

## 第七章：可计算性，不可解性，不可判定性

沈榆平

yuping.shen.ilc@gmail.com

中山大学逻辑与认知研究所

2013年12月

在1900及1928年世界数学家大会上, Hilbert分别提出了一些著名的问题,比如:

- 有一个机械过程可以判定任意多项式丢番图方程有整数解吗? (1900)

## 算法

在1900及1928年世界数学家大会上，Hilbert分别提出了一些著名的问题，比如：

- 有一个机械过程可以判定任意多项式丢番图方程有整数解吗？(1900)
- 数学是完全的吗？即，是否任何数学命题都能被证明或者证否？(1928)

- 有一个机械过程可以判定任意多项式丢番图方程有整数解吗？(1900)
- 数学是完全的吗？即，是否任何数学命题都能被证明或者证否？(1928)
- 数学是一致的吗？即，是否从正确的证明过程不会得到错误的结论？(1928)

- 有一个机械过程可以判定任意多项式丢番图方程有整数解吗？(1900)
- 数学是完全的吗？即，是否任何数学命题都能被证明或者证否？(1928)
- 数学是一致的吗？即，是否从正确的证明过程不会得到错误的结论？(1928)
- 数学是可判定的吗？即，对任一个数学命题，是否有一机械过程可以处理并给出正确判断？(1928)

- 有一个机械过程可以判定任意多项式丢番图方程有整数解吗？(1900)
- 数学是完全的吗？即，是否任何数学命题都能被证明或者证否？(1928)
- 数学是一致的吗？即，是否从正确的证明过程不会得到错误的结论？(1928)
- 数学是可判定的吗？即，对任一个数学命题，是否有一机械过程可以处理并给出正确判断？(1928)

## 讨论

除了第三个问题答案为“是”，其它答案都为“否”。

- 有一个机械过程可以判定任意多项式丢番图方程有整数解吗？(1900)
- 数学是完全的吗？即，是否任何数学命题都能被证明或者证否？(1928)
- 数学是一致的吗？即，是否从正确的证明过程不会得到错误的结论？(1928)
- 数学是可判定的吗？即，对任一个数学命题，是否有一机械过程可以处理并给出正确判断？(1928)

除了第三个问题答案为"是", 其它答案都为“否”。回答这些问题与算法(又称"能行过程, effective procedure")的概念紧密相关。



- 有一个机械过程可以判定任意多项式丢番图方程有整数解吗？(1900)
- 数学是完全的吗？即，是否任何数学命题都能被证明或者证否？(1928)
- 数学是一致的吗？即，是否从正确的证明过程不会得到错误的结论？(1928)
- 数学是可判定的吗？即，对任一个数学命题，是否有一机械过程可以处理并给出正确判断？(1928)

除了第三个问题答案为"是", 其它答案都为“否”。回答这些问题与算法(又称"能行过程, effective procedure")的概念紧密相关。那么, 算法(能行过程), 是如何定义的?

一个算法是一个计算过程的明确可行的指令集合，可用于解答给定问题类中的任一实例。

一个算法是一个计算过程的明确可行的指令集合，可用于解答给定问题类中的任一实例。

和集合的定义类似，算法的定义不是一个严格数学定义，而更接近一个哲学定义。

一个算法是一个计算过程的明确可行的指令集合，可用于解答给定问题类中的任一实例。

和集合的定义类似，算法的定义不是一个严格数学定义，而更接近一个哲学定义。我们可以注意到它的概念很难再进一步细分下去。

一个算法是一个计算过程的明确可行的指令集合，可用于解答给定问题类中的任一实例。

和集合的定义类似，算法的定义不是一个严格数学定义，而更接近一个哲学定义。我们可以注意到它的概念很难再进一步细分下去。比如，什么是过程？什么是明确可行？等等。

一个算法是一个计算过程的明确可行的指令集合，可用于解答给定问题类中的任一实例。

和集合的定义类似，算法的定义不是一个严格数学定义，而更接近一个哲学定义。我们可以注意到它的概念很难再进一步细分下去。比如，什么是过程？什么是明确可行？等等。

- $\{f(n)=2n \text{ 的值是什么? } | n \in \mathbb{N}\}$

一个算法是一个计算过程的明确可行的指令集合，可用于解答给定问题类中的任一实例。

和集合的定义类似，算法的定义不是一个严格数学定义，而更接近一个哲学定义。我们可以注意到它的概念很难再进一步细分下去。比如，什么是过程？什么是明确可行？等等。

- $\{f(n)=2n \text{ 的值是什么? } | n \in \mathbb{N}\}$
- $\{2 \text{ 是 } n \text{ 的一个因子吗? } | n \in \mathbb{N}\}$

一个算法是一个计算过程的明确可行的指令集合，可用于解答给定问题类中的任一实例。

和集合的定义类似，算法的定义不是一个严格数学定义，而更接近一个哲学定义。我们可以注意到它的概念很难再进一步细分下去。比如，什么是过程？什么是明确可行？等等。

- $\{f(n)=2n \text{ 的值是什么? } | n \in \mathbb{N}\}$
- $\{2 \text{ 是 } n \text{ 的一个因子吗? } | n \in \mathbb{N}\}$
- $\{n \text{ 是一个素数吗? } | n \in \mathbb{N}\}$



一个算法是一个计算过程的明确可行的指令集合，可用于解答给定问题类中的任一实例。

和集合的定义类似，算法的定义不是一个严格数学定义，而更接近一个哲学定义。我们可以注意到它的概念很难再进一步细分下去。比如，什么是过程？什么是明确可行？等等。

- $\{f(n)=2n \text{ 的值是什么? } | n \in \mathbb{N}\}$
- $\{2 \text{ 是 } n \text{ 的一个因子吗? } | n \in \mathbb{N}\}$
- $\{n \text{ 是一个素数吗? } | n \in \mathbb{N}\}$

## 问题的算法

- 1  $\{f(n)=2n \text{ 的值是什么? } | n \in \mathbb{N}\}$

## 问题的算法

- 1  $\{f(n)=2n \text{ 的值是什么? } | n \in \mathbb{N}\}$   
对给出的 $n$ 用乘法算出 $2n$ 即可。

### 问题的算法

- 1  $\{f(n)=2n \text{ 的值是什么? } | n \in \mathbb{N}\}$   
对给出的 $n$ 用乘法算出 $2n$ 即可。
- 2  $\{2 \text{ 是 } n \text{ 的一个因子吗? } | n \in \mathbb{N}\}$

## 问题的算法

- 1  $\{f(n)=2n \text{ 的值是什么? } | n \in \mathbb{N}\}$   
对给出的 $n$ 用乘法算出 $2n$ 即可。
- 2  $\{2 \text{ 是 } n \text{ 的一个因子吗? } | n \in \mathbb{N}\}$   
将 $n$ 连续除于2，如果最终余数是0，那么回答是，否则回答否。

### 问题的算法

- ①  $\{f(n)=2n \text{ 的值是什么? } | n \in \mathbb{N}\}$   
对给出的 $n$ 用乘法算出 $2n$ 即可。
- ②  $\{2 \text{ 是 } n \text{ 的一个因子吗? } | n \in \mathbb{N}\}$   
将 $n$ 连续除于2，如果最终余数是0，那么回答是，否则回答否。
- ③  $\{n \text{ 是一个素数吗? } | n \in \mathbb{N}\}$

### 问题的算法

- 1  $\{f(n)=2n \text{ 的值是什么? } | n \in \mathbb{N}\}$   
对给出的 $n$ 用乘法算出 $2n$ 即可。
- 2  $\{2 \text{ 是 } n \text{ 的一个因子吗? } | n \in \mathbb{N}\}$   
将 $n$ 连续除于2, 如果最终余数是0, 那么回答是, 否则回答否。
- 3  $\{n \text{ 是一个素数吗? } | n \in \mathbb{N}\}$   
辗转相除法....

### 问题的算法

- ①  $\{f(n)=2n \text{ 的值是什么? } | n \in \mathbb{N}\}$   
对给出的 $n$ 用乘法算出 $2n$ 即可。
- ②  $\{2 \text{ 是 } n \text{ 的一个因子吗? } | n \in \mathbb{N}\}$   
将 $n$ 连续除于2，如果最终余数是0，那么回答是，否则回答否。
- ③  $\{n \text{ 是一个素数吗? } | n \in \mathbb{N}\}$   
辗转相除法....

为了方便讨论，我们假设所有问题都有问题1的形式。



## 如何描述算法？

- 用一种理论上精确定义的抽象的计算机器，如图灵机

## 如何描述算法？

- 用一种理论上精确定义的抽象的计算机器，如图灵机
- 描述计算过程的形式系统,如 $\lambda$ 演算

## 如何描述算法？

- 用一种理论上精确定义的抽象的计算机器，如图灵机
- 描述计算过程的形式系统，如 $\lambda$ 演算
- 给出函数类的形式构造，如递归函数

### 如何描述算法？

- 用一种理论上精确定义的抽象的计算机器，如图灵机
- 描述计算过程的形式系统,如 $\lambda$ 演算
- 给出函数类的形式构造，如递归函数
- 前两者是等价的。

# 算法

## 如何描述算法？

- 用一种理论上精确定义的抽象的计算机器，如图灵机
- 描述计算过程的形式系统,如 $\lambda$ 演算
- 给出函数类的形式构造，如递归函数
- 前两者是等价的。

## Church论题(Church's Thesis)

所有可以由算法计算的函数，就是递归函数；所有递归函数，也存在计算的算法。

# 算法

## 如何描述算法？

- 用一种理论上精确定义的抽象的计算机器，如图灵机
- 描述计算过程的形式系统,如 $\lambda$ 演算
- 给出函数类的形式构造，如递归函数
- 前两者是等价的。

## Church论题(Church's Thesis)

所有可以由算法计算的函数，就是递归函数；所有递归函数，也存在计算的算法。

Church's Thesis不是定理，只是一个哲学论断。但绝大多数数学家都接受这一论断。

# 算法

## 如何描述算法？

- 用一种理论上精确定义的抽象的计算机器，如图灵机
- 描述计算过程的形式系统，如 $\lambda$ 演算
- 给出函数类的形式构造，如递归函数
- 前两者是等价的。

## Church论题(Church's Thesis)

所有可以由算法计算的函数，就是递归函数；所有递归函数，也存在计算的算法。

Church's Thesis不是定理，只是一个哲学论断。但绝大多数数学家都接受这一论断。我们接下来介绍一种描述算法的机器——图灵机。

# 图灵机

数学家、逻辑学家、计算机科学家阿兰图灵



**Figure:** Alan Mathison Turing (1912-1954)



## 图灵机

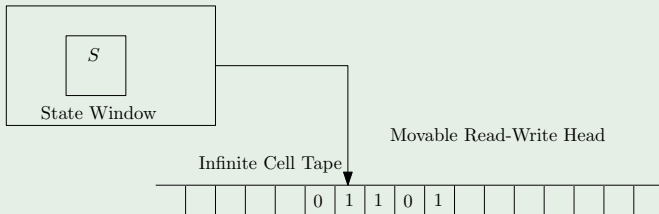
数学家、逻辑学家、计算机科学家阿兰图灵



**Figure:** Alan Mathison Turing (1912-1954)

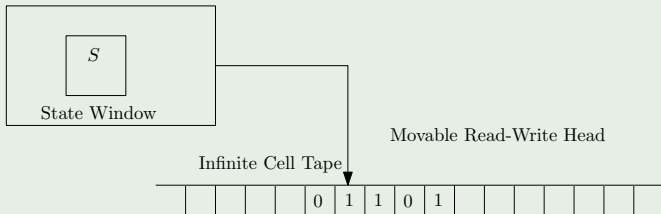
## 图灵机

## 图灵机构造

**Figure:** A Turing Machine

# 图灵机

## 图灵机构造



**Figure:** A Turing Machine

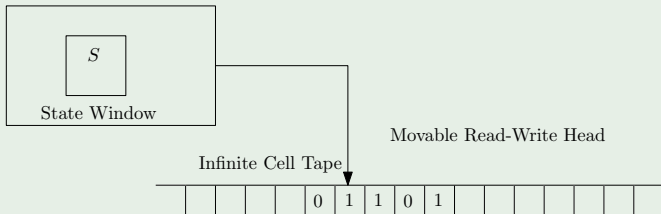
## 图灵机读写头操作

- 读/写一个单元格的字符;
- 向左/右移动一个单元格;

但如何决定读写头的操作呢?

图灵机

## 图灵机构造



### Figure: A Turing Machine

## 图灵机读写头操作

- 读/写一个单元格的字符;
- 向左/右移动一个单元格;

但如何决定读写头的操作呢？由图灵机的指令给出。

## 图灵机

一条指令告诉图灵机根据：

## 图灵机

一条指令告诉图灵机根据:

- 当前状态窗口中的字符(即当前状态)

## 图灵机

一条指令告诉图灵机根据：

- 当前状态窗口中的字符(即当前状态)
- 与当前读取的字符  
来决定：

## 图灵机

一条指令告诉图灵机根据:

- 当前状态窗口中的字符(即当前状态)
- 与当前读取的字符  
来决定:
- 下一状态窗口中的字符(即下一状态)



## 图灵机

一条指令告诉图灵机根据：

- 当前状态窗口中的字符(即当前状态)
- 与当前读取的字符  
来决定：
- 下一状态窗口中的字符(即下一状态)
- 读写头操作(左移/右移/写入字符)

## 图灵机

一条指令告诉图灵机根据:

- 当前状态窗口中的字符(即当前状态)
- 与当前读取的字符  
来决定:
- 下一状态窗口中的字符(即下一状态)
- 读写头操作(左移/右移/写入字符)

### 图灵机的指令

图灵机的一条指令是一个四元组 $(q, w, A, q')$ ,其中:

## 图灵机

一条指令告诉图灵机根据：

- 当前状态窗口中的字符(即当前状态)
- 与当前读取的字符  
来决定：
- 下一状态窗口中的字符(即下一状态)
- 读写头操作(左移/右移/写入字符)

### 图灵机的指令

图灵机的一条指令是一个四元组 $(q, w, A, q')$ ,其中：

- $q$ 是当前状态

## 图灵机

一条指令告诉图灵机根据：

- 当前状态窗口中的字符(即当前状态)
- 与当前读取的字符  
来决定：
- 下一状态窗口中的字符(即下一状态)
- 读写头操作(左移/右移/写入字符)

### 图灵机的指令

图灵机的一条指令是一个四元组 $(q, w, A, q')$ ,其中：

- $q$ 是当前状态
- $w$ 是当前字符

## 图灵机

一条指令告诉图灵机根据:

- 当前状态窗口中的字符(即当前状态)
- 与当前读取的字符  
来决定:
- 下一状态窗口中的字符(即下一状态)
- 读写头操作(左移/右移/写入字符)

### 图灵机的指令

图灵机的一条指令是一个四元组 $(q, w, A, q')$ ,其中:

- $q$ 是当前状态
- $w$ 是当前字符
- $A \in \{L, R, w'\}$ 是一个操作, 其中L/R表示读写头向左/右移,  $w'$ 是一个被写入的字符

## 图灵机

一条指令告诉图灵机根据:

- 当前状态窗口中的字符(即当前状态)
- 与当前读取的字符  
来决定:
- 下一状态窗口中的字符(即下一状态)
- 读写头操作(左移/右移/写入字符)

### 图灵机的指令

图灵机的一条指令是一个四元组 $(q, w, A, q')$ ,其中:

- $q$ 是当前状态
- $w$ 是当前字符
- $A \in \{L, R, w'\}$ 是一个操作, 其中L/R表示读写头向左/右移,  $w'$ 是一个被写入的字符
- $q'$ 是一个操作后的状态

## 图灵机

一条指令告诉图灵机根据:

- 当前状态窗口中的字符(即当前状态)
- 与当前读取的字符  
来决定:
- 下一状态窗口中的字符(即下一状态)
- 读写头操作(左移/右移/写入字符)

### 图灵机的指令

图灵机的一条指令是一个四元组 $(q, w, A, q')$ ,其中:

- $q$ 是当前状态
- $w$ 是当前字符
- $A \in \{L, R, w'\}$ 是一个操作, 其中L/R表示读写头向左/右移,  $w'$ 是一个被写入的字符
- $q'$ 是一个操作后的状态

## 图灵机

## 图灵机TM

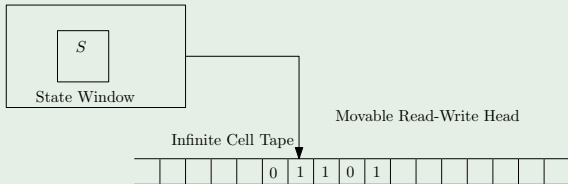




Diagram illustrating a Turing Machine configuration:

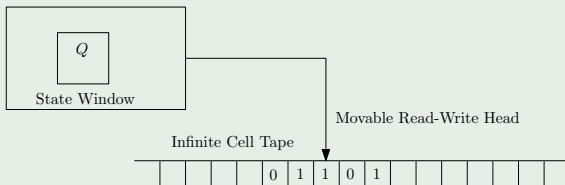
- State Window:** A box containing the state  $S$ .
- Infinite Cell Tape:** A horizontal sequence of cells. The first five cells contain the symbols 0, 1, 1, 0, and 1, followed by empty cells.
- Movable Read-Write Head:** An arrow points from the State Window to the head, which is positioned over the second cell (containing 1) of the tape.

Diagram illustrating a Turing Machine configuration:

- A **State Window** contains the state  $Q$ .
- An arrow points from the State Window to the **Movable Read-Write Head**.
- The **Movable Read-Write Head** is positioned over the third cell of the **Infinite Cell Tape**.
- The **Infinite Cell Tape** contains the sequence: 0 1 1 0 1 (with empty cells before and after).

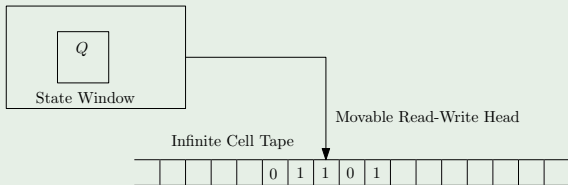
## 图灵机

## 图灵机TM

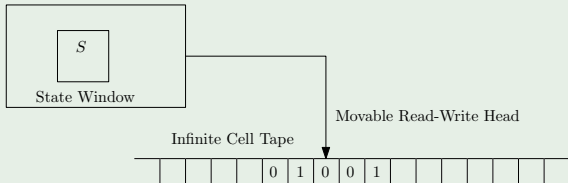


## 图灵机

## 图灵机TM



## 图灵机TM执行(Q,1,0,S)



# 图灵机

## 图灵机定义

一个图灵机是一个三元组  $\langle \Sigma, \Omega, \Delta \rangle$ , 其中:

# 图灵机

## 图灵机定义

一个图灵机是一个三元组  $\langle \Sigma, \Omega, \Delta \rangle$ , 其中:

- $\Sigma$  是一个有穷字符集合;

# 图灵机

## 图灵机定义

一个图灵机是一个三元组 $\langle \Sigma, \Omega, \Delta \rangle$ ,其中:

- $\Sigma$ 是一个有穷字符集合;
- $\Omega$ 是一个有穷状态集合;

# 图灵机

## 图灵机定义

一个图灵机是一个三元组  $\langle \Sigma, \Omega, \Delta \rangle$ , 其中:

- $\Sigma$  是一个有穷字符集合;
- $\Omega$  是一个有穷状态集合;
- $\Delta$  是一个有穷指令集合.

# 图灵机

## 图灵机定义

一个图灵机是一个三元组  $\langle \Sigma, \Omega, \Delta \rangle$ , 其中:

- $\Sigma$  是一个有穷字符集合;
- $\Omega$  是一个有穷状态集合;
- $\Delta$  是一个有穷指令集合.

## 停机状态

特别的, 我们可以指定某个状态, 如  $S_{end}$  作为图灵机停机状态, 也就是规定图灵机遇到此状态时就停止。



# 图灵机

## 图灵机定义

一个图灵机是一个三元组  $\langle \Sigma, \Omega, \Delta \rangle$ , 其中:

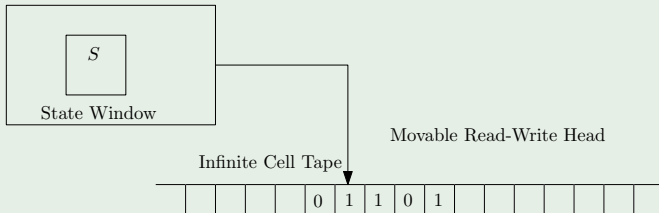
- $\Sigma$  是一个有穷字符集合;
- $\Omega$  是一个有穷状态集合;
- $\Delta$  是一个有穷指令集合.

## 停机状态

特别的, 我们可以指定某个状态, 如  $S_{end}$  作为图灵机停机状态, 也就是规定图灵机遇到此状态时就停止。此外, 图灵机在没有指令可执行时, 也停止。

## 图灵机

## 图灵机TM

**Figure:** A Turing Machine

## 图灵机

## 图灵机TM

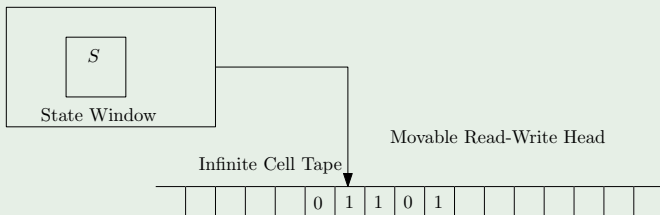


Figure: A Turing Machine

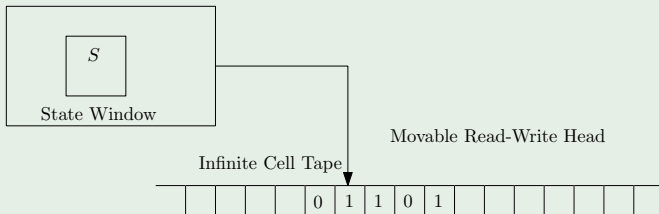
## TM的形式定义

这前讨论的图灵机可以表示

为  $\langle \{0, 1\}, \{Q, S\}, \{(Q, 1, 0, S), (S, 1, R, Q)\} \rangle$ 。

## 图灵机

## 图灵机TM

**Figure:** A Turing Machine

## TM的形式定义

这前讨论的图灵机可以表示

为  $\langle \{0, 1\}, \{Q, S\}, \{(Q, 1, 0, S), (S, 1, R, Q)\} \rangle$ 。注意，纸带的输入并没有被包含在图灵机的定义中。

## 图灵机

## 图灵机TM

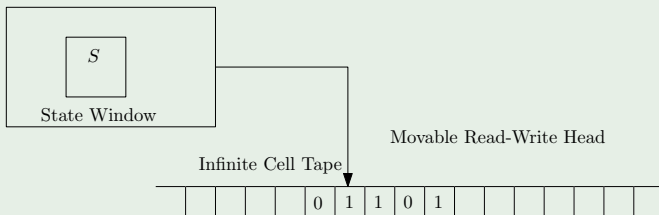


Figure: A Turing Machine

## TM的形式定义

这前讨论的图灵机可以表示

为  $\langle \{0, 1\}, \{Q, S\}, \{(Q, 1, 0, S), (S, 1, R, Q)\} \rangle$ 。注意，纸带的输入并没有被包含在图灵机的定义中。纸带事实上是一个计算的输入。

# 图灵机

## 图灵机的歌德尔编码

既然图灵机也可以用符号定义给出，那么我们可以很自然地想到可以利用歌德尔编码的方法，对图灵机进行编码。

## 图灵机

### 图灵机的歌德尔编码

既然图灵机也可以用符号定义给出，那么我们可以很自然地想到可以利用歌德尔编码的方法，对图灵机进行编码。事实上，每个图灵机都可以编码成一个自然数，而一个自然数(如果存在对应)也可以被解码成一个图灵机。

## 图灵机

### 图灵机的歌德尔编码

既然图灵机也可以用符号定义给出，那么我们可以很自然地想到可以利用歌德尔编码的方法，对图灵机进行编码。事实上，每个图灵机都可以编码成一个自然数，而一个自然数(如果存在对应)也可以被解码成一个图灵机。从这个角度说，**程序(图灵机)就是数，数就是程序(图灵机)**。



# 图灵机

## 图灵机的歌德尔编码

既然图灵机也可以用符号定义给出，那么我们可以很自然地想到可以利用歌德尔编码的方法，对图灵机进行编码。事实上，每个图灵机都可以编码成一个自然数，而一个自然数(如果存在对应)也可以被解码成一个图灵机。从这个角度说，**程序(图灵机)就是数，数就是程序(图灵机)**。

## 命题7.20

所有图灵机(或其编码)是能行可枚举的。即，我们有方法把所有图灵机(理论上)排成一行： $T_0, T_1, T_2, \dots$ 。

# 图灵机

## 讨论

- 所有图灵机都存在一个等价的只使用两个字符 $\{0, 1\}$ 的图灵机(0可以看成是空字符B)。

# 图灵机

## 讨论

- 所有图灵机都存在一个等价的只使用两个字符 $\{0, 1\}$ 的图灵机(0可以看成是空字符B)。
- 所有图灵机都存在一个等价的只使用两个状态 $\{Q, S\}$ 的图灵机。

# 图灵机

## 讨论

- 所有图灵机都存在一个等价的只使用两个字符 $\{0, 1\}$ 的图灵机(0可以看成是空字符B)。
- 所有图灵机都存在一个等价的只使用两个状态 $\{Q, S\}$ 的图灵机。
- 但并非所有图灵机都能转换成只有两个字符及两个状态的图灵机。

# 图灵机

## 讨论

- 所有图灵机都存在一个等价的只使用两个字符 $\{0, 1\}$ 的图灵机(0可以看成是空字符B)。
- 所有图灵机都存在一个等价的只使用两个状态 $\{Q, S\}$ 的图灵机。
- 但并非所有图灵机都能转换成只有两个字符及两个状态的图灵机。

前二者的原因在于任意有穷的字符集都可以被二进制编码表示，但只有两个状态和两个字符的图灵机个数是有穷的，所以不足以表示所有图灵机(无穷可数多个)。

## 图灵机

### 图灵论题

所有算法可以计算的函数，都可以由图灵机计算,反之亦然。

# 图灵机

## 图灵论题

所有算法可以计算的函数，都可以由图灵机计算,反之亦然。

## 图灵可计算函数

此处图灵可计算函数的含义是图灵机在计算这个函数时最终将停机。

# 图灵机

## 图灵论题

所有算法可以计算的函数，都可以由图灵机计算,反之亦然。

## 图灵可计算函数

此处图灵可计算函数的含义是图灵机在计算这个函数时最终将停机。以下几个概念是等价的：

- 图灵可计算的



# 图灵机

## 图灵论题

所有算法可以计算的函数，都可以由图灵机计算,反之亦然。

## 图灵可计算函数

此处图灵可计算函数的含义是图灵机在计算这个函数时最终将停机。以下几个概念是等价的：

- 图灵可计算的
- 可计算的

# 图灵机

## 图灵论题

所有算法可以计算的函数，都可以由图灵机计算,反之亦然。

## 图灵可计算函数

此处图灵可计算函数的含义是图灵机在计算这个函数时最终将停机。以下几个概念是等价的：

- 图灵可计算的
- 可计算的
- 递归的

# 图灵机

## 图灵论题

所有算法可以计算的函数，都可以由图灵机计算,反之亦然。

## 图灵可计算函数

此处图灵可计算函数的含义是图灵机在计算这个函数时最终将停机。以下几个概念是等价的：

- 图灵可计算的
- 可计算的
- 递归的
- 可判定的

# 图灵机

## 图灵论题

所有算法可以计算的函数，都可以由图灵机计算,反之亦然。

## 图灵可计算函数

此处图灵可计算函数的含义是图灵机在计算这个函数时最终将停机。以下几个概念是等价的：

- 图灵可计算的
- 可计算的
- 递归的
- 可判定的
- 可解的

# 图灵机

## 图灵论题

所有算法可以计算的函数，都可以由图灵机计算,反之亦然。

## 图灵可计算函数

此处图灵可计算函数的含义是图灵机在计算这个函数时最终将停机。以下几个概念是等价的：

- 图灵可计算的
- 可计算的
- 递归的
- 可判定的
- 可解的

一个判定问题即答案为是或者否的问题。

# 图灵机

## L的判定问题

对任意 $L$ 的公式 $A$ ，问 $A$ 是否为 $L$ 的定理？。

## L的可判定性

这个问题是可判定的，

# 图灵机

## L的判定问题

对任意 $L$ 的公式 $A$ ，问 $A$ 是否为 $L$ 的定理？。

## L的可判定性

这个问题是可判定的，因为可以构造一个验证真值表的图灵机，

# 图灵机

## L的判定问题

对任意 $L$ 的公式 $A$ ，问 $A$ 是否为 $L$ 的定理？。

## L的可判定性

这个问题是可判定的，因为可以构造一个验证真值表的图灵机，如果公式 $A$ 是个重言式，它可以打印出1，反之打印0。



# 图灵机

## L的判定问题

对任意 $L$ 的公式 $A$ ，问 $A$ 是否为 $L$ 的定理？。

## L的可判定性

这个问题是可判定的，因为可以构造一个验证真值表的图灵机，如果公式 $\varphi$ 是个重言式，它可以打印出1，反之打印0。

## 不可判定性

虽然图灵机能够计算很多问题，

# 图灵机

## L的判定问题

对任意 $L$ 的公式 $A$ , 问 $A$ 是否为 $L$ 的定理？。

## L的可判定性

这个问题是可判定的, 因为可以构造一个验证真值表的图灵机, 如果公式 $\varphi$ 是个重言式, 它可以打印出1, 反之打印0.

## 不可判定性

虽然图灵机能够计算很多问题, 但也存在图灵机不能解决的问题(即根据实际情况回答是或者否), 即不可判定的问题。

## 图灵机

## 停机问题(Halting Problem)

任给一个图灵机  $T_m$  和一个输入  $n$ , 它可以停机吗?

## 图灵机

## 停机问题(Halting Problem)

任给一个图灵机  $T_m$  和一个输入  $n$ , 它可以停机吗? (注意, 停机意味着运算正常终止)

# 图灵机

## 停机问题(Halting Problem)

任给一个图灵机  $T_m$  和一个输入  $n$ , 它可以停机吗? (注意, 停机意味着运算正常终止)

## 停机问题不可判定

这个问题是不可判定的。

# 图灵机

## 停机问题(Halting Problem)

任给一个图灵机  $T_m$  和一个输入  $n$ ，它可以停机吗？(注意，停机意味着运算正常终止)

## 停机问题不可判定

这个问题是不可判定的。也就是说，任意给一个图灵机及输入，

# 图灵机

## 停机问题(Halting Problem)

任给一个图灵机  $T_m$  和一个输入  $n$ ，它可以停机吗？(注意，停机意味着运算正常终止)

## 停机问题不可判定

这个问题是不可判定的。也就是说，任意给一个图灵机及输入，我们没有一个通用的办法确定它是否可以正常结束(停机)。

## 图灵机

### 停机问题(Halting Problem)

任给一个图灵机  $T_m$  和一个输入  $n$ , 它可以停机吗? (注意, 停机意味着运算正常终止)

### 停机问题不可判定

这个问题是不可判定的。也就是说, 任意给一个图灵机及输入, 我们没有一个通用的办法确定它是否可以正常结束(停机)。

### K的判定问题

任给一个  $\mathcal{L}$  的公式  $\mathcal{A}$ , 问  $\mathcal{A}$  是否为  $K$  的定理?



## 图灵机

### 停机问题(Halting Problem)

任给一个图灵机  $T_m$  和一个输入  $n$ , 它可以停机吗? (注意, 停机意味着运算正常终止)

### 停机问题不可判定

这个问题是不可判定的。也就是说, 任意给一个图灵机及输入, 我们没有一个通用的办法确定它是否可以正常结束(停机)。

### K的判定问题

任给一个  $\mathcal{L}$  的公式  $\mathcal{A}$ , 问  $\mathcal{A}$  是否为  $K$  的定理?

### K的不可判定性

这个问题是不可判定的, 因为可以证明  $K$  可判定当且仅当停机问题可判定。

# 图灵机

## 不可判定性讨论

事实上，存在一些图灵机对可停机的图灵机及K的定理作出正确回答（是），

# 图灵机

## 不可判定性讨论

事实上，存在一些图灵机对可停机的图灵机及K的定理作出正确回答（是），但对不可停机的图灵机或者不是K的定理的公式则无法回答。

# 图灵机

## 不可判定性讨论

事实上，存在一些图灵机对可停机的图灵机及K的定理作出正确回答（是），但对不可停机的图灵机或者不是K的定理的公式则无法回答。因此，停机问题与系统K定理证明问题又称半可判定的，或者部分可判定的。

## 图灵机

### 不可判定性讨论

事实上，存在一些图灵机对可停机的图灵机及K的定理作出正确回答（是），但对不可停机的图灵机或者不是K的定理的公式则无法回答。因此，停机问题与系统K定理证明问题又称半可判定的，或者部分可判定的。

### 另一种理论计算机

我们已经知道图灵机刻画所有可计算的函数，

## 图灵机

### 不可判定性讨论

事实上，存在一些图灵机对可停机的图灵机及K的定理作出正确回答（是），但对不可停机的图灵机或者不是K的定理的公式则无法回答。因此，停机问题与系统K定理证明问题又称半可判定的，或者部分可判定的。

### 另一种理论计算机

我们已经知道图灵机刻画所有可计算的函数，但图灵机有时不够灵活，表示不够直观，

## 图灵机

### 不可判定性讨论

事实上，存在一些图灵机对可停机的图灵机及K的定理作出正确回答（是），但对不可停机的图灵机或者不是K的定理的公式则无法回答。因此，停机问题与系统K定理证明问题又称半可判定的，或者部分可判定的。

### 另一种理论计算机

我们已经知道图灵机刻画所有可计算的函数，但图灵机有时不够灵活，表示不够直观，我们介绍一种等价于图灵机的抽象机器，称为Unlimited Register Machine(无限制寄存机)。





# URM机器

## URM寄存器

3	1	5	0	1											
$r_0$	$r_1$	$r_2$	$r_3$	$r_4$	$r_5$	$r_6$									

URM直观上是一条一端无限延升的纸带，

# URM机器

## URM寄存器

3	1	5	0	1											
$r_0$	$r_1$	$r_2$	$r_3$	$r_4$	$r_5$	$r_6$									

URM直观上是一条一端无限延升的纸带，上面每个单元格可用于存储自然数，

# URM机器

## URM寄存器

3	1	5	0	1											
$r_0$	$r_1$	$r_2$	$r_3$	$r_4$	$r_5$	$r_6$									

URM直观上是一条一端无限延升的纸带，上面每个单元格可用于存储自然数，被称为寄存器。

## URM机器

## URM寄存器

3	1	5	0	1											
$r_0$	$r_1$	$r_2$	$r_3$	$r_4$	$r_5$	$r_6$									

URM直观上是一条一端无限延升的纸带，上面每个单元格可用于存储自然数，被称为寄存器。第一个寄存器(单元格)被标为 $r_0$ ，第二个被标为 $r_1$ ，等等。

## URM机器

## URM寄存器

3	1	5	0	1											
$r_0$	$r_1$	$r_2$	$r_3$	$r_4$	$r_5$	$r_6$									

URM直观上是一条一端无限延升的纸带，上面每个单元格可用于存储自然数，被称为寄存器。第一个寄存器(单元格)被标为 $r_0$ ，第二个被标为 $r_1$ ，等等。

# URM机器

## URM程序

URM程序是一个以下指令组成的序列:

---

$Z(n)$	将第 $n$ 个单元格清零
--------	---------------

$S(n)$	将第 $n$ 个单元格值加1
--------	----------------

---

$T(m,n)$	将第 $m$ 个单元格的内容放到第 $n$ 个单元格
----------	----------------------------

$J(m,n,q)$	如果第 $m,n$ 个单元格相同, 则转至第 $q$ 条指令, 否则执行下一指令
------------	---

---

# Instruction Examples

4	5	2	7	
---	---	---	---	--

 $\Rightarrow Z(2) \Rightarrow$ 

4	0	2	7	
---	---	---	---	--

# Instruction Examples

4	5	2	7		$\Rightarrow Z(2) \Rightarrow$	4	0	2	7	
4	0	2	7		$\Rightarrow S(3) \Rightarrow$	4	0	3	7	



# Instruction Examples

4	5	2	7		$\Rightarrow Z(2) \Rightarrow$	4	0	2	7	
4	0	2	7		$\Rightarrow S(3) \Rightarrow$	4	0	3	7	
4	0	3	7		$\Rightarrow T(1,2) \Rightarrow$	4	4	3	7	

# Instruction Examples

4	5	2	7		$\Rightarrow Z(2) \Rightarrow$	4	0	2	7	
4	0	2	7		$\Rightarrow S(3) \Rightarrow$	4	0	3	7	
4	0	3	7		$\Rightarrow T(1,2) \Rightarrow$	4	4	3	7	

# URM

## URM程序的执行

- 除非有跳转，URM按顺序执行指令。

## URM程序的执行

- 除非有跳转，URM按顺序执行指令。
- 如果没有合适的指令可被执行，则停止运行。

## URM程序的执行

- 除非有跳转，URM按顺序执行指令。
- 如果没有合适的指令可被执行，则停止运行。

## URM程序例子

$l_1$ . J(1,2,6)	9	7	0	0	$l_1$	9	8	1	0	$l_2$
$l_2$ . S(2)	9	7	0	0	$l_2$	9	9	1	0	$l_3$
$l_3$ . S(3)	9	8	0	0	$l_3$	9	9	2	0	$l_4$
$l_4$ . J(1,2,6)	9	8	1	0	$l_4$	9	9	2	0	$l_6$
$l_5$ . J(1,1,2)	9	8	1	0	$l_5$	2	9	2	0	$l_7$
$l_6$ . T(3,1)										

这个程序计算了减法 $9 - 7 = 2$ ，为方便，我们在纸带后部标出下一步指令 $l_n$ 。

## 练习计算URM程序

---

 $l_1.$  J(1,2,6) $l_2.$  S(2) $l_3.$  S(3) $l_4.$  J(1,2,6) $l_5.$  J(1,1,2) $l_6.$  T(3,1)

---

2	3	0	0	0
---	---	---	---	---

结果：

## 练习计算URM程序

 $l_1.$  J(1,2,6) $l_2.$  S(2) $l_3.$  S(3) $l_4.$  J(1,2,6) $l_5.$  J(1,1,2) $l_6.$  T(3,1)

2	3	0	0	0
---	---	---	---	---

结果：不停机！

# URM与图灵机

## URM程序计算sg函数

$$f(x) = \begin{cases} 0 & \text{if } x = 0, \\ 1 & \text{otherwise} \end{cases}$$



## URM与图灵机

## URM程序计算sg函数

$$f(x) = \begin{cases} 0 & \text{if } x = 0, \\ 1 & \text{otherwise} \end{cases}$$

①  $J(1, 2, 4)$

②  $Z(1)$

③  $S(1)$

## URM与图灵机

URM程序计算 $x = y$ 特征函数

$$f(x, y) = \begin{cases} 0 & \text{if } x = y, \\ 1 & \text{otherwise} \end{cases}$$

## URM与图灵机

URM程序计算 $x = y$ 特征函数

$$f(x, y) = \begin{cases} 0 & \text{if } x = y, \\ 1 & \text{otherwise} \end{cases}$$

- ①  $J(1, 2, 5)$
- ②  $Z(1)$
- ③  $S(1)$
- ④  $J(1, 1, 6)$
- ⑤  $Z(1)$

## URM与图灵机

URM程序计算 $x + y$

## URM与图灵机

URM程序计算 $x + y$ 

---

 $l_1.$  J(3,2,5) $l_2.$  S(1) $l_3.$  S(3) $l_4.$  J(1,1,1)

---

$x+k$	$y$	$k$	0	...
-------	-----	-----	---	-----

## URM与图灵机

### URM与图灵机等价

每一个URM机器都有一图灵机与之等价，反之亦然。

## URM与图灵机

### URM与图灵机等价

每一个URM机器都有一图灵机与之等价，反之亦然。

### 可计算函数

一个自然数域上的函数 $f$ 称为可计算的(递归的)，如果存在一个UMR程序计算 $f$  (输出结果并正常停机)。

## URM与图灵机

作业:

URM程序计算 $x \leq y$ 特征函数

$$f(x, y) = \begin{cases} 0 & \text{if } x \leq y, \\ 1 & \text{otherwise} \end{cases}$$

URM程序计算 $x-1$ 

$$f(x) = \begin{cases} x-1 & \text{if } x > 0, \\ 0 & \text{if } x = 0 \end{cases}$$

URM程序计算 $x/3$ 

$$f(x) = \begin{cases} x/3 & \text{if } x \text{ is a multiple of } 3, \\ \text{undefined} & \text{otherwise} \end{cases}$$