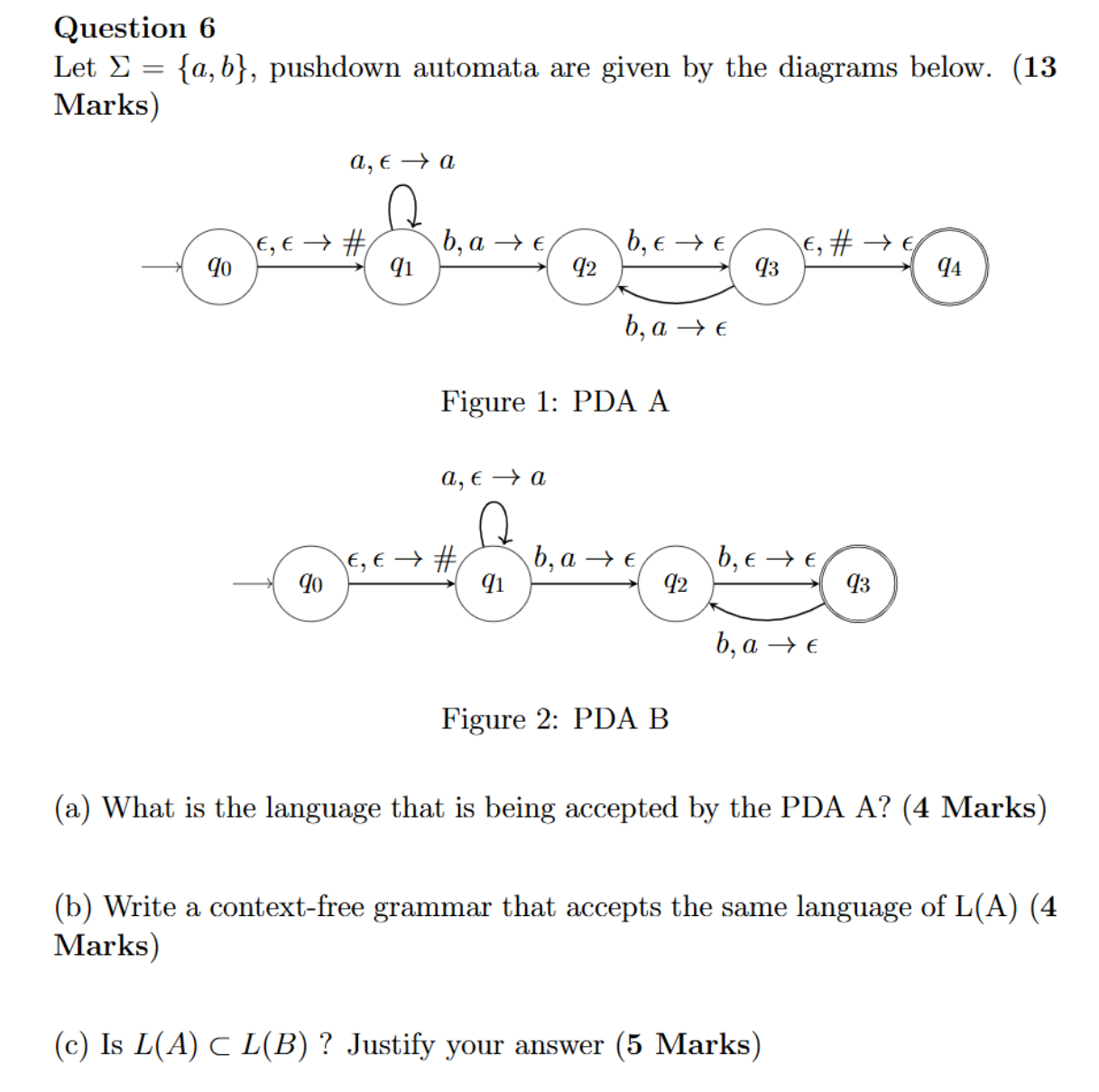
X5 → a

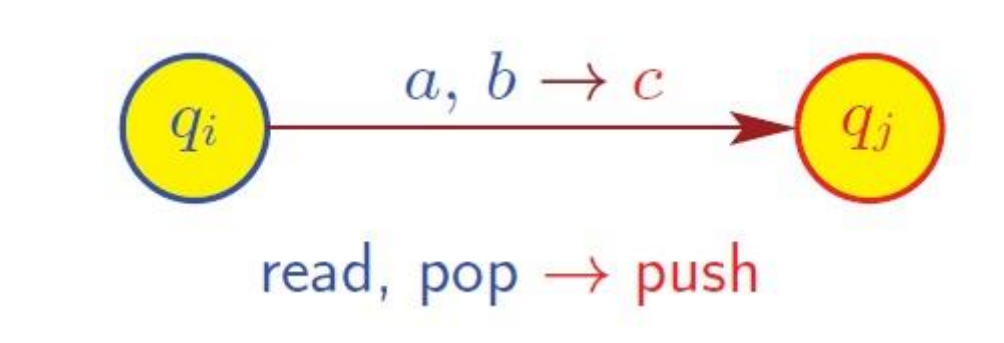
上述过程将原始CFG转换为了其等价的乔姆斯基范式 (CNF)，包含的每个步骤确保了每个字符正确且符合规范。

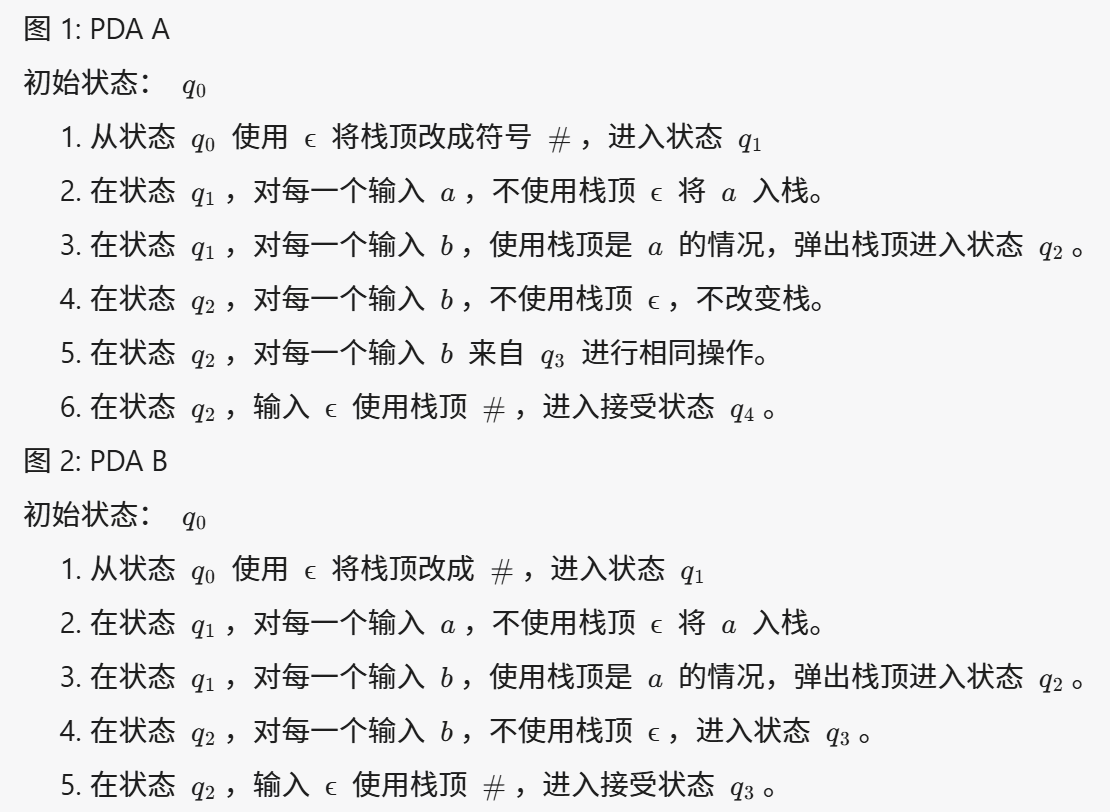






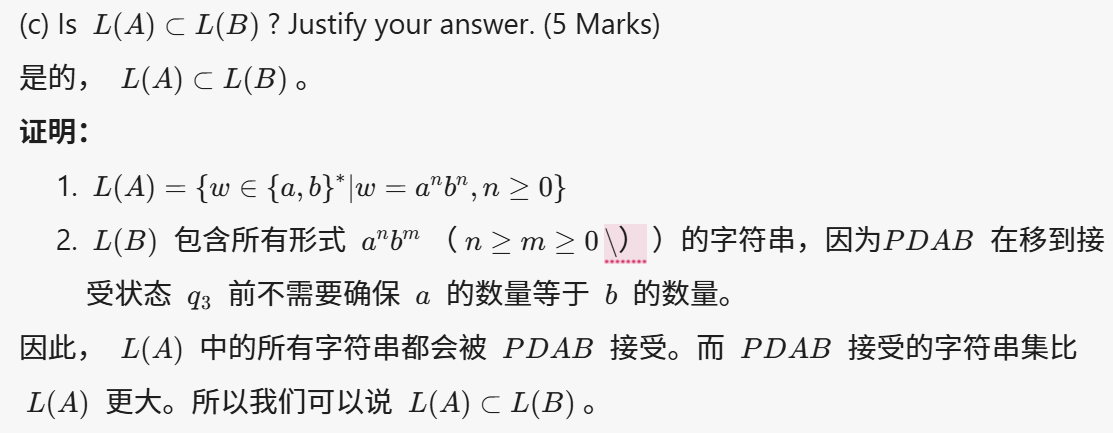




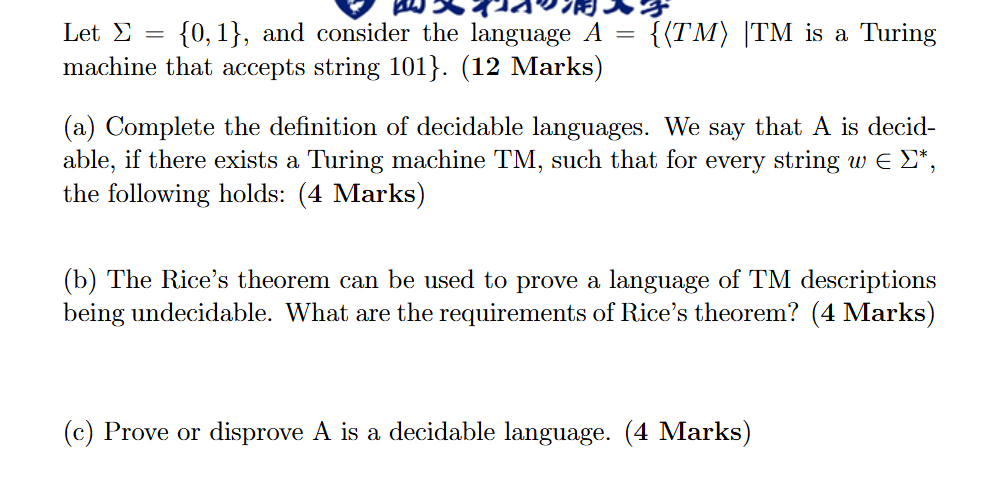


(a)PDA A接受的语言是所有形式为 IMG_256*anbn*的字符串，其中 IMG_257*n*≥0。这意味着语言中的每个字符串都包含等数量的 IMG_258*a* 和 IMG_259*b*，如 ""（空字符串），"ab"，"aabb" 等。

(b)S -> aSb | ε



# Q7



Σ = {0,1}，考虑语言 A = {〈TM〉 | TM 是接受字符串 101 的图灵机}。(12 分)

(a) 完成可判定语言的定义。我们说 A 是可判定的，如果存在一个图灵机 TM，使得对于每个字符串 w ∈ Σ\*，满足以下条件：(4 分)

解答：

A 是可判定的，如果存在一个图灵机 TM，使得对于每个字符串 w ∈ Σ\*，该图灵机最终在有限的步骤内停止并给出一个确定的结果（接受或拒绝）。换句话说，对于每个字符串 w ∈ Σ\*，TM 必须在有限时间内停止。

(b) 可以使用赖斯定理来证明TM描述的某个语言是不可判定的。赖斯定理的要求是什么？(4 分)

解答：

赖斯定理的要求是：

1. 语言必须是关于图灵机的语言，即语言中的每一个元素表示一个图灵机的编码。

2. 该语言必须是关于图灵机行为的非平凡性质，即它不能是所有图灵机的集合或者空集合，而必须描述一个图灵机的一些特定特性（例如是否接受某个特定的输入）。

3. 该性质必须仅依赖于图灵机接受的语言，而不能依赖于图灵机的具体描述或实现。

(c) 证明或反驳 A 是一个可判定的语言。(4 分)

解答：

A = {〈TM〉 | TM 是接受字符串 101 的图灵机} 不是一个可判定的语言。

证明：假设存在一个图灵机 M 能决定 A。也就是说，M 接受 〈TM〉 当且仅当 TM 接受字符串 101。然而，根据图灵机的不可判定性，不能总是确定一个任意图灵机 TM 是否会接受一个特定输入。在这个例子中，就是不能总是确定 TM 是否接受字符串 101。因此，语言 A 是不可判定的。

Let Σ = {0,1}, and consider the language A = {〈TM〉 | TM is a Turing machine that accepts string 101}. (12 Marks)

(a) Complete the definition of decidable languages. We say that A is decidable, if there exists a Turing machine TM, such that for every string w ∈ Σ\*, the following holds: (4 Marks)

Answer:

Language A is decidable if there exists a Turing machine TM such that for every string w ∈ Σ\*, that Turing machine will halt after a finite number of steps and provide a definite result (accept or reject). In other words, for every string w ∈ Σ\*, TM must halt in finite time.

(b) The Rice’s theorem can be used to prove a language of TM descriptions being undecidable. What are the requirements of Rice’s theorem? (4 Marks)

Answer:

The requirements of Rice’s theorem are:

1. The language must be a language about Turing machines, i.e., each element of the language represents the encoding of a Turing machine.

2. The language must be concerning a non-trivial property of Turing machine behavior, meaning it cannot be the set of all Turing machines or the empty set, but must describe some specific property of the Turing machine (for example, whether it accepts a particular input).

3. The property must depend only on the language accepted by the Turing machine and not on its specific description or implementation.

(c) Prove or disprove A is a decidable language. (4 Marks)

Answer:

A = {〈TM〉 | TM is a Turing machine that accepts string 101} is not a decidable language.

Proof: Assume there exists a Turing machine M that can decide A. This would mean M accepts 〈TM〉 if and only if TM accepts the string 101. However, it is not possible to always determine whether an arbitrary Turing machine TM will accept a specific input due to the undecidability of the halting problem. In this case, it means it is impossible to always determine if TM accepts the string 101. Therefore, the language A is undecidable.