2 Decidability
可判定性

杨雅君

yjyang@tju.edu.cn

智能与计算学部

2022

Outline

Outline

判定问题

问题与问题的实例

判定问题

问题与问题的实例

- 问题的实例：一个具体的问题
- 问题：抽象化描述，该问题全部实例的集合

判定问题

问题与问题的实例

- 问题的实例：一个具体的问题
- 问题：抽象化描述，该问题全部实例的集合

Definition (判定问题)

一个问题 Π 被称为判定问题，如果 Π 的每个实例 I 的解为“是”或者“否”.

判定问题

问题与问题的实例

- 问题的实例：一个具体的问题
- 问题：抽象化描述，该问题全部实例的集合

Definition (判定问题)

一个问题 Π 被称为判定问题，如果 Π 的每个实例 I 的解为“是”或者“否”.

例：语言 $L \subseteq \{0,1\}^*$ 是递归的吗?

例：图 $G = (V, E)$ 是哈密尔顿图吗?

判定问题

问题与问题的实例

- 问题的实例：一个具体的问题
- 问题：抽象化描述，该问题全部实例的集合

Definition (判定问题)

一个问题 Π 被称为判定问题，如果 Π 的每个实例 I 的解为“是”或者“否”.

例：语言 $L \subseteq \{0,1\}^*$ 是递归的吗?

例：图 $G = (V, E)$ 是哈密尔顿图吗?

编码，问题对应的语言

- 设 Π 是一个问题，对每个实例 I 用某个字母表 Σ 上的字符串编码.
- 问题 $\xrightarrow[\text{编码}]{\text{对实例}}$ 答案为“是”的实例的编码之集 $L \Longleftrightarrow$ 对应的语言

判定问题

编码，问题对应的语言

判定问题

编码，问题对应的语言

Definition (问题对应的语言)

问题 Π 的答案为“是”的实例的编码串之集称为 Π 对应的语言.

判定问题

编码, 问题对应的语言

Definition (问题对应的语言)

问题 Π 的答案为"是"的实例的编码串之集称为 Π 对应的语言.

判定问题与不可判定问题

Definition (判定问题)

问题 Π 是**可判定的**当且仅当 Π 对应的语言是递归的; 如果 Π 对应的语言不是递归, 则 Π 称为**不可判定问题**. 如果不可判定问题 Π 对应的语言是递归可枚举的, 则 Π 称为**半可判定的**.

判定问题

问题的分类

判定问题

问题的分类

- 问题
  - 非判定问题
  - 判定问题
    - 可判定问题
    - 不可判定问题
      - 半可判定问题
      - 非半可判定问题

递归语言的性质

Theorem

如果语言 L 是递归的, 则 $\overline{L}$ 也是递归语言.

递归语言的性质

Theorem

如果语言 L 是递归的, 则 $\overline{L}$ 也是递归语言.

证明：对于某个总停机的图灵机 M, 设 $L = L(M)$. 构造一个新的图灵机 $\overline{M}$, 使得 $\overline{L} = L(\overline{M})$.

递归语言的性质

Theorem

如果语言 L 是递归的, 则 $\overline{L}$ 也是递归语言.

证明：对于某个总停机的图灵机 M, 设 $L = L(M)$. 构造一个新的图灵机 $\overline{M}$, 使得 $\overline{L} = L(\overline{M})$.

- M 的接受状态都改变成 $\overline{M}$ 的非接受状态, 在这些状态中, M 将停机不接受.
- $\overline{M}$ 有一个新的接受状态 r, 没有从 r 出发的转移.
- 对于使得 M 没有转移(即 M 停机不接受)的 M 的非接受状态都添加一个到接受状态 r 的转移.

递归语言的性质

Theorem

如果语言 L 是递归的, 则 $\overline{L}$ 也是递归语言.

证明：对于某个总停机的图灵机 M, 设 $L = L(M)$. 构造一个新的图灵机 $\overline{M}$, 使得 $\overline{L} = L(\overline{M})$.

- M 的接受状态都改变成 $\overline{M}$ 的非接受状态, 在这些状态中, M 将停机不接受.
- $\overline{M}$ 有一个新的接受状态 r, 没有从 r 出发的转移.
- 对于使得 M 没有转移(即 M 停机不接受)的 M 的非接受状态都添加一个到接受状态 r 的转移.

由于 M 保证停机, 则 $\overline{M}$ 也保证停机. 而且, $\overline{M}$ 恰好接受 M 所不接受的串. 因此 $\overline{M}$ 接受 $\overline{L}$.

递归语言的性质

Theorem

如果一个语言 L 和它的补 $\overline{L}$ 都是递归可枚举的, 则 L 是递归的.

递归语言的性质

Theorem

如果一个语言 L 和它的补 $\overline{L}$ 都是递归可枚举的, 则 L 是递归的.

证明: 设 $L = L(M_1)$ 且 $\overline{L} = L(M_2)$. 构造图灵机 M 平行地模拟 M_1 和 M_2. 考虑 M 是 2 带图灵机, M 的一条纸带模拟 M_1 的纸带进行工作, M 的另一条纸带模拟 M_2 的纸带进行工作. M_1 和 M_2 的状态分别是 M 的状态的一个分量.

递归语言的性质

Theorem

如果一个语言 L 和它的补 $\overline{L}$ 都是递归可枚举的, 则 L 是递归的.

证明: 设 $L = L(M_1)$ 且 $\overline{L} = L(M_2)$. 构造图灵机 M 平行地模拟 M_1 和 M_2. 考虑 M 是 2 带图灵机, M 的一条纸带模拟 M_1 的纸带进行工作, M 的另一条纸带模拟 M_2 的纸带进行工作. M_1 和 M_2 的状态分别是 M 的状态的一个分量.

1. 如果 M 的输入 w 属于 L, 则 M_1 最终将接受. 此时, M 接受且停机.
2. 如果 w 不属于 L, 则 $w \in \overline{L}$, 所以 M_2 最终将接受. 当 M_2 接受时, M 停机不接受.

因此, 在所有输入上 M 都停机, $L(M)$ 恰好是 L. 因为 M 总是停机, 且 $L(M) = L$, 所以 L 是递归的.

图灵机的编码

设计图灵机的**二进制编码**, 使得可以认为具有输入字母表 $\{0,1\}$ 的每个图灵机都是二进制串.

为了把 TM $M=(Q,(0,1),\Gamma,\delta,q_1,q_2)$ 表示成二进制串, 必须把整数指派给状态、带符号以及方向 L 和 R.

图灵机的编码

设计图灵机的**二进制编码**, 使得可以认为具有输入字母表 $\{0,1\}$ 的每个图灵机都是二进制串.

为了把 TM $M=(Q,(0,1),\Gamma,\delta,q_1,q_2)$ 表示成二进制串, 必须把整数指派给状态、带符号以及方向 L 和 R.

1. 假设对于某个 r, 状态是 $q_1,q_2,\cdots,q_r$. 初始状态总是 q_1, q_2 是唯一的接受状态.
2. 假设对于某个 s, 带符号是 $X_1,X_2,\cdots,X_r$. X_1 永远是符号 0, X_2 永远是符号 1, X_3 永远是符号 $\sqcup$(空格). 其他带符号可以任意地指派给其余整数.
3. 我们把方向 L 称为 D_1, 方向 R 称为 D_2.

图灵机的编码

一旦选定了整数表示每个状态、符号以及方向, 就能编码转移函数 δ.

图灵机的编码

一旦选定了整数表示每个状态、符号以及方向, 就能编码转移函数 δ.

1. 假设对于某些整数 i, j, k, l, m, 一条转移规则是

$$\delta(q_i, X_j) = (q_k, X_l, D_m)$$

2. 把这条规则编码成串 $0^i 1 0^j 1 0^k 1 0^l 1 0^m$.
3. 注意 i, j, k, l, m 都至少为 1, 所以在单个转移的编码中, 没有两个以上连续的 1.

图灵机的编码

整个图灵机 M 的编码由所有转移的编码组成, 按照某种顺序排列, 用成对的 1 分隔: $C_1 11 C_2 11 \cdots C_{n-1} 11 C_n$, 其中每个 C_i 都是 M 的一个转移的编码.

图灵机的编码

Example (讨论图灵机 $M = (\{q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, \sqcup\}, \delta, q_1, q_2)$)

δ 包括规则：

1. $\delta(q_1, 1) = (q_3, 0, R)$:
2. $\delta(q_3, 0) = (q_1, 1, R)$:
3. $\delta(q_3, 1) = (q_2, 0, R)$:
4. $\delta(q_3, \sqcup) = (q_3, 1, L)$:

图灵机的编码

Example (讨论图灵机 $M = (\{q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, \sqcup\}, \delta, q_1, q_2)$)

δ 包括规则：

1. $\delta(q_1, 1) = (q_3, 0, R)$: 0100100010100
2. $\delta(q_3, 0) = (q_1, 1, R)$: 0001010100100
3. $\delta(q_3, 1) = (q_2, 0, R)$: 00010010010100
4. $\delta(q_3, \sqcup) = (q_3, 1, L)$: 0001000100010010

图灵机的编码

Example (讨论图灵机 $M = (\{q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, \sqcup\}, \delta, q_1, q_2)$)

δ 包括规则：

1. $\delta(q_1, 1) = (q_3, 0, R)$: 0100100010100
2. $\delta(q_3, 0) = (q_1, 1, R)$: 0001010100100
3. $\delta(q_3, 1) = (q_2, 0, R)$: 00010010010100
4. $\delta(q_3, \sqcup) = (q_3, 1, L)$: 0001000100010010

图灵机 M 的编码是：

0100100010100**11**000101010010011000100100101001**11**0001000100010010

图灵机的编码

Example (讨论图灵机 $M = (\{q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, \sqcup\}, \delta, q_1, q_2)$)

δ 包括规则：

1. $\delta(q_1, 1) = (q_3, 0, R)$: 0100100010100
2. $\delta(q_3, 0) = (q_1, 1, R)$: 0001010100100
3. $\delta(q_3, 1) = (q_2, 0, R)$: 00010010010100
4. $\delta(q_3, \sqcup) = (q_3, 1, L)$: 0001000100010010

图灵机 M 的编码是：

0100100010100110001010100100110001001001010011000100010010010

图灵机 M 共有多少种可能的编码?

图灵机的编码

Example (讨论图灵机 $M = (\{q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, \sqcup\}, \delta, q_1, q_2)$)

δ 包括规则：

1. $\delta(q_1, 1) = (q_3, 0, R)$: 0100100010100
2. $\delta(q_3, 0) = (q_1, 1, R)$: 0001010100100
3. $\delta(q_3, 1) = (q_2, 0, R)$: 00010010010100
4. $\delta(q_3, \sqcup) = (q_3, 1, L)$: 0001000100010010

图灵机 M 的编码是：

010010001010011000101010010011000100100101001100010001000100010010

图灵机 M 共有多少种可能的编码？ $4! = 24$ 种

与正则语言相关的可判定性问题

DFA 接受问题：

与正则语言相关的可判定性问题

DFA 接受问题： 检测一个特定的确定型有穷自动机是否接受给定的串

与正则语言相关的可判定性问题

***DFA* 接受问题：** 检测一个特定的确定型有穷自动机是否接受给定的串考虑语言

$$A_{\text{DFA}} = \{\langle B, w\rangle | B \text{ 是 DFA 并且接受输入串 } w\}$$

与正则语言相关的可判定性问题

***DFA* 接受问题：**检测一个特定的确定型有穷自动机是否接受给定的串考虑语言

$$A_{\text{DFA}} = \{\langle B, w\rangle | B \text{ 是 DFA 并且接受输入串 } w\}$$

问题“DFA B 是否接受输入 w”与问题“$\langle B, w\rangle$ 是否是 A_{DFA} 的元素”等价

与正则语言相关的可判定性问题

***DFA* 接受问题：** 检测一个特定的确定型有穷自动机是否接受给定的串考虑语言

$$A_{\text{DFA}} = \{\langle B, w\rangle | B \text{ 是 DFA 并且接受输入串 } w\}$$

问题“DFA B 是否接受输入 w”与问题“$\langle B, w\rangle$ 是否是 A_{DFA} 的元素”等价

Theorem

A_{DFA} 是一个可判定语言.

与正则语言相关的可判定性问题

***DFA* 接受问题：**检测一个特定的确定型有穷自动机是否接受给定的串考虑语言

$$A_{\text{DFA}} = \{\langle B, w\rangle | B \text{ 是 DFA 并且接受输入串 } w\}$$

问题“DFA B 是否接受输入 w”与问题“$\langle B, w\rangle$ 是否是 A_{DFA} 的元素”等价

Theorem

A_{DFA} 是一个可判定语言.

该定理证明了问题“一个给定的有穷自动机是否接受一个给定的串”是可判定的.

与正则语言相关的可判定性问题

Theorem

A_{DFA} 是一个可判定语言.

与正则语言相关的可判定性问题

Theorem

A_{DFA} 是一个可判定语言.

证明：只要设计一个判定 A_{DFA} 的图灵机 M 即可.

M 对于输入 $\langle B, w\rangle$, 其中 B 是 DFA, w 是输入字符串：

1. 在输入 w 上模拟 B
2. 如果模拟以接受状态结束, 则接受; 如果以非接受状态结束, 则拒绝.

与正则语言相关的可判定性问题

Theorem

A_{DFA} 是一个可判定语言.

证明：只要设计一个判定 A_{DFA} 的图灵机 M 即可.

M 对于输入 $\langle B, w\rangle$, 其中 B 是 DFA, w 是输入字符串：

1. 在输入 w 上模拟 B
2. 如果模拟以接受状态结束, 则接受; 如果以非接受状态结束, 则拒绝.

输入 $\langle B, w\rangle$ 的编码为 $B111w$. 当 M 收到输入时, 首先检查它是否正确地表示了 DFA B 和字符串 w, 如果不是, 则拒绝. 然后 M 直接执行模拟.

与正则语言相关的可判定性问题

对于非确定型有穷自动机，可以证明类似的定理. 设

$$A_{\text{NFA}} = \{\langle B, w\rangle | B \text{ 是 NFA 并且接受输入串 } w\}$$

Theorem

A_{NFA} 是一个可判定语言.

与正则语言相关的可判定性问题

对于非确定型有穷自动机，可以证明类似的定理. 设

$$A_{\text{NFA}} = \{\langle B, w\rangle | B \text{ 是 NFA 并且接受输入串 } w\}$$

Theorem

A_{NFA} 是一个可判定语言.

证明：构造一个判定 A_{NFA} 的图灵机 N. 可以考虑将 N 设计成与 M 一样. 下面说明一个新的想法：

与正则语言相关的可判定性问题

对于非确定型有穷自动机，可以证明类似的定理. 设

$$A_{\text{NFA}} = \{\langle B, w\rangle | B \text{ 是 NFA 并且接受输入串 } w\}$$

Theorem

A_{NFA} 是一个可判定语言.

证明：构造一个判定 A_{NFA} 的图灵机 N. 可以考虑将 N 设计成与 M 一样. 下面说明一个新的想法：

用 M 作为 N 的子程序. 因为 M 只接收 DFA 作为输入，故先将 N 作为输入所收到的 NFA 转换成 DFA，然后再将它传给 M.

与正则语言相关的可判定性问题

Theorem

A_{NFA} 是一个可判定语言.

证明：用 M 作为 N 的子程序. 因为 M 只接收 DFA 作为输入, 故先将 N 作为输入所收到的 NFA 转换成 DFA, 然后再将它传给 M.

与正则语言相关的可判定性问题

Theorem

A_{NFA} 是一个可判定语言.

证明：用 M 作为 N 的子程序. 因为 M 只接收 DFA 作为输入, 故先将 N 作为输入所收到的 NFA 转换成 DFA, 然后再将它传给 M.

N 对于输入 $\langle B, w\rangle$, 其中 B 是 NFA, w 是输入字符串：

1. 将 NFA B 转换成一个等价的 DFA C.
2. 在输入 $\langle C, w\rangle$ 上运行图灵机 M.
3. 如果 M 接受, 则**接受**, 否则**拒绝**.

与正则语言相关的可判定性问题

Theorem

A_{NFA} 是一个可判定语言.

证明：用 M 作为 N 的子程序. 因为 M 只接收 DFA 作为输入, 故先将 N 作为输入所收到的 NFA 转换成 DFA, 然后再将它传给 M.

N 对于输入 $\langle B, w\rangle$, 其中 B 是 NFA, w 是输入字符串：

1. 将 NFA B 转换成一个等价的 DFA C.
2. 在输入 $\langle C, w\rangle$ 上运行图灵机 M.
3. 如果 M 接受, 则**接受**, 否则**拒绝**.

第 2 步中, "运行图灵机 M" 的含义是：将 M 作为一个子程序加进 N 的设计中.

与正则语言相关的可判定性问题

令：$E_{\text{DFA}} = \{\langle A \rangle | A \text{ 是 DFA 且 } L(A) = \emptyset\}$

Theorem

E_{DFA} 是一个可判定语言.

与正则语言相关的可判定性问题

令：$E_{\text{DFA}} = \{\langle A \rangle | A \text{ 是 DFA 且 } L(A) = \emptyset\}$

Theorem

E_{DFA} 是一个可判定语言.

证明：DFA 接受一个串当且仅当：从起始状态出发，沿着此 DFA 的箭头方向，能够到达一个接收状态. 设计一个使用标记算法的图灵机 T.

与正则语言相关的可判定性问题

令：$E_{\text{DFA}} = \{\langle A\rangle | A \text{ 是 DFA 且 } L(A) = \emptyset\}$

Theorem

E_{DFA} 是一个可判定语言.

证明：DFA 接受一个串当且仅当：从起始状态出发，沿着此 DFA 的箭头方向，能够到达一个接收状态. 设计一个使用标记算法的图灵机 T. T 对于输入 $\langle A\rangle$，其中 A 是一个 DFA:

1. 标记 A 的起始状态.
2. 重复下列步骤，知道所有状态都被标记.
3. 对于一个状态，如果有一个到达它的转移是从某个已经标记过的状态出发的，则将其标记.
4. 如果没有接受状态被标记，则接受，否则拒绝.

与正则语言相关的可判定性问题

检查两个 DFA 是否识别同一个语言是可判定的. 设

$$EQ_{\text{DFA}} = \{\langle A, B\rangle | A \text{ 和 } B \text{ 都是 DFA 且 } L(A) = L(B)\}$$

Theorem

EQ_{DFA} 是一个可判定语言.

与正则语言相关的可判定性问题

检查两个 DFA 是否识别同一个语言是可判定的. 设

$$EQ_{\text{DFA}} = \{\langle A, B\rangle | A \text{ 和 } B \text{ 都是 DFA 且 } L(A) = L(B)\}$$

Theorem

EQ_{DFA} 是一个可判定语言.

证明：由 A 和 B 来构造一个新的 DFA C, 使得 C 只接受这样的串： A 或 B 接受但不是都接受. 显然, 如果 A 和 B 识别相同的语言, 则 C 不接受任何串. 即 C 的语言是

与正则语言相关的可判定性问题

检查两个 DFA 是否识别同一个语言是可判定的. 设

$$EQ_{\text{DFA}} = \{\langle A, B\rangle | A \text{ 和 } B \text{ 都是 DFA 且 } L(A) = L(B)\}$$

Theorem

EQ_{DFA} 是一个可判定语言.

证明：由 A 和 B 来构造一个新的 DFA C, 使得 C 只接受这样的串： A 或 B 接受但不是都接受. 显然, 如果 A 和 B 识别相同的语言, 则 C 不接受任何串. 即 C 的语言是

$$L(C) = (L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B))$$

与正则语言相关的可判定性问题

Theorem

EQ_{DFA} 是一个可判定语言.

与正则语言相关的可判定性问题

Theorem

EQ_{DFA} 是一个可判定语言.

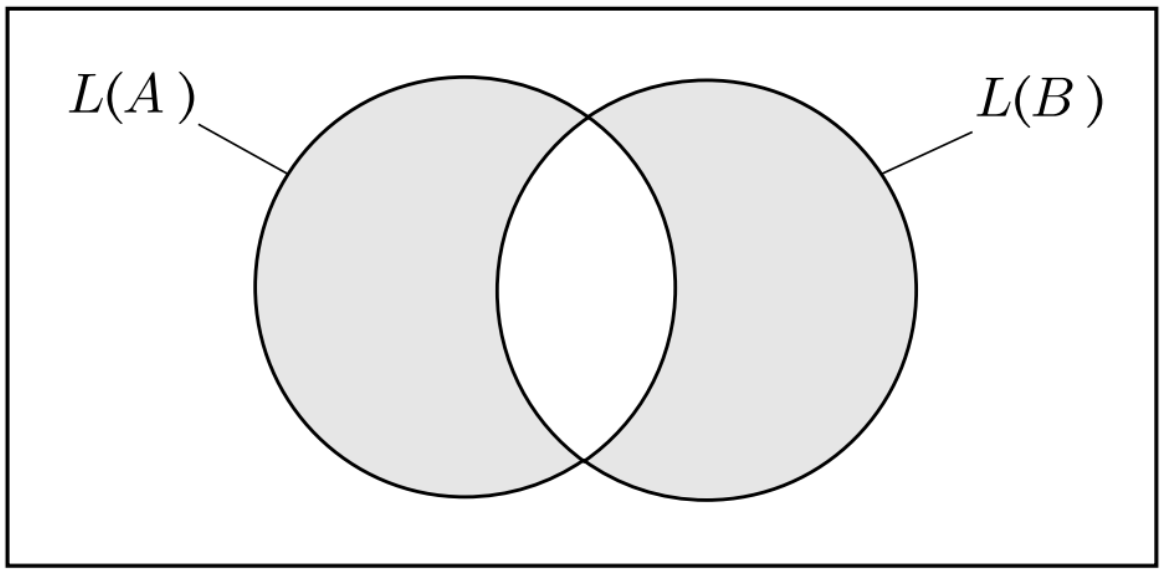
证明：$L(C)$ 表示 $L(A)$ 和 $L(B)$ 的对称差

与正则语言相关的可判定性问题

Theorem

EQ_{DFA} 是一个可判定语言.

证明：$L(C)$ 表示 $L(A)$ 和 $L(B)$ 的对称差



与正则语言相关的可判定性问题

Theorem

EQ_{DFA} 是一个可判定语言.

与正则语言相关的可判定性问题

Theorem

EQ_{DFA} 是一个可判定语言.

证明：检查 $L(C)$ 是否为空, 如果它是空的, $L(A)$ 与 $L(B)$ 必定相等.

与正则语言相关的可判定性问题

Theorem

EQ_{DFA} 是一个可判定语言.

证明：检查 $L(C)$ 是否为空, 如果它是空的, $L(A)$ 与 $L(B)$ 必定相等. F 对于输入 $\langle A, B\rangle$, 其中 A 和 B 都是 DFA.

1. 如上描述构造 DFA C.
2. 在输入 $\langle C\rangle$ 上运行上一定理中的图灵机 T.
3. 如果 T 接受, 则**接受**; 如果 T 拒绝, 则**拒绝**.

与上下文无关语言相关的可判定性问题

检查某个 CFG 是否派生一个特定的串

与上下文无关语言相关的可判定性问题

检查某个 CFG 是否派生一个特定的串

Theorem

A_{CFG} 是一个可判定语言.

与上下文无关语言相关的可判定性问题

检查某个 CFG 是否派生一个特定的串

Theorem

A_{CFG} 是一个可判定语言.

检查某个 CFG 是否派生一个特定的串

与上下文无关语言相关的可判定性问题

检查某个 CFG 是否派生一个特定的串

Theorem

A_{CFG} 是一个可判定语言.

检查某个 CFG 是否派生一个特定的串检查一个 CFG 是否不派生任何串.

与上下文无关语言相关的可判定性问题

检查某个 CFG 是否派生一个特定的串

Theorem

A_{CFG} 是一个可判定语言.

检查某个 CFG 是否派生一个特定的串检查一个 CFG 是否不派生任何串.

Theorem

E_{CFG} 是一个可判定语言.

与上下文无关语言相关的可判定性问题

检查某个 CFG 是否派生一个特定的串

Theorem

A_{CFG} 是一个可判定语言.

检查某个 CFG 是否派生一个特定的串检查一个 CFG 是否不派生任何串.

Theorem

E_{CFG} 是一个可判定语言.

上下文无关语言都可以用图灵机判定

与上下文无关语言相关的可判定性问题

检查某个 CFG 是否派生一个特定的串

Theorem

A_{CFG} 是一个可判定语言.

检查某个 CFG 是否派生一个特定的串检查一个 CFG 是否不派生任何串.

Theorem

E_{CFG} 是一个可判定语言.

上下文无关语言都可以用图灵机判定

Theorem

每个上下文无关语言都是可判定的.

Outline

非递归可枚举语言

受限制的图灵机

考虑图灵机 $M = (Q, \Sigma, \Gamma, \delta, q_1, q_2)$

1. $Q = q_1, q_2, \cdots, q_n$
2. $\Sigma = \{0, 1\}$
3. $\Gamma = \{0, 1, \sqcup\}$
4. q_1 为起始状态
5. q_2 为接受状态
6. 简便起见，省略拒绝状态 q_{reject}

非递归可枚举语言

受限制的图灵机

考虑图灵机 $M = (Q, \Sigma, \Gamma, \delta, q_1, q_2)$

1. $Q = q_1, q_2, \cdots, q_n$
2. $\Sigma = \{0, 1\}$
3. $\Gamma = \{0, 1, \sqcup\}$
4. q_1 为起始状态
5. q_2 为接受状态
6. 简便起见, 省略拒绝状态 q_{reject}

这样的图灵机是一个受限制的图灵机, 显然, 通过对字符的 $0, 1$ 编码, 非受限制的图灵机与受限制的图灵机等价.

非递归可枚举语言

我们可以对受限制的图灵机进行编码,

- 如果 w_i 为某个图灵机的编码, 则这个图灵机记为 M_i,
- 如果 w_i 不是图灵机的编码, 约定 w_i 也是一个图灵机的编码, 记为 M_i 且 $L(M_i) = \emptyset$,
- 于是有图灵机序列 $M_1, M_2, \cdots$.

非递归可枚举语言

我们可以对受限制的图灵机进行编码,

- 如果 w_i 为某个图灵机的编码, 则这个图灵机记为 M_i,
- 如果 w_i 不是图灵机的编码, 约定 w_i 也是一个图灵机的编码, 记为 M_i 且 $L(M_i) = \emptyset$,
- 于是有图灵机序列 $M_1, M_2, \cdots$.

因为由 0,1 构成的所有字符串的集合 $\{0,1\}^*$ 是可数集合, 我们可知

非递归可枚举语言

我们可以对受限制的图灵机进行编码,

- 如果 w_i 为某个图灵机的编码, 则这个图灵机记为 M_i,
- 如果 w_i 不是图灵机的编码, 约定 w_i 也是一个图灵机的编码, 记为 M_i 且 $L(M_i) = \emptyset$,
- 于是有图灵机序列 $M_1, M_2, \cdots$.

因为由 0,1 构成的所有字符串的集合 $\{0,1\}^*$ 是可数集合, 我们可知

Corollary

递归可枚举语言有可数个

非递归可枚举语言

我们可以对受限制的图灵机进行编码,

- 如果 w_i 为某个图灵机的编码, 则这个图灵机记为 M_i,
- 如果 w_i 不是图灵机的编码, 约定 w_i 也是一个图灵机的编码, 记为 M_i 且 $L(M_i) = \emptyset$,
- 于是有图灵机序列 $M_1, M_2, \cdots$.

因为由 0,1 构成的所有字符串的集合 $\{0,1\}^*$ 是可数集合, 我们可知

Corollary

递归可枚举语言有可数个

这说明不可判定问题比可判定问题要多得多.

对角化方法

Definition (可数)

如果集合 A 是有限的或者可与自然数集 N 一一对应, 则称 A 是可数的.

对角化方法

Definition (可数)

如果集合 A 是有限的或者可与自然数集 N 一一对应, 则称 A 是可数的.

Example

正有理数集合 $Q = \{\frac{m}{n} | m, n \in N\}$ 是可数的.

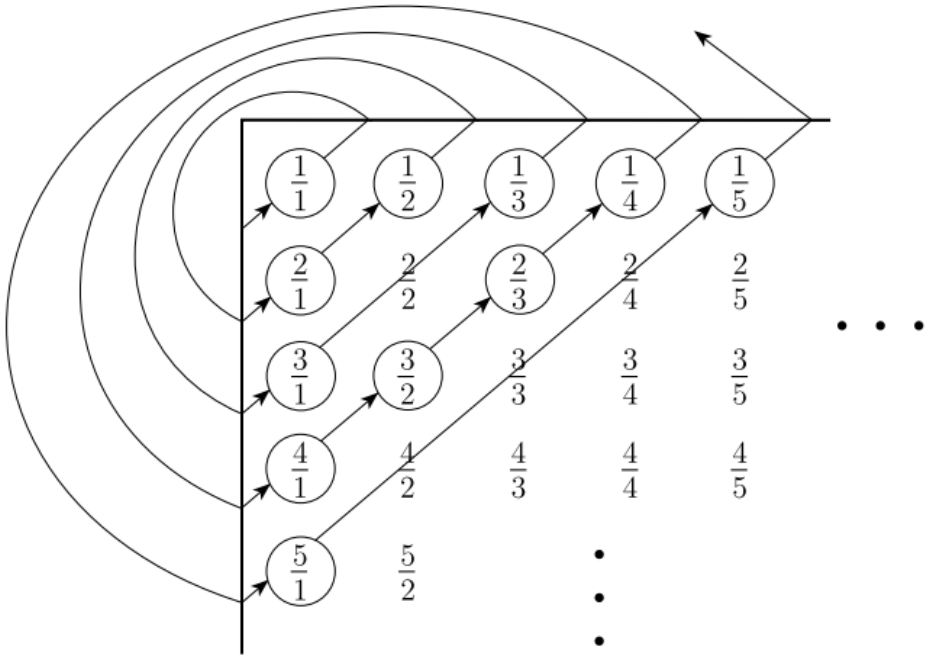
对角化方法

Definition (可数)

如果集合 A 是有限的或者可与自然数集 N 一一对应, 则称 A 是可数的.

Example

正有理数集合 $Q = \{\frac{m}{n} | m, n \in N\}$ 是可数的.



对角化方法

Example

实数集 R 是不可数的.

对角化方法

Example

实数集 R 是不可数的.

证明：采用康托对角线法进行证明.

因为 $f:(0,1)\to R$ 是双射函数, 令 $S=\{x|x\in R\bigwedge(0<x<1)\}$. 若能证明 S 是不可数集, 则 R 也是不可数集.

对角化方法

Example

实数集 R 是不可数的.

证明：采用康托对角线法进行证明.

因为 $f:(0,1)\to R$ 是双射函数, 令 $S=\{x|x\in R\bigwedge(0<x<1)\}$. 若能证明 S 是不可数集, 则 R 也是不可数集.

反证法. 假设 S 可数, 则 S 可表示为：$S=\{S_1,S_2,\cdots\}$, 其中 S_i 为 $(0,1)$ 区间的任一实数. 设 $S_i=0.y_1y_2y_3\cdots$, 其中 $y_i\in\{0,1,2,\cdots,9\}$, 设

对角化方法

Example

实数集 R 是不可数的.

证明：采用康托对角线法进行证明.

因为 $f:(0,1)\to R$ 是双射函数, 令 $S=\{x|x\in R\bigwedge(0<x<1)\}$. 若能证明 S 是不可数集, 则 R 也是不可数集.

反证法. 假设 S 可数, 则 S 可表示为：$S=\{S_1,S_2,\cdots\}$, 其中 S_i 为 $(0,1)$ 区间的任一实数. 设 $S_i=0.y_1y_2y_3\cdots$, 其中 $y_i\in\{0,1,2,\cdots,9\}$, 设

$$S_1=0.a_{11}a_{12}a_{13}\cdots a_{1n}\cdots$$
$$S_2=0.a_{21}a_{22}a_{23}\cdots a_{2n}\cdots$$
$$S_3=0.a_{31}a_{22}a_{33}\cdots a_{3n}\cdots$$
$$\cdots$$

对角化方法

Example

实数集 R 是不可数的.

对角化方法

Example

实数集 R 是不可数的.

证明：其次, 我们构造一个实数 $r = 0.b_1b_2b_3\cdots$ 使

对角化方法

Example

实数集 R 是不可数的.

证明：其次, 我们构造一个实数 $r = 0.b_1b_2b_3\cdots$ 使

$$b_j = \begin{cases} 1, & a_{jj} \neq 1, j = 1, 2, \cdots \\ 2, & a_{jj} = 1, j = 1, 2, \cdots \end{cases}$$

对角化方法

Example

实数集 R 是不可数的.

证明：其次, 我们构造一个实数 $r = 0.b_1b_2b_3\cdots$ 使

$$b_j = \begin{cases} 1, & a_{jj} \neq 1, j = 1, 2, \cdots \\ 2, & a_{jj} = 1, j = 1, 2, \cdots \end{cases}$$

这样, r 与所有实数 $S_1, S_2, \cdots, S_n, \cdots$ 不同, 因为它与 S_1 在位置 1 不同, 与 S_2 在位置 2 不同, $\cdots$, 等等. 这证明了 $r \notin S$, 产生矛盾. 因此 S 是不可数的, 进而 R 是不可数集.

非递归可枚举语言

根据康托定理, $(0,1)^*$ 上的所有语言的集合为不可数集. 因此,

Corollary

一定存在不能被任何图灵机识别的语言.

非递归可枚举语言

根据康托定理, $(0,1)^*$ 上的所有语言的集合为不可数集. 因此,

Corollary

一定存在不能被任何图灵机识别的语言.

我们可以找到一个非递归可枚举语言.

非递归可枚举语言

根据康托定理, $(0,1)^*$ 上的所有语言的集合为不可数集. 因此,

Corollary

一定存在不能被任何图灵机识别的语言.

我们可以找到一个非递归可枚举语言.

考虑问题 0: 第 i 个图灵机 M_i 不接受第 i 个字符串 w_i 吗?

非递归可枚举语言

根据康托定理, $(0,1)^*$ 上的所有语言的集合为不可数集. 因此,

Corollary

一定存在不能被任何图灵机识别的语言.

我们可以找到一个非递归可枚举语言.

考虑问题 0: 第 i 个图灵机 M_i 不接受第 i 个字符串 w_i 吗? 其对应语言为 L_d, 即

$$L_d = \{w_i | w_i \in \{0,1\}^*, w_i \notin L(M_i)\}$$

非递归可枚举语言

根据康托定理, $(0,1)^*$ 上的所有语言的集合为不可数集. 因此,

Corollary

一定存在不能被任何图灵机识别的语言.

我们可以找到一个非递归可枚举语言.

考虑问题 0: 第 i 个图灵机 M_i 不接受第 i 个字符串 w_i 吗? 其对应语言为 L_d, 即

$$L_d = \{w_i | w_i \in \{0,1\}^*, w_i \notin L(M_i)\}$$

Theorem

L_d 不是递归可枚举语言. 也就是说, 不存在接受语言 L_d 的图灵机.

非递归可枚举语言

Theorem

L_d 不是递归可枚举语言. 也就是说, 不存在接受语言 L_d 的图灵机.

非递归可枚举语言

Theorem

L_d 不是递归可枚举语言. 也就是说, 不存在接受语言 L_d 的图灵机.

证明: 反证法. 假设 L_d 是递归可枚举语言, 则存在某个图灵机 M, 使得 $L(M) = L_d$. 因为 L_d 是字母表 $\{0, 1\}$ 上的语言, 则 M 至少有一个编码 j, 即 $M = M_j$.

非递归可枚举语言

Theorem

L_d 不是递归可枚举语言. 也就是说, 不存在接受语言 L_d 的图灵机.

证明: 反证法. 假设 L_d 是递归可枚举语言, 则存在某个图灵机 M, 使得 $L(M) = L_d$. 因为 L_d 是字母表 $\{0, 1\}$ 上的语言, 则 M 至少有一个编码 j, 即 $M = M_j$.

现在询问 w_j 是否属于 $L(M_j)$.

非递归可枚举语言

Theorem

L_d 不是递归可枚举语言. 也就是说, 不存在接受语言 L_d 的图灵机.

证明: 反证法. 假设 L_d 是递归可枚举语言, 则存在某个图灵机 M, 使得 $L(M) = L_d$. 因为 L_d 是字母表 $\{0, 1\}$ 上的语言, 则 M 至少有一个编码 j, 即 $M = M_j$.

现在询问 w_j 是否属于 $L(M_j)$.

- 若 $w_j \in L(M_j)$, 则 $w_j \in L_d$. 由 L_d 定义, 可知 w_j 不能被 $L(M_j)$ 接受, 即 $w_j \notin L(M_j)$, 矛盾.

非递归可枚举语言

Theorem

L_d 不是递归可枚举语言. 也就是说, 不存在接受语言 L_d 的图灵机.

证明: 反证法. 假设 L_d 是递归可枚举语言, 则存在某个图灵机 M, 使得 $L(M) = L_d$. 因为 L_d 是字母表 $\{0, 1\}$ 上的语言, 则 M 至少有一个编码 j, 即 $M = M_j$.

现在询问 w_j 是否属于 $L(M_j)$.

- 若 $w_j \in L(M_j)$, 则 $w_j \in L_d$. 由 L_d 定义, 可知 w_j 不能被 $L(M_j)$ 接受, 即 $w_j \notin L(M_j)$, 矛盾.
- 若 $w_j \notin L(M_j)$, 则 $w_j \notin L_d$, 由 L_d 定义, 从而 $w_j \in L(M_j)$. 矛盾

非递归可枚举语言

Theorem

L_d 不是递归可枚举语言. 也就是说, 不存在接受语言 L_d 的图灵机.

证明: 反证法. 假设 L_d 是递归可枚举语言, 则存在某个图灵机 M, 使得 $L(M) = L_d$. 因为 L_d 是字母表 $\{0, 1\}$ 上的语言, 则 M 至少有一个编码 j, 即 $M = M_j$.

现在询问 w_j 是否属于 $L(M_j)$.

- 若 $w_j \in L(M_j)$, 则 $w_j \in L_d$. 由 L_d 定义, 可知 w_j 不能被 $L(M_j)$ 接受, 即 $w_j \notin L(M_j)$, 矛盾.
- 若 $w_j \notin L(M_j)$, 则 $w_j \notin L_d$, 由 L_d 定义, 从而 $w_j \in L(M_j)$. 矛盾

因此, 假设存在矛盾, L_d 不是递归可枚举语言.

通用图灵机

考虑问题 1：图灵机 M 是否接受一个字符串 w.

通用图灵机

考虑问题 1：图灵机 M 是否接受一个字符串 w. 其对应语言为 L_u, 即

$$L_u = \{\langle M, w\rangle | w \in \{0,1\}^*, M \text{ 是一个图灵机且接受 } w\}$$

通用图灵机

考虑问题 1：图灵机 M 是否接受一个字符串 w. 其对应语言为 L_u, 即

$$L_u = \{\langle M, w\rangle | w \in \{0,1\}^*, M \text{ 是一个图灵机且接受 } w\}$$

语言 L_u 称为通用语言. 如果存在一台图灵机 M_u, 使得 $L(M_u) = L_u$, 则称 M_u 为通用图灵机.

通用图灵机

考虑问题 1：图灵机 M 是否接受一个字符串 w. 其对应语言为 L_u, 即

$$L_u = \{\langle M, w\rangle | w \in \{0,1\}^*, M \text{ 是一个图灵机且接受 } w\}$$

语言 L_u 称为通用语言. 如果存在一台图灵机 M_u, 使得 $L(M_u) = L_u$, 则称 M_u 为通用图灵机.

Theorem

L_u 是递归可枚举语言, 即通用图灵机 M_u 是存在的.

通用图灵机

考虑问题 1：图灵机 M 是否接受一个字符串 w. 其对应语言为 L_u, 即

$$L_u = \{\langle M, w\rangle | w \in \{0,1\}^*, M \text{ 是一个图灵机且接受 } w\}$$

语言 L_u 称为通用语言. 如果存在一台图灵机 M_u, 使得 $L(M_u) = L_u$, 则称 M_u 为通用图灵机.

Theorem

L_u 是递归可枚举语言, 即通用图灵机 M_u 是存在的.

通用图灵机的存在性说明了现代电子计算机是可以制造出来的.

通用图灵机

Theorem

L_u 是递归可枚举语言，即通用图灵机 M_u 是存在的.

通用图灵机

Theorem

L_u 是递归可枚举语言, 即通用图灵机 M_u 是存在的.

证明：设计 M_u 为一个 3 带图灵机, 其中

1. 第 1 带存放 $\langle M, w\rangle$ 的编码, 即 $M111w$.
2. 第 2 带用于 M_u 模拟 M 在 w 上的运算.
3. 第 3 带始终存放 M 的当前状态 q_i (0^i).

通用图灵机

Theorem

L_u 是递归可枚举语言，即通用图灵机 M_u 是存在的.

证明：设计 M_u 为一个 3 带图灵机，其中

1. 第 1 带存放 $\langle M, w\rangle$ 的编码，即 $M111w$.
2. 第 2 带用于 M_u 模拟 M 在 w 上的运算.
3. 第 3 带始终存放 M 的当前状态 q_i (0^i).

M_u 的工作方式如下：

通用图灵机

Theorem

L_u 是递归可枚举语言，即通用图灵机 M_u 是存在的.

证明：设计 M_u 为一个 3 带图灵机，其中

1. 第 1 带存放 $\langle M, w\rangle$ 的编码，即 $M111w$.
2. 第 2 带用于 M_u 模拟 M 在 w 上的运算.
3. 第 3 带始终存放 M 的当前状态 q_i (0^i).

M_u 的工作方式如下：

1. M_u 检查输入是否正确.

通用图灵机

Theorem

L_u 是递归可枚举语言，即通用图灵机 M_u 是存在的.

证明：设计 M_u 为一个 3 带图灵机，其中

1. 第 1 带存放 $\langle M, w\rangle$ 的编码，即 $M111w$.
2. 第 2 带用于 M_u 模拟 M 在 w 上的运算.
3. 第 3 带始终存放 M 的当前状态 q_i (0^i).

M_u 的工作方式如下：

1. M_u 检查输入是否正确.
2. 初始化：
   - 把第 1 带上的 w 抄写到第 2 带上，读写头指向 w 的第 1 个符号；
   - 在第 3 带上打印 0 (q_1).

通用图灵机

Theorem

L_u 是递归可枚举语言, 即通用图灵机 M_u 是存在的.

1. M_u 检查输入是否正确.
2. 初始化:
3. M_u 模拟 M 的每一个动作, 若当前 M 处于状态 q_i (第 3 带上为 0^i), 第 2 带读写头读入的符号为 x_j, 若 δ 有定义 $\delta(q_i, x_j) = (q_k, x_l, D_m)$ 则 M_u 在第 1 带上找其编码 $0^i10^j10^k10^l10^m$, 然后执行:

通用图灵机

Theorem

L_u 是递归可枚举语言, 即通用图灵机 M_u 是存在的.

1. M_u 检查输入是否正确.
2. 初始化:
3. M_u 模拟 M 的每一个动作, 若当前 M 处于状态 q_i (第 3 带上为 0^i), 第 2 带读写头读入的符号为 x_j, 若 δ 有定义 $\delta(q_i, x_j) = (q_k, x_l, D_m)$ 则 M_u 在第 1 带上找其编码 $0^i10^j10^k10^l10^m$, 然后执行:
    1. 抹去第 3 带上的 0^i 并打印 0^k (状态 q_k).

通用图灵机

Theorem

L_u 是递归可枚举语言, 即通用图灵机 M_u 是存在的.

1. M_u 检查输入是否正确.
2. 初始化:
3. M_u 模拟 M 的每一个动作, 若当前 M 处于状态 q_i (第 3 带上为 0^i), 第 2 带读写头读入的符号为 x_j, 若 δ 有定义 $\delta(q_i, x_j) = (q_k, x_l, D_m)$ 则 M_u 在第 1 带上找其编码 $0^i10^j10^k10^l10^m$, 然后执行:
   1. 抹去第 3 带上的 0^i 并打印 0^k (状态 q_k).
   2. 第 2 带读写头注视的方格上打印 0^l (符号 x_l).

通用图灵机

Theorem

L_u 是递归可枚举语言, 即通用图灵机 M_u 是存在的.

1. M_u 检查输入是否正确.
2. 初始化:
3. M_u 模拟 M 的每一个动作, 若当前 M 处于状态 q_i (第 3 带上为 0^i), 第 2 带读写头读入的符号为 x_j, 若 δ 有定义 $\delta(q_i, x_j) = (q_k, x_l, D_m)$ 则 M_u 在第 1 带上找其编码 $0^i10^j10^k10^l10^m$, 然后执行:
   1. 抹去第 3 带上的 0^i 并打印 0^k (状态 q_k).
   2. 第 2 带读写头注视的方格上打印 0^l (符号 x_l).
   3. 第 2 带读写头根据 $m = 1$ 或 $m = 2$, 执行向左移动或者向右移动.

通用图灵机

Theorem

L_u 是递归可枚举语言, 即通用图灵机 M_u 是存在的.

1. M_u 检查输入是否正确.
2. 初始化:
3. M_u 模拟 M 的每一个动作, 若当前 M 处于状态 q_i (第 3 带上为 0^i), 第 2 带读写头读入的符号为 x_j, 若 δ 有定义 $\delta(q_i, x_j) = (q_k, x_l, D_m)$ 则 M_u 在第 1 带上找其编码 $0^i10^j10^k10^l10^m$, 然后执行:
   1. 抹去第 3 带上的 0^i 并打印 0^k (状态 q_k).
   2. 第 2 带读写头注视的方格上打印 0^l (符号 x_l).
   3. 第 2 带读写头根据 $m = 1$ 或 $m = 2$, 执行向左移动或者向右移动.
4. 如果 δ 无定义(第二带上找不到编码), 则 M_u 停机.

通用图灵机

通用图灵机的伟大贡献：

1. 通用图灵机 M_u 是现代电子计算机的抽象数学模型，证明了现在电子计算机是可以制造出来的.
2. 计算机硬件各种不同的设计都是等价的，它们的能力都一样，但性能有所差异.
3. 为以后高级语言的解释器奠定了基础.
4. 可以作为操作语义.

不可判定语言

Theorem

通用语言 L_u 不是递归语言，即 L_u 是不可判定语言.

不可判定语言

Theorem

通用语言 L_u 不是递归语言，即 L_u 是不可判定语言.

证明：根据递归语言的性质，只需证明 $\overline{L_u}$ 不是递归语言即可.

不可判定语言

Theorem

通用语言 L_u 不是递归语言，即 L_u 是不可判定语言.

证明：根据递归语言的性质，只需证明 $\overline{L_u}$ 不是递归语言即可.

反证法. 假设 $\overline{L_u}$ 是递归的，则存在一台总停机的图灵机 M 接受 $\overline{L_u}$，从而可以构造一台图灵机 M' 接受 L_d. 这与 L_d 是非递归可枚举语言矛盾.

不可判定语言

Theorem

通用语言 L_u 不是递归语言，即 L_u 是不可判定语言.

证明：根据递归语言的性质，只需证明 $\overline{L}_u$ 不是递归语言即可.

反证法. 假设 $\overline{L}_u$ 是递归的，则存在一台总停机的图灵机 M 接受 $\overline{L}_u$，从而可以构造一台图灵机 M' 接受 L_d. 这与 L_d 是非递归可枚举语言矛盾.

因为 $L(M) = \overline{L}_u$，如下图所示，可以将图灵机 M 改造成 M'，M' 接受 L_d 的工作方式如下:

不可判定语言

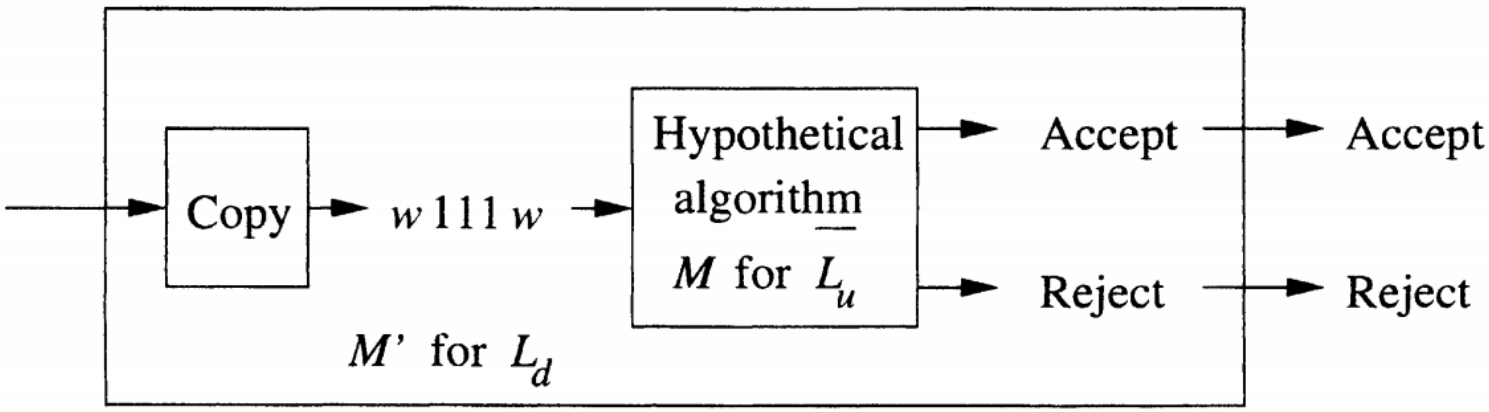
Theorem

通用语言 L_u 不是递归语言，即 L_u 是不可判定语言.

证明：根据递归语言的性质，只需证明 $\overline{L}_u$ 不是递归语言即可.

反证法. 假设 $\overline{L}_u$ 是递归的，则存在一台总停机的图灵机 M 接受 $\overline{L}_u$，从而可以构造一台图灵机 M' 接受 L_d. 这与 L_d 是非递归可枚举语言矛盾.

因为 $L(M) = \overline{L}_u$，如下图所示，可以将图灵机 M 改造成 M'，M' 接受 L_d 的工作方式如下:



不可判定语言

Theorem

通用语言 L_u 不是递归语言，即 L_u 是不可判定语言.

不可判定语言

Theorem

通用语言 L_u 不是递归语言, 即 L_u 是不可判定语言.

证明:

1. 给定串 w 作为输入, 在检查了 w 中不含连续三个 1 之后(若 w 含 111, 则立即拒绝 w), M' 把输入改成 $w111w$.

不可判定语言

Theorem

通用语言 L_u 不是递归语言, 即 L_u 是不可判定语言.

证明:

1. 给定串 w 作为输入, 在检查了 w 中不含连续三个 1 之后(若 w 含 111, 则立即拒绝 w), M' 把输入改成 $w111w$.
2. M' 在新的输入上模拟 M. 若 w 是图灵机的编码 w_i, 则 M' 确定 M_i 是否接受 w_i. M 接受 $\overline{L}_u$, 所以 M 接受当且仅当 M_i 不接受 w_i, 即 $w_i \in L_d$.

不可判定语言

Theorem

通用语言 L_u 不是递归语言, 即 L_u 是不可判定语言.

证明:

1. 给定串 w 作为输入, 在检查了 w 中不含连续三个 1 之后(若 w 含 111, 则立即拒绝 w), M' 把输入改成 $w111w$.
2. M' 在新的输入上模拟 M. 若 w 是图灵机的编码 w_i, 则 M' 确定 M_i 是否接受 w_i. M 接受 $\overline{L}_u$, 所以 M 接受当且仅当 M_i 不接受 w_i, 即 $w_i \in L_d$.

因此 M' 接受 w 当且仅当 $w_i \in L_d$, 因此 M' 存在, 其与 L_d 是非递归可枚举语言矛盾. 因此, L_u 不是递归语言.

不可判定语言

考虑图灵停机问题：图灵机 M 在输入串 w 上停机吗?

不可判定语言

考虑图灵停机问题：图灵机 M 在输入串 w 上停机吗?

$$L_h = \{\langle M, w\rangle | w \in \{0,1\}^*, M \text{ 是一个图灵机且对 } w \text{ 停机}\}$$

不可判定语言

考虑图灵停机问题：图灵机 M 在输入串 w 上停机吗?

$$L_h = \{\langle M, w\rangle | w \in \{0,1\}^*, M \text{ 是一个图灵机且对 } w \text{ 停机}\}$$

Theorem

L_h 不是递归语言，即图灵停机问题是不可判定的.

不可判定语言

考虑图灵停机问题：图灵机 M 在输入串 w 上停机吗?

$$L_h = \{\langle M, w\rangle | w \in \{0,1\}^*, M \text{ 是一个图灵机且对 } w \text{ 停机}\}$$

Theorem

L_h 不是递归语言，即图灵停机问题是不可判定的.

证明：反证法. 假设 L_h 是递归的，则存在一台总停机的图灵机 M 接受 L_h，从而可以构造一台图灵机 M' 接受 L_d，构造方法与上一定理类似，从而得出矛盾，因此 L_h 不是递归语言.

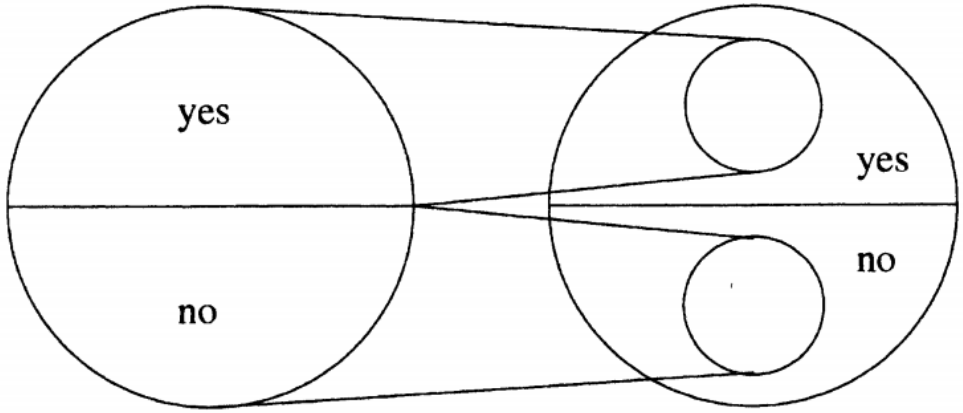
Outline

归约

Definition (归约)

如果有一个算法把问题 P_1 的全部实例转化成问题 P_2 的具有相同答案的实例, 则称 P_1 归约到 P_2.

- 必须把 P_1 具有"是"回答的任何实例都变成 P_2 具有"是"回答的实例
- 必须把 P_1 具有"否"回答的任何实例都变成 P_2 具有"否"回答的实例



归约

Definition (归约)

如果有一个算法把问题 P_1 的全部实例转化成问题 P_2 的具有相同答案的实例, 则称 P_1 归约到 P_2.

归约表明 P_2 至少和 P_1 一样难. 下面给出归约的形式化定义.

归约

Definition (归约)

如果有一个算法把问题 P_1 的全部实例转化成问题 P_2 的具有相同答案的实例, 则称 P_1 归约到 P_2.

归约表明 P_2 至少和 P_1 一样难. 下面给出归约的形式化定义.

Definition (映射可归约)

语言 A 是映射可归约到语言 B 的, 如果存在可计算函数 $f : \Sigma^* \to \Sigma^*$ 使得对每个 w,

$$w \in A \Leftrightarrow f(w) \in B$$

记作 $A \leqslant_m B$. 称函数 f 为从 A 到 B 的归约.

归约

Theorem (归约)

如果存在从 P_1 到 P_2 的归约，那么：

1. 若 P_1 是不可判定的，则 P_2 也是不判定的.
2. 若 P_1 是非递归可枚举的，则 P_2 也是非递归可枚举的.

归约

Theorem (归约)

如果存在从 P_1 到 P_2 的归约，那么：

1. 若 P_1 是不可判定的，则 P_2 也是不判定的.
2. 若 P_1 是非递归可枚举的，则 P_2 也是非递归可枚举的.

证明：

1. 若可以判定 P_2，则可以把从 P_1 到 P_2 的归约与判定 P_2 的算法组合起来，构造一个判定 P_1 的算法. 具体地说，给定 P_1 的一个实例 x，对 x 使用归约算法，这个算法将 x 转化成 P_2 的一个实例 y. 对 y 使用判定 P_2 的算法.
   - 若算法结果为“是”，因为 P_1 可归约到 P_2，所以 P_1 对于 w 回答为是.
   - 若算法结果为“否”，因为 P_1 可归约到 P_2，所以 P_1 对于 w 回答为否.

归约

Theorem (归约)

如果存在从 P_1 到 P_2 的归约，那么：

1. 若 P_1 是不可判定的，则 P_2 也是不判定的.
2. 若 P_1 是非递归可枚举的，则 P_2 也是非递归可枚举的.

归约

Theorem (归约)

如果存在从 P_1 到 P_2 的归约，那么：

1. 若 P_1 是不可判定的，则 P_2 也是不判定的.
2. 若 P_1 是非递归可枚举的，则 P_2 也是非递归可枚举的.

证明：

2. 假设 P_1 是非递归可枚举的，而 P_2 是递归可枚举的，则存在一台图灵机识别 P_2. 从 P_1 的一个实例 x 开始，将其归约为 P_2 的一个实例 y.
   - 如果 x 属于 P_1，则 y 属于 P_2，所以图灵机接受 x，
   - 如果 x 不属于 P_1，则 y 不属于 P_2，所以图灵机将不接受 x.

   这就构造出一台识别 P_1 的图灵机，这与 P_1 是非递归可枚举矛盾.

莱斯定理

考虑以下问题：

Example

- 图灵机 M 接受的语言是空的吗？

莱斯定理

考虑以下问题：

Example

- 图灵机 M 接受的语言是空的吗？$L_e = \{\langle M \rangle | L(M) = \emptyset\}$

莱斯定理

考虑以下问题：

Example

- 图灵机 M 接受的语言是空的吗？$L_e = \{\langle M \rangle | L(M) = \emptyset\}$
- 图灵机 M 接受的语言是非空的吗？

莱斯定理

考虑以下问题：

Example

- 图灵机 M 接受的语言是空的吗？$L_e = \{\langle M \rangle | L(M) = \emptyset\}$
- 图灵机 M 接受的语言是非空的吗？$L_{ne} = \{\langle M \rangle | L(M) \neq \emptyset\}$

莱斯定理

考虑以下问题：

Example

- 图灵机 M 接受的语言是空的吗？$L_e = \{\langle M\rangle | L(M) = \emptyset\}$
- 图灵机 M 接受的语言是非空的吗？$L_{ne} = \{\langle M\rangle | L(M) \neq \emptyset\}$

Definition (递归可枚举语言的性质)

递归可枚举语言的任一集合 P 就称为递归可枚举语言的一个性质. 如果 P 非空, 也不是所有的递归可枚举语言, 则 P 称为递归可枚举语言的非平凡性质.

莱斯定理

考虑以下问题：

Example

- 图灵机 M 接受的语言是空的吗？$L_e = \{\langle M\rangle | L(M) = \emptyset\}$
- 图灵机 M 接受的语言是非空的吗？$L_{ne} = \{\langle M\rangle | L(M) \neq \emptyset\}$

Definition (递归可枚举语言的性质)

递归可枚举语言的任一集合 P 就称为递归可枚举语言的一个性质. 如果 P 非空, 也不是所有的递归可枚举语言, 则 P 称为递归可枚举语言的非平凡性质.

P 对应的语言：

莱斯定理

考虑以下问题：

Example

- 图灵机 M 接受的语言是空的吗？$L_e = \{\langle M\rangle | L(M) = \emptyset\}$
- 图灵机 M 接受的语言是非空的吗？$L_{ne} = \{\langle M\rangle | L(M) \neq \emptyset\}$

Definition (递归可枚举语言的性质)

递归可枚举语言的任一集合 P 就称为递归可枚举语言的一个性质. 如果 P 非空, 也不是所有的递归可枚举语言, 则 P 称为递归可枚举语言的非平凡性质.

P 对应的语言：$L_p = \{\langle M\rangle | L(M) \in P\}$

莱斯定理

Theorem

递归可枚举语言的每个非平凡性质都是不可判定的

莱斯定理

Theorem

递归可枚举语言的每个非平凡性质都是不可判定的

莱斯定理的应用:

莱斯定理

Theorem

递归可枚举语言的每个非平凡性质都是不可判定的

莱斯定理的应用：所有只涉及图灵机接受的语言的与图灵机有关的问题都是不可判定的.

Example

- 一台图灵机接受的语言是否为空
- 一台图灵机接受的语言是否有穷
- 一台图灵机接受的语言是否是正则语言
- 一台图灵机接受的语言是否是上下文无关语言

莱斯定理

Theorem

递归可枚举语言的每个非平凡性质都是不可判定的

莱斯定理的应用：所有只涉及图灵机接受的语言的与图灵机有关的问题都是不可判定的.

Example

- 一台图灵机接受的语言是否为空
- 一台图灵机接受的语言是否有穷
- 一台图灵机接受的语言是否是正则语言
- 一台图灵机接受的语言是否是上下文无关语言

莱斯定理并没有蕴含与一台图灵机有关的每个问题都是不可判定的.

总结

- 可判定语言
  - 判定问题
  - 图灵机的编码
  - 与正则语言相关的可判定性问题
  - 与上下文无关语言相关的可判定性问题

总结

- 可判定语言
  - 判定问题
  - 图灵机的编码
  - 与正则语言相关的可判定性问题
  - 与上下文无关语言相关的可判定性问题
- 不可判定语言
  - 非递归可枚举语言
  - 通用图灵机
  - 不可判定语言
- 归约
  - 归约
  - 莱斯定理