### Undecidable Problems

不可判定问题的定义（Undecidable Problem）

不可判定问题是指不存在一个能够对所有输入都停止运行的图灵机（Turing Machine, TM）来识别该问题的相关语言。换句话说，对于不可判定问题，没有一个算法能够对所有可能的情况给出明确的“是”或“否”的答案。

不可识别问题（Unrecognizable Problem）

不可识别问题是指不存在一个图灵机能够识别该问题的相关语言。这意味着即使给予无限的时间和资源，也无法构建一个能够对所有输入都给出正确判断的图灵机。

图灵机语言的不可判定性

图灵机语言LTM是指所有能够被图灵机接受的字符串的集合，即LTM = { M, w : M是接受字符串w的图灵机 }。这个语言是不可判定的，但它是可识别的。这意味着虽然我们无法决定一个给定的图灵机和字符串对是否属于这个语言，但是我们可以识别出这个语言中的元素，只要给予足够的时间。

不可判定问题的不可解性（Undecidable problems are unsolvable）

通常，我们认为不可判定问题是无法解决的。这意味着对于不可判定问题，没有一个算法能够在有限的时间内给出答案。例如，如果一个图灵机已经运行了6周还没有给出响应，我们无法确定它是否能够接受给定的字符串。它可能不在该语言中，也可能在，只是需要等待更长的时间。

可判定语言与不可判定语言的区别（Decidable langauge VS Undecidable language）

- 可判定语言：对于可判定语言，存在一个算法，能够在有限的时间内对每个输入给出明确的答案。这意味着有一个确定的上限等待时间，算法在这个时间内一定能够给出结果。

- 不可判定语言：对于不可判定语言，即使给予无限的时间，也无法保证能够得到答案。等待时间没有上限，算法可能永远无法给出结果。

### Reducibility

归约（Reduction）是一种将一个问题转换成另一个问题的方法，使得第二个问题的解决方案可以用来解决第一个问题。以下是对归约概念的详细中文解释：

1. 归约的定义：

- 归约是一种技术，它允许我们将一个问题A转换成另一个问题B，这样解决B问题的方案也能用来解决A问题。如果问题A可以归约到问题B，那么任何解决B问题的解也能解决A问题。

2. 归约的性质：

- 如果A可以归约到B，那么A不会比B更难：这意味着如果存在一个算法能够将A问题的任何实例转换为B问题的实例，并且B问题的解可以用来解决A问题，那么A问题的难度不会超过B问题。

- 如果A可以归约到B，并且B是可判定的，那么A也是可判定的：可判定性意味着存在一个算法（或图灵机）能够在有限时间内确定问题的解。如果A可以转换为B，并且存在一个算法能够解决B问题，那么我们可以利用这个算法来解决A问题，因此A也是可判定的。

- 如果A可以归约到B，并且A是不可判定的，那么B也是不可判定的：如果A问题没有算法能够解决，而A可以转换为B，那么任何声称能解决B问题的算法也可以用来解决A问题，这与A的不可判定性相矛盾。因此，如果A是不可判定的，那么B也必须是不可判定的。

### Reduction method

1. 找到一个已知不可判定的问题A：

- 首先，你需要找到一个已经被证明是不可判定的问题A。不可判定意味着不存在图灵机（Turing Machine，简称TM）能够判定这个问题的所有实例。

2. 假设L是可判定的：

- 接下来，你假设语言L是可判定的。这意味着存在一个图灵机R能够判定L中的所有实例。

3. 构造一个图灵机S，使用R作为子程序来决定A：

- 利用假设中存在的图灵机R（能够判定L），你尝试构造另一个图灵机S，使得S能够利用R作为子程序来决定问题A。

4. 但是A是不可判定的：

- 由于问题A是已知不可判定的，这意味着不存在任何图灵机能够判定A。

5. 结论：L是不可判定的：

- 因为你假设了存在一个图灵机R能够判定L，并且尝试使用R来构造一个能够判定A的图灵机S，但我们知道A是不可判定的，这导致了矛盾。因此，你的假设（L是可判定的）必须是错误的。所以，结论是L也是不可判定的。

这种归约方法的核心在于，如果能够从已知不可判定的问题A构造出一个新的图灵机S来决定A，那么这个构造过程必须依赖于L的可判定性。如果A的不可判定性被保留下来，那么L的可判定性假设就会导致矛盾，从而证明L也是不可判定的。这是一种非常强大的证明技术，因为它允许我们利用已知的不可判定问题来证明新问题的不可判定性。

### Mapping reduction

映射归约（Mapping Reduction）是计算理论中用来描述两个问题（或语言）之间关系的一种归约方式。以下是对映射归约概念的详细中文解释：

1. 映射归约的定义：

- 假设A和B是两个语言，其中A定义在字母表Σ1上，即A是Σ1\*的子集；B定义在字母表Σ2上，即B是Σ2\*的子集。

2. 映射归约的条件：

- 如果存在一个可计算的函数f（可以被图灵机通过纸带模拟），将Σ1\*映射到Σ2\*，并且对于Σ1\*中的每一个字符串w，满足以下条件：w属于A当且仅当f(w)属于B，那么我们说A是映射归约到B的，记作A ≤m B。

3. 映射归约的含义：

- 映射归约意味着我们可以通过转换函数f将A语言中的字符串w映射到B语言中的字符串f(w)，并且保持了字符串的“属于性”。也就是说，如果w是A语言中的一个有效字符串，那么经过函数f映射后的f(w)也必须是B语言中的一个有效字符串；反之亦然。

4. 映射归约的作用：

- 映射归约的函数f被称为从A到B的归约。这个函数f提供了一种将问题A的实例转换为问题B的实例的方法。通过这种方式，如果我们知道如何解决B问题，那么我们也可以解决A问题，因为我们可以简单地将A问题的实例通过函数f转换为B问题的实例，然后解决B问题，从而间接解决A问题。

5. 映射归约的应用：

- 在计算理论中，映射归约是一种强有力的工具，它可以帮助我们证明某些问题的复杂性或不可解性。如果能够证明一个已知的难问题B可以映射归约到另一个问题A，那么我们可以推断出A问题至少和B问题一样难，或者A问题也是不可解的。

### 映射约归和一般约归

映射归约（Mapping Reduction）与一般归约（General Reduction）是计算理论中描述问题之间关系的两种不同概念。以下是对这两种归约方式的比较和解释：

1. 映射归约转一般归约：

- 映射归约可以被用来构造一般归约。映射归约是一种特殊形式的归约，其中存在一个明确定义的、可计算的函数f，将问题A的实例映射到问题B的实例。这种映射保证了如果w属于A，则f(w)属于B，反之亦然。

2. 一般归约的定义：

- 一般归约不要求有一个明确的映射函数，而是允许更灵活的转换方式。在一般归约中，我们可以设计一个算法或图灵机，它模拟另一个问题的解决过程，并根据这个模拟的结果来决定原问题的解。

3. 一般归约的灵活性：

- 与映射归约相比，一般归约提供了更多的灵活性。例如，在一般归约中，我们可以设计算法A，使得：

- 如果B接受f(w)，则A接受w。

- 如果B拒绝f(w)，则A拒绝w。

- 如果B在f(w)上循环，则A也循环。

- 此外，一般归约还允许其他选项，如：

- A在接受时B拒绝。

- A可以多次调用B。

- A可以在计算过程中根据B的解来调整自己的行为。

4. 一般归约的应用：

- 一般归约的这种灵活性使其在证明问题之间的复杂性关系时非常有用。通过构造一个算法，该算法**根据另一个问题B的解来决定问题A的解**，我们可以**展示A和B之间的复杂性关系，即使这种关系不是简单的一对一映射**。

5. 总结：

- 映射归约是一种特殊的一般归约，它通过一个明确的函数f来建立问题A和B之间的关系。而一般归约则更加灵活，允许算法根据另一个问题的解来决定当前问题的解，这种关系可以是复杂的、条件性的，甚至可以涉及多次调用或循环。

通过这种比较，我们可以看到一般归约提供了一种更广泛的方式来描述和证明问题之间的复杂性关系，而映射归约则是一般归约的一个特例，它通过一个明确的映射函数来实现问题之间的转换。

### 

如果语言A映射归约到语言B，并且语言B是可判定的，那么语言A也是可判定的。以下是

定理表述

证明过程

1. 假设：已知B是可判定的，即存在图灵机MB，对任何属于Σ2\*的字符串都能判定其是否属于B。

2. 构造图灵机MA：我们构造一个新的图灵机MA，用来判定语言A。MA的构造如下：

- MA接收一个属于Σ1\*的字符串w。

- MA计算函数f(w)，将w转换为一个新的字符串f(w)，该字符串属于Σ2\*。

- MA模拟图灵机MB在输入f(w)上的行为。

3. MA的行为：

- 如果MB接受f(w)，则MA接受w。

- 如果MB拒绝f(w)，则MA拒绝w。

- 如果MB在f(w)上无限循环，则MA也在w上无限循环。

4. 可判定性的证明：

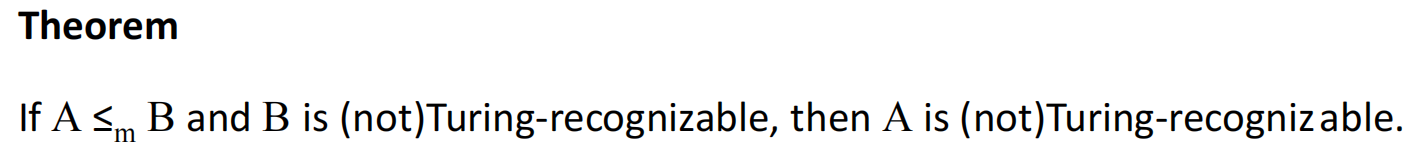
- 由于MB是可判定的，对于任何输入f(w)，MB都能在有限步内停机并给出接受或拒绝的判断。

- 因此，对于任何属于A的字符串w，MA最终会接受w（因为f(w)属于B，MB接受f(w)）。

- 对于任何不属于A的字符串w，MA最终会拒绝w（因为f(w)不属于B，MB拒绝f(w)）。

- 由于MA对所有输入都能在有限步内停机并给出接受或拒绝的判断，MA是可判定的。

5. 结论：由于我们构造了一个图灵机MA，它能够判定语言A，因此语言A是可判定的。



这个定理表述的是：如果语言A映射归约到语言B，并且语言B是图灵可识别的（Turing-recognizable），那么语言A也是图灵可识别的。以下是这个定理的中文详细解释和证明过程：

证明过程

1. 假设：已知B是图灵可识别的，即存在图灵机MB，对任何属于Σ2\*的字符串都能识别其是否属于B。

2. 构造图灵机MA：我们构造一个新的图灵机MA，用来识别语言A。MA的构造如下：

- MA接收一个属于Σ1\*的字符串w。

- MA计算函数f(w)，将w转换为一个新的字符串f(w)，该字符串属于Σ2\*。

- MA模拟图灵机MB在输入f(w)上的行为。

3. MA的行为：

- 如果MB接受f(w)，则MA接受w。

- 如果MB拒绝f(w)或在f(w)上无限循环，则MA拒绝w或在w上无限循环。

4. 图灵可识别性的证明：

- 由于MB是图灵可识别的，对于任何属于B的字符串f(w)，MB都能在有限步内停机并接受。

- 因此，对于任何属于A的字符串w，MA最终会接受w（因为f(w)属于B，MB接受f(w)）。

- 对于任何不属于A的字符串w，MA可能拒绝w（因为f(w)不属于B，MB拒绝f(w)），也可能无限循环（因为f(w)不属于B，MB在f(w)上无限循环）。

5. 结论：由于我们构造了一个图灵机MA，它能够识别语言A中的字符串（对于A中的字符串，MA最终会接受），因此语言A是图灵可识别的。

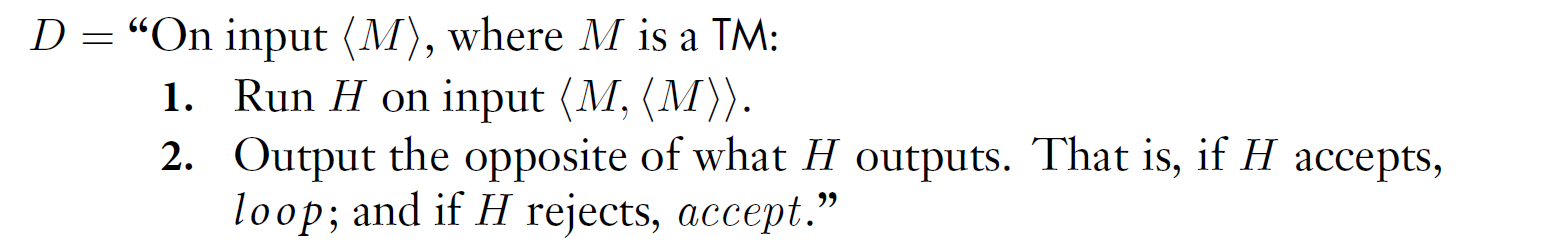
### Halting problem for TMs is undecidable

停机问题的不可判定性证明（反证法）

1. 假设停机问题是可判定的：

- 假设存在一个图灵机H，能够判定任何给定的图灵机TM和输入字符串是否会停机。

2. 构造图灵机D：

3. 提出问题：

- 问：D在输入<D>（即D的编码）时会停机吗？

4. 分析两种情况：

- 情况1：如果D在输入<D>时停机，根据H的假设，H会接受<D>。但根据D的定义，当H接受<D>时，D应该循环，这与D停机矛盾。

- 情况2：如果D在输入<D>时不停机，根据H的假设，H会拒绝<D>。但根据D的定义，当H拒绝<D>时，D应该接受（停机），这同样与D不停机矛盾。

5. 得出结论：

- 无论D在输入<D>时是否停机，我们都得到了矛盾，因此假设错误，停机问题是不可判定的。

### Emptiness of TMs is undecidable

图灵机空集问题的不可判定性证明（归约法）

1. 定义图灵机空集问题（ETM）：

𝐸𝑇𝑀 = {< 𝑀 > | 𝑀 𝑖𝑠 𝑎 𝑇𝑀 𝑎𝑛𝑑 𝐿(𝑀) = ∅}.

即所有图灵机M的编码的集合，这些图灵机接受的语言是空集。

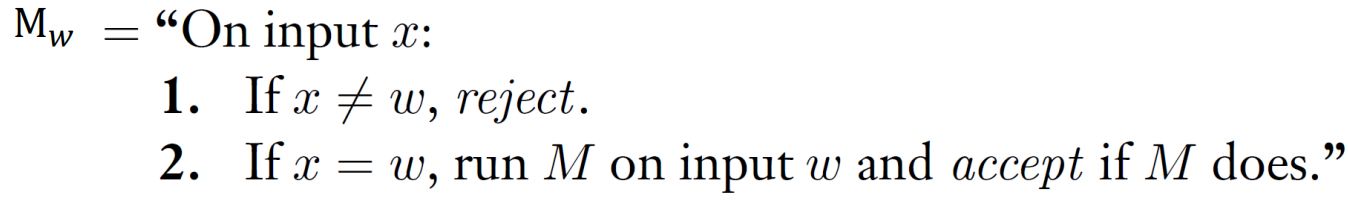
2. 直观理解：

- 我们需要证明对于任何字符串，图灵机都不会进入接受状态。但与上下文无关文法（CFG）或确定性有限自动机（DFA）的情况不同，我们难以枚举所有可能的推导。

3. 定理：ETM是不可判定的。

4. 从停机问题（ATM）归约：

- 对于给定的图灵机M和输入w，构造一个新的图灵机Mw，使得M接受w当且仅当L(Mw) ≠ ∅。



5. 构造Mw的过程：

- 如果M接受w：

- 在第一步，Mw拒绝所有除了w之外的字符串。

- 在第二步，Mw接受w，因此L(Mw) ≠ ∅。

- 如果L(Mw) ≠ ∅：

- 那么它拒绝所有除了w之外的字符串，并且只有在M接受w时才接受w。

- 因此，M接受w。

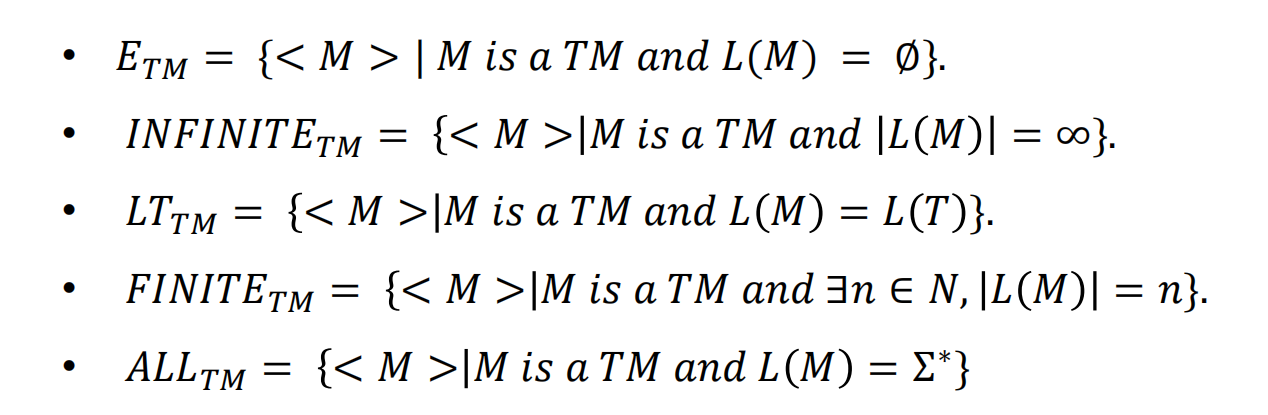
6. 证明过程：

- 通过构造Mw，我们将停机问题（ATM）归约到了图灵机空集问题（ETM）。由于停机问题是不可判定的，根据归约的性质，如果ATM是不可判定的，那么ETM也必须是不可判定的。

7. 得出结论：

- 因此，图灵机空集问题（ETM）是不可判定的。

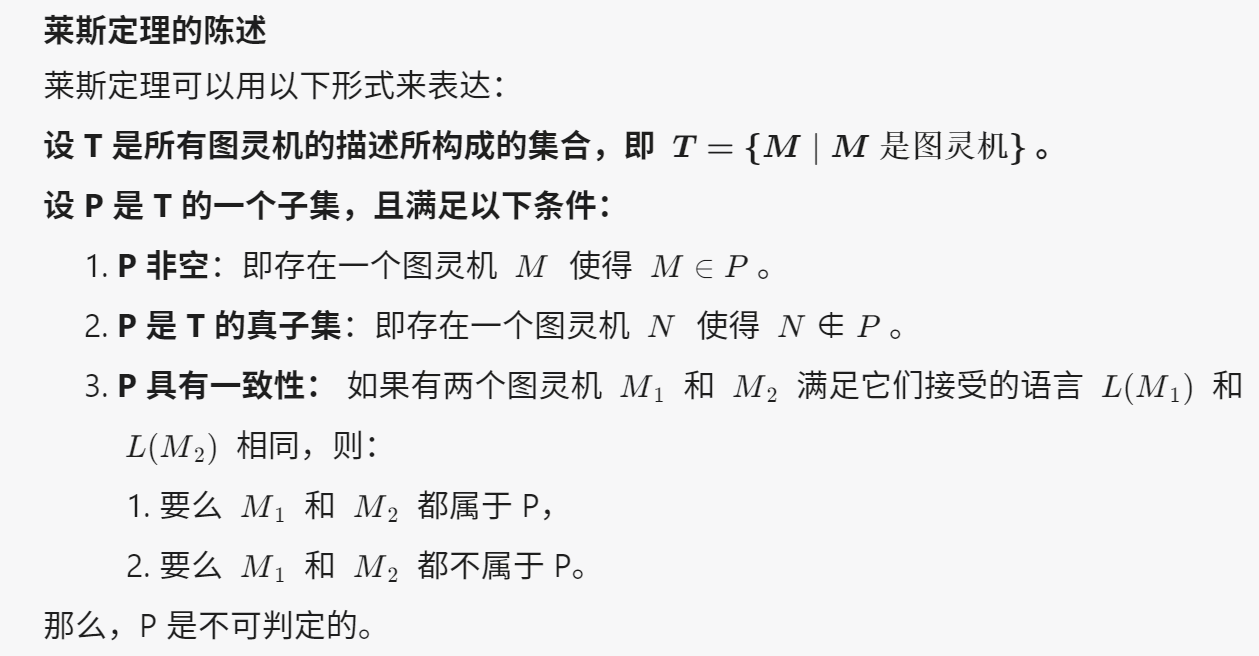
这个证明通过从停机问题（已知不可判定）归约到图灵机空集问题，展示了ETM的不可判定性。这种归约方法是一种强有力的证明技术，它允许我们利用已知的不可判定问题来证明新问题的不可判定性。



### How to prove them all at once?

### Rice’s Theorem

莱斯定理（Rice's Theorem）是计算理论中的一个重要定理，它指出了关于图灵机语言性质的一个非常强的结果。这个定理的核心思想是：对于任何非平凡的图灵机语言属性，判断一个图灵机是否拥有该属性是不可判定的。



莱斯定理的核心结论是：对于任何关于图灵机语言的非平凡性质，判断一个图灵机是否具备这个性质是不可判定的。

1. 非空：这意味着集合 P 里至少有一个图灵机，它的语言具有某种特性。例如，P 可以是那些接受某种特定语言的图灵机。

2. 真子集：P 并不包含所有图灵机的描述，说明 P 是关于某种图灵机语言特性（比如接受某种特定语言）的子集，并且其中的一部分图灵机满足该特性，而另一部分不满足。

3. 一致性条件：如果两个图灵机的语言相同（即它们所接受的语言相同），那么它们要么都属于 P，要么都不属于 P。这意味着 P 描述的是图灵机的某种语言性质，而这种性质对所有具有相同语言的图灵机是相同的。

理解不可判定性

通过这个定理，莱斯定理告诉我们，如果一个问题可以通过判断图灵机语言的某个特性来解决，而这个特性不是“所有图灵机都满足”或“没有图灵机满足”，那么这个问题是不可判定的。这意味着没有一个通用的算法可以判断一个任意给定的图灵机是否满足该语言特性。

### Rice定理的应用

Rice 定理是计算理论中的一个重要定理，它描述了关于图灵机语言集合的某些属性是否可判定的问题。以下是对您提供的内容的中文详细解释：

Rice 定理及其应用

设 P 是 T 的一个子集，满足以下条件：

1. P 不为空。

2. P 是 T 的真子集。

3. 对于任意两个图灵机 M1 和 M2，如果它们接受的语言相同（L(M1) = L(M2)），则：

(a) 要么 M1 和 M2 都在 P 中，

(b) 要么 M1 和 M2 都不在 P 中。

#### ETM 集合

ETM 是指所有接受空语言（不接受任何输入）的图灵机的集合。

1. 一个总是拒绝所有输入的图灵机足以在集合中。

2. 一个接受所有输入的图灵机不在集合中。

3. 如果 L(M1) = L(M2)，那么它们要么都接受空语言，要么都不接受空语言。因此，要么两者都在 P 中，要么都不在。

因此，根据 Rice 定理，ETM 是不可判定的。

INFINTETM 集合

INFINTETM 是指所有接受无穷多输入的图灵机的集合。

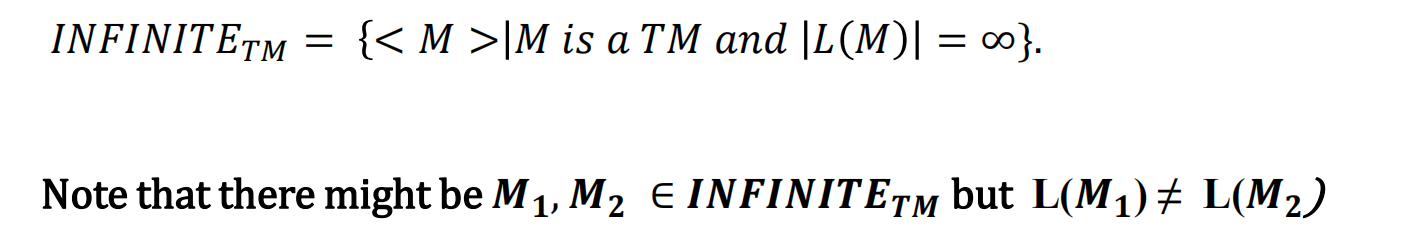
1. 一个总是接受所有输入的图灵机足以在集合中。

2. 一个总是拒绝所有输入的图灵机不在集合中。

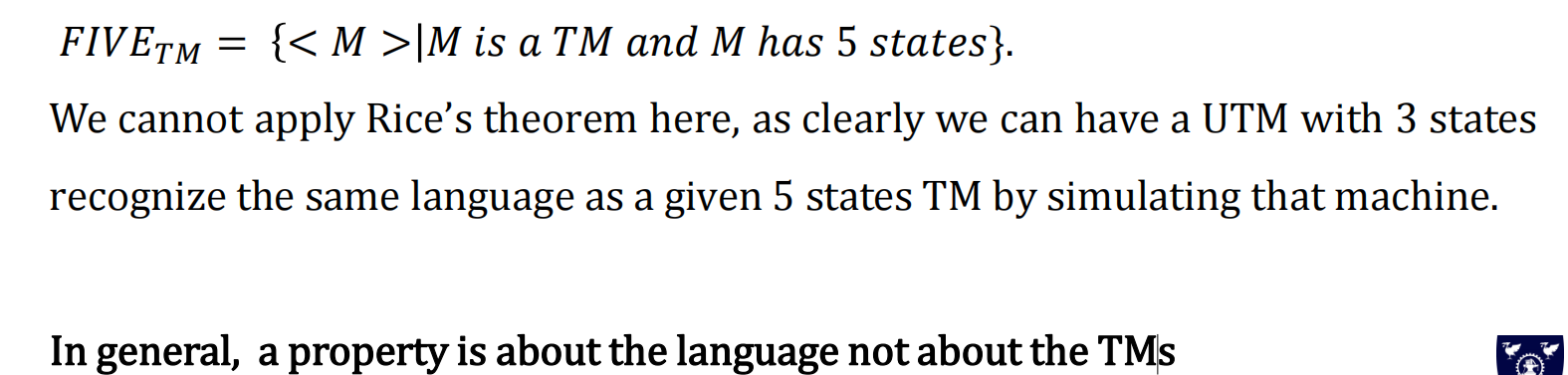
3. 如果 L(M1) = L(M2)，那么 |L(M1)| = |L(M2)|，所以它们要么都接受无穷多的输入，要么都接受有限个输入。因此，要么两者都在 P 中，要么都不在。

因此，根据 Rice 定理，INFINTETM 也是不可判定的。

需要注意的是，尽管 INFINTETM 中的图灵机都接受无穷多的输入，但它们接受的具体语言可能不同。



#### Rice 定理的非应用



1. FIVETM 集合的定义：

- FIVETM 是一个集合，包含所有具有 5 个状态的图灵机的编码（即图灵机的描述）。

2. 为什么应用 Rice 定理：

- Rice 定理适用于那些仅依赖于图灵机接受的语言（即语言属性）的集合。

- FIVETM 集合的定义是基于图灵机的状态数量，这是一个关于图灵机结构的属性，而不是关于它接受的语言的属性。

3. 反例说明：

- 假设我们有一个具有 5 个状态的图灵机 M1，它接受某种语言 L。

- 我们可以构造一个只有 3 个状态的通用图灵机 UTM，这个 UTM 能够模拟 M1 的行为，从而接受相同的语言 L。

- 这意味着，即使两个图灵机的状态数量不同，它们仍然可以接受相同的语言。

4. 一般性结论：

- 一个性质如果是关于图灵机接受的语言的，那么它可能适用于 Rice 定理。

- 但是，如果一个性质是关于图灵机本身的（比如状态数量、使用的符号等），那么 Rice 定理就不适用。

总结：

这句话的核心在于区分图灵机的“语言属性”和“结构属性”。**Rice 定理只适用于语言属性，而 FIVETM 集合是基于结构属性定义的，因此不能应用 Rice 定理**。通过构造一个具有不同状态数量的图灵机来接受相同语言的反例，进一步说明了这一点。

### Rice 定理的证明

要通过归约来证明莱斯定理，我们需要理解归约的概念以及莱斯定理本身。归约是一种证明方法，其中一个问题的可判定性或不可判定性被用来证明另一个问题的可判定性或不可判定性。**莱斯定理表明，对于任何非平凡的递归可枚举语言属性，其判定问题都是不可判定的。**

这里，递归可枚举语言是指可以被图灵机识别的语言，非平凡性质是指既不是所有递归可枚举语言都具有的性质，也不是没有任何递归可枚举语言具有的性质。

