NP完全性

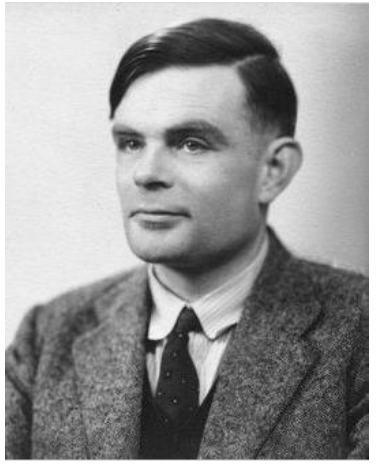
Teacher：张明卫
Email：zhangmw@swc.neu.edu
Office: 东北大学浑南校区信息楼B434

CONTENTS

PART 01 问题的复杂性分类

1.1 图灵机计算模型

阿兰·图灵（Alan Turning）



- 图灵被誉为计算机科学之父、人工智能之父。1912年6月23日生于英国帕丁顿，1931年进入剑桥大学国王学院，1938年在美国普林斯顿大学取得博士学位，二战爆发后返回剑桥，曾协助军方破解德国的著名密码系统Enigma，帮助盟军取得了二战的胜利。1954年6月7日在曼彻斯特去世。
- 图灵是计算机逻辑的奠基者，提出了“图灵机”和“图灵测试”等重要概念。人们为纪念其在计算机领域的卓越贡献而专门设立了“图灵奖”。
- 图灵还是一位世界级的长跑运动员。他的马拉松最好成绩是2小时46分3秒，比1948年奥运会金牌成绩慢11分钟。1948年的一次跨国赛跑比赛中，他跑赢了同年奥运会银牌得主汤姆·理查兹（Tom Richards）。

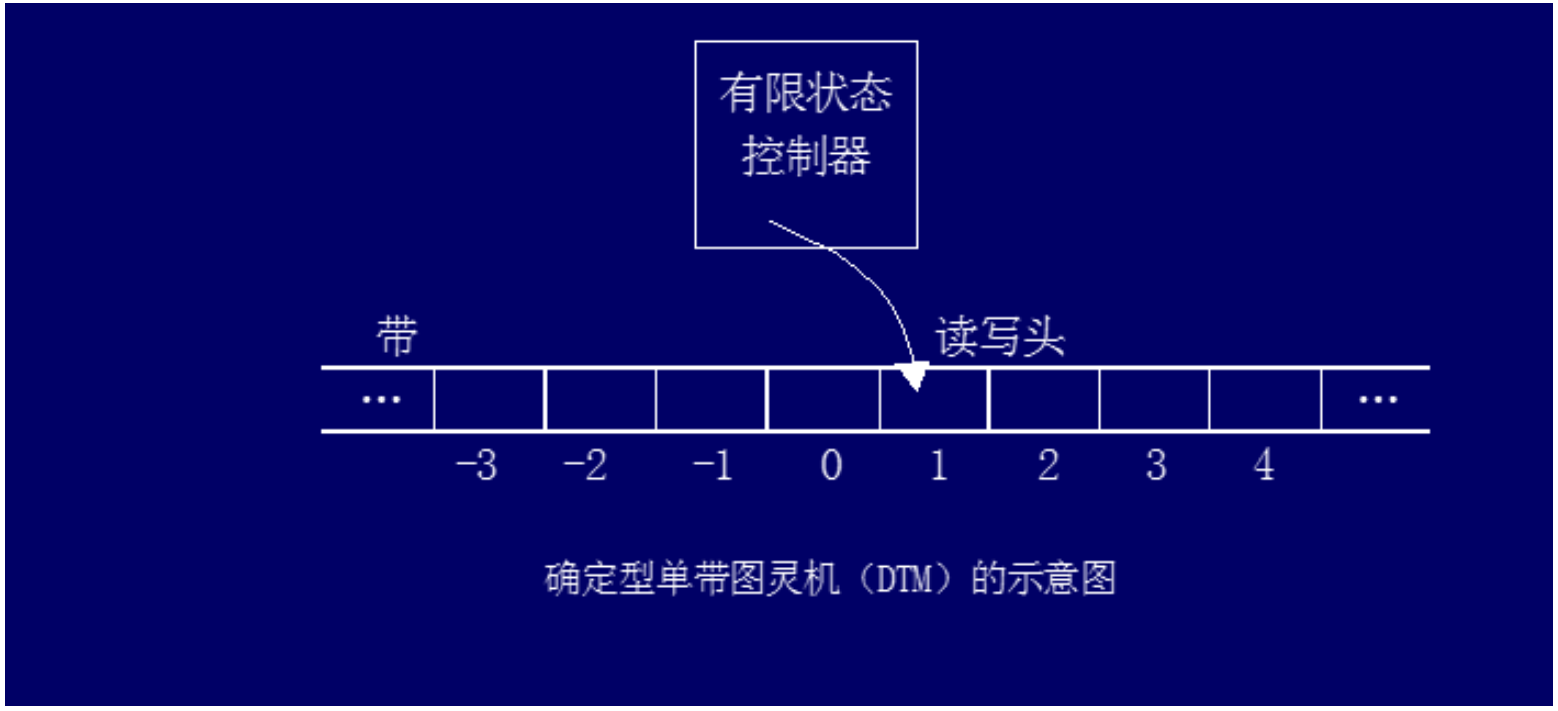
1.1 图灵机计算模型

图灵机产生背景

- 希尔伯特问题：在1900年8月巴黎国际数学家代表大会上，希尔伯特发表了题为《数学问题》的著名讲演。他根据过去特别是十九世纪数学研究的成果和发展趋势，提出了23个最重要的数学问题。
- 希尔伯特第十问题：能否通过有限步骤来判定不定方程是否存在有理整数解？1970年。苏联数学家马蒂塞维奇最终证明：在一般情况答案是否定的。
- 图灵机：1935-1936年间，图灵在剑桥大学国王学院研究希尔伯特判定问题。1936年5月，图灵完成论文《论可计算数及其在判定问题上的应用》，提出图灵机作为通用计算模型，并基于图灵机重新定义了可计算函数，提出可计算问题（即数学上的某些计算问题，是不是只要给数学家足够长的时间，就能够通过“有限次”的简单而机械的演算步骤而得到最终答案呢？），同时给出了一个图灵机无法判定的问题，即停机问题。

1.1 图灵机计算模型

■在进行问题的计算复杂性分析之前，首先必须建立求解问题所用的计算模型，包括定义该计算模型中所用的基本运算。

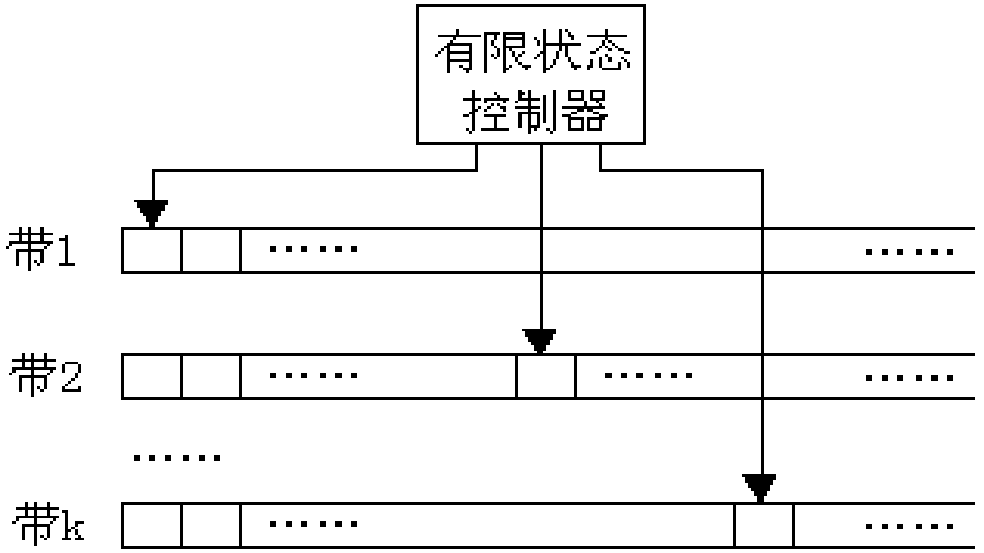


确定型单带图灵机（DTM）的示意图

1.1 图灵机计算模型

确定性K带图灵机(DTM)的组成

- k条(k>=1)存储带
  - 双向无限延长
  - 上有一个个小方格
  - 每个小方格可存储一个数字/字母
- 一个控制器
  - 包含k个读写头，可以读、写、更改存储带上每一格的数字/字母
  - 可以接受设定好的程序语句（定义好的状态转移函数）；
  - 可以存储当前自身的状态；
  - 可以根据读到的字母/数字变换自身的状态
  - 可以沿着存储带一格一格地左移/右移



1.1 图灵机计算模型

确定性K带图灵机(DTM)的工作步骤

1. 准备

(1) 存储带上符号初始化；

(2) 控制器设置好自身当前状态（初始化状态）；

(3) 读写头置于存储带上的起始位置；

(4) 准备好工作程序（定义好状态转移函数）；

2. 反复执行以下工作直到停机

(1) 读写头读出存储带上当前方格中的字母/数字；

(2) 根据自身当前状态和所读到的字符，找到相应的程序语句；

(3) 根据相应程序语句，做三个动作：

- 改变有限状态控制器中的状态。
- 清除当前读写头下的方格中原有带符号并写上新的带符号。
- 独立地将任何一个或所有读写头，向左移动一个方格(L)或向右移动一个方格(R)或停在当前单元不动(H)。

1.1 图灵机计算模型

确定性K带图灵机(DTM)的定义

■ **k带图灵机**可形式化地描述为一个7元组$(Q, T, I, \delta, b, q_0, q_f)$，其中:

(1) Q是有限个状态的集合;

(2) T是有限个带符号的集合;

(3) I是输入符号的集合，$I \subseteq T$，不包含特殊的空白符;

(4) b是唯一的空白符，$b \in T-I$;

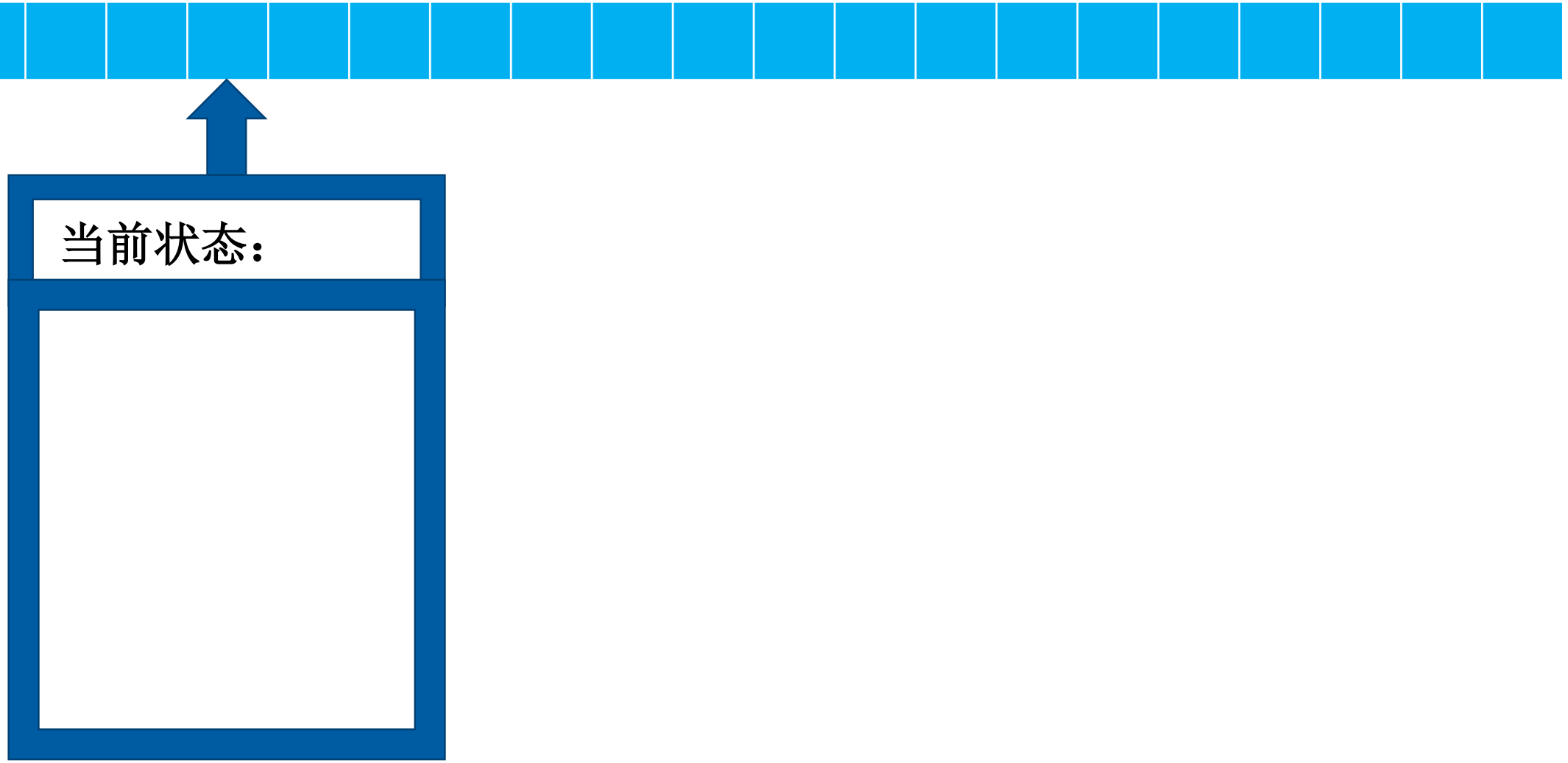
(5) q_0是初始状态;

(6) q_f是终止（接受或拒绝）状态。

(7) δ是状态转移函数。它是从$Q \times T^k$的某一子集映射到$Q \times (T \times \{L, R, H\})^k$的函数。

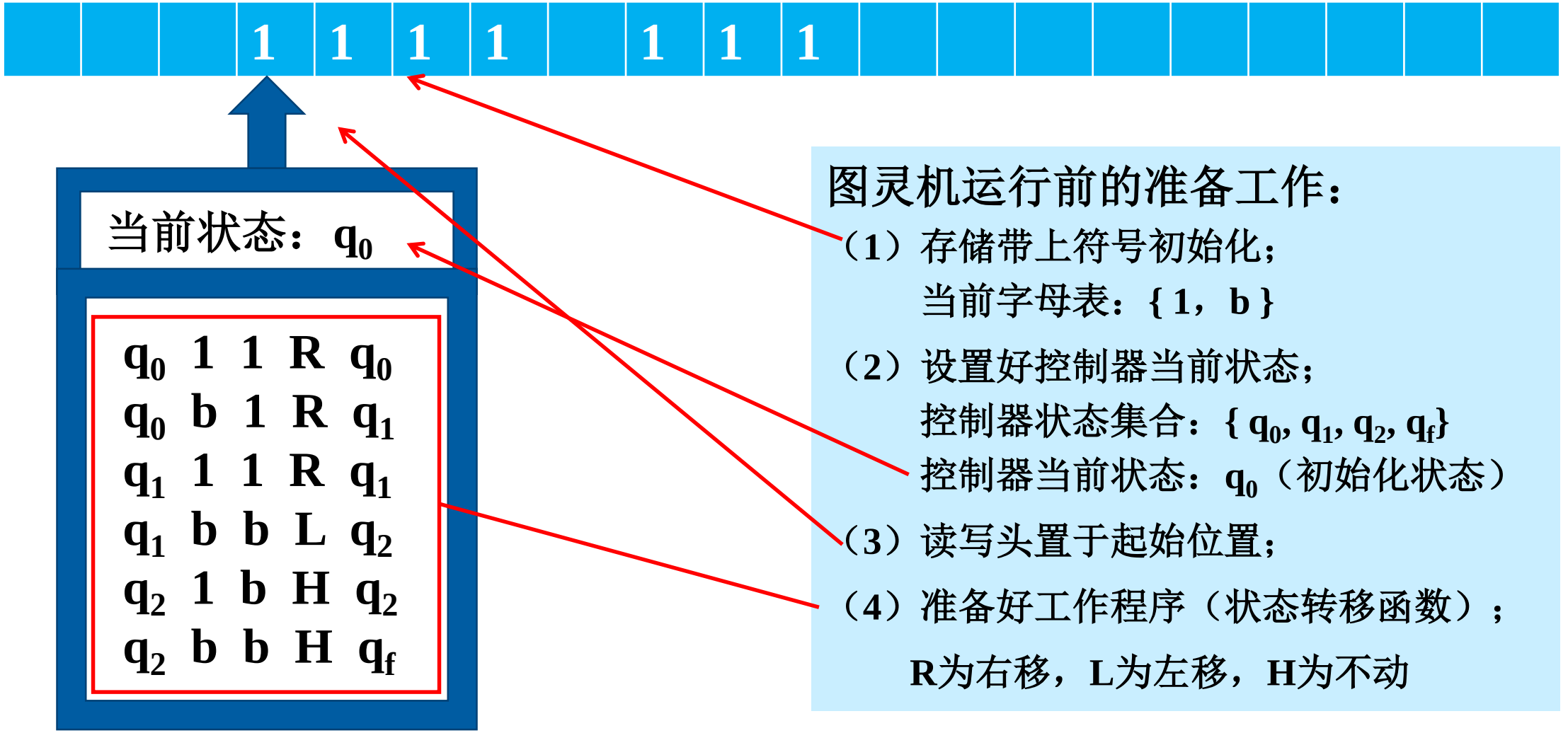
1.1 图灵机计算模型

图灵机示例1



1.1 图灵机计算模型

图灵机示例1



图灵机运行前的准备工作：

（1）存储带上符号初始化；

当前字母表：{ 1，b }

（2）设置好控制器当前状态；

控制器状态集合：{ q_0, q_1, q_2, q_f}

控制器当前状态：q_0（初始化状态）

（3）读写头置于起始位置；

（4）准备好工作程序（状态转移函数）；

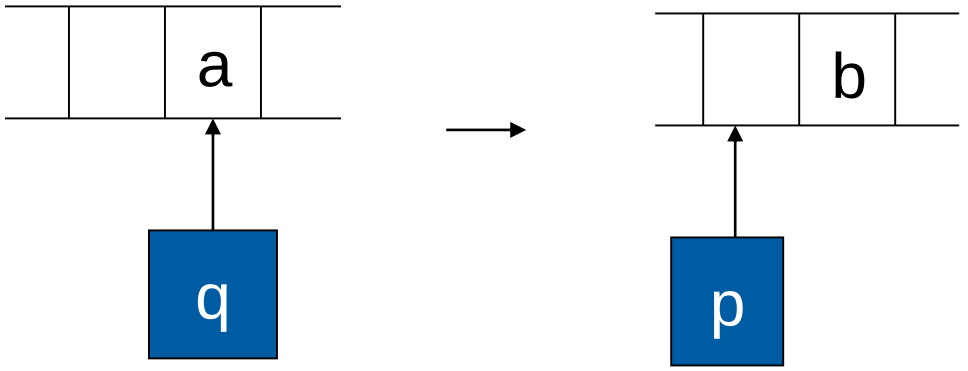
R为右移，L为左移，H为不动

1.1 图灵机计算模型

图灵机示例1

当前状态：q_0

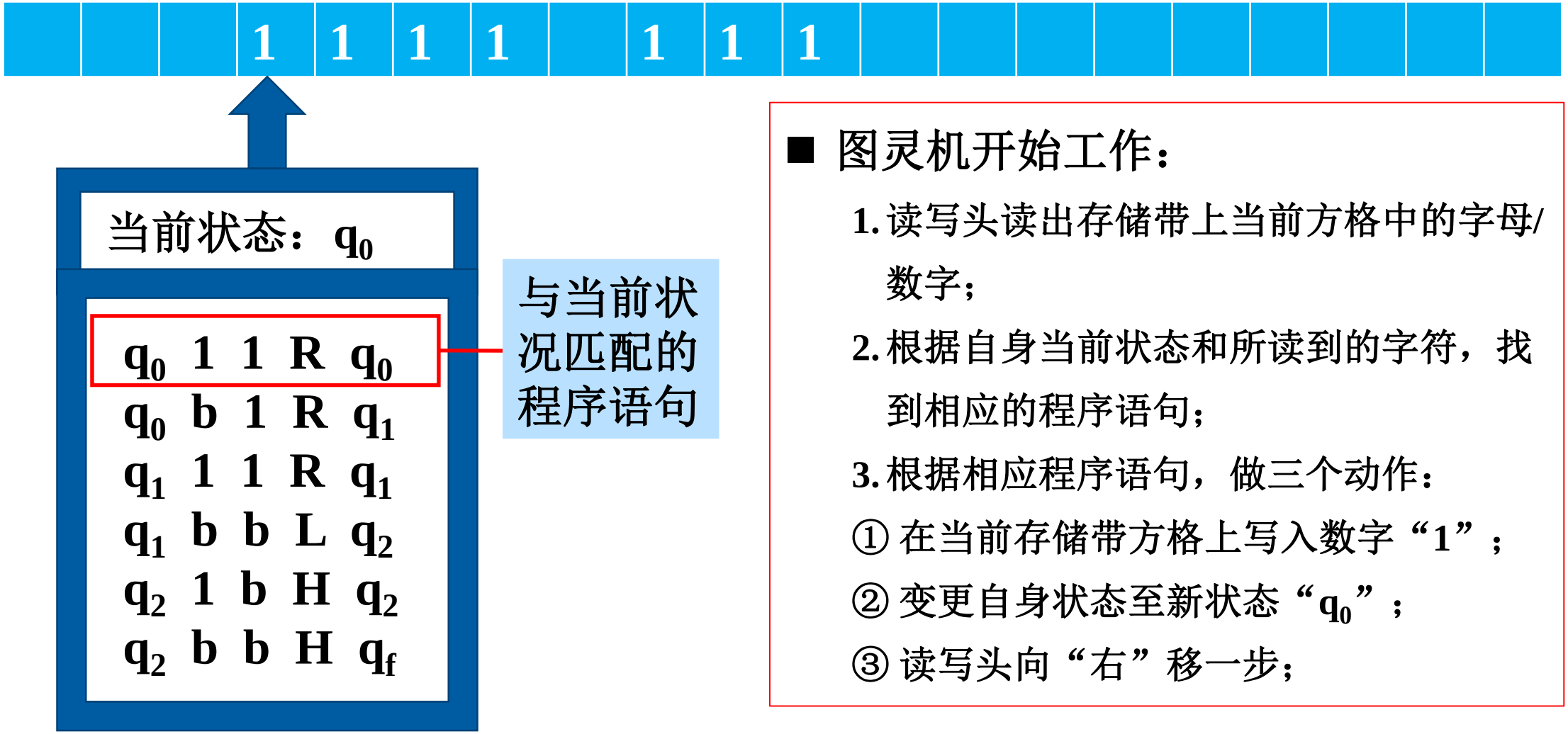
q_0	1	1	R	q_0
q_0	b	1	R	q_1
q_1	1	1	R	q_1
q_1	b	b	L	q_2
q_2	1	b	H	q_2
q_2	b	b	H	q_f



- $\delta(q, a) = (p, b, L)$ ，表示：如果读写头读到符号为a，当前状态为q，则修改a为b，转移状态到p，向左移动一位；
- $\delta(q_0, 1) = (q_0, 1, R)$

1.1 图灵机计算模型

图灵机示例1



- 图灵机开始工作：
  1. 读写头读出存储带上当前方格中的字母/数字；
  2. 根据自身当前状态和所读到的字符，找到相应的程序语句；
  3. 根据相应程序语句，做三个动作：

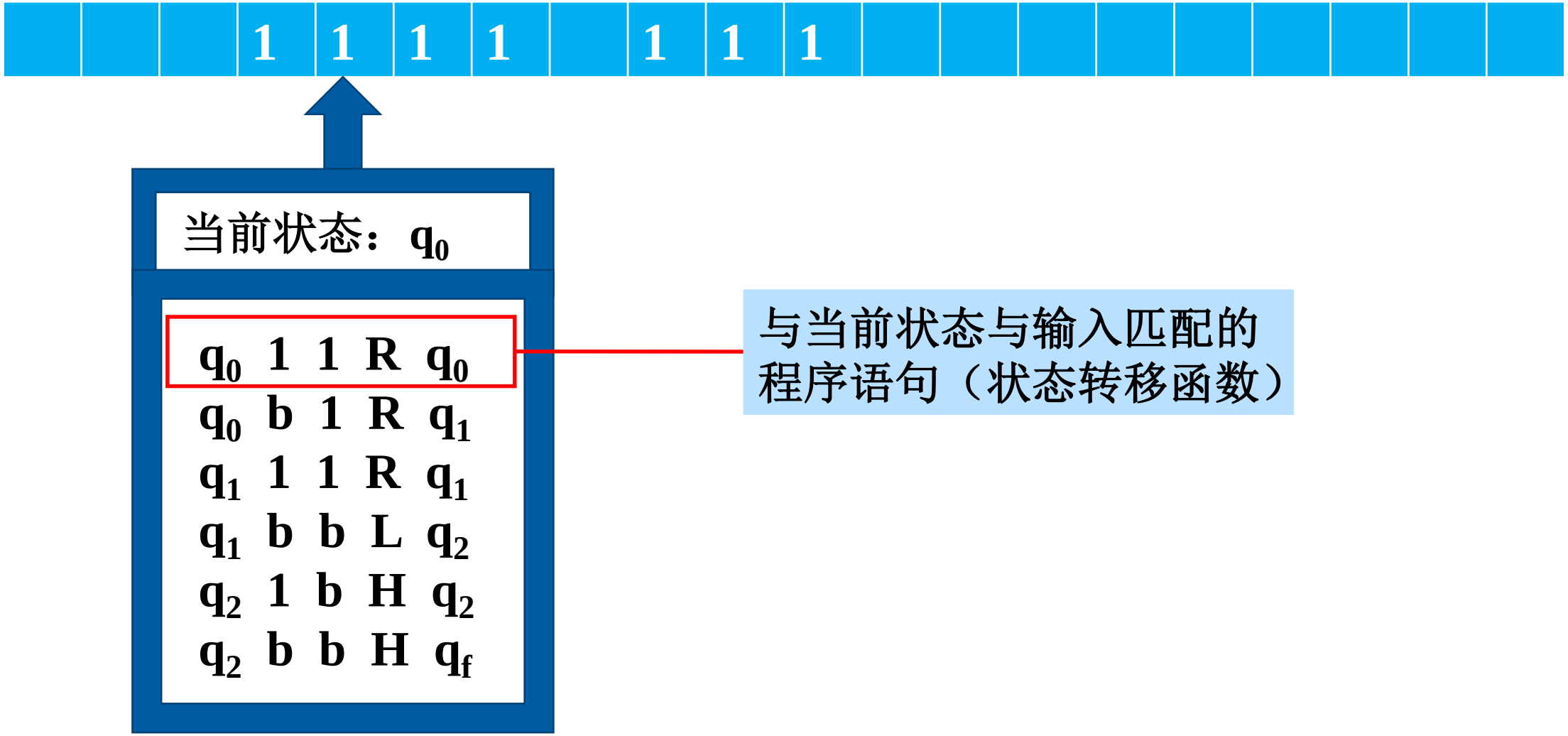
  ① 在当前存储带方格上写入数字“1”；

  ② 变更自身状态至新状态“q_0”；

  ③ 读写头向“右”移一步；

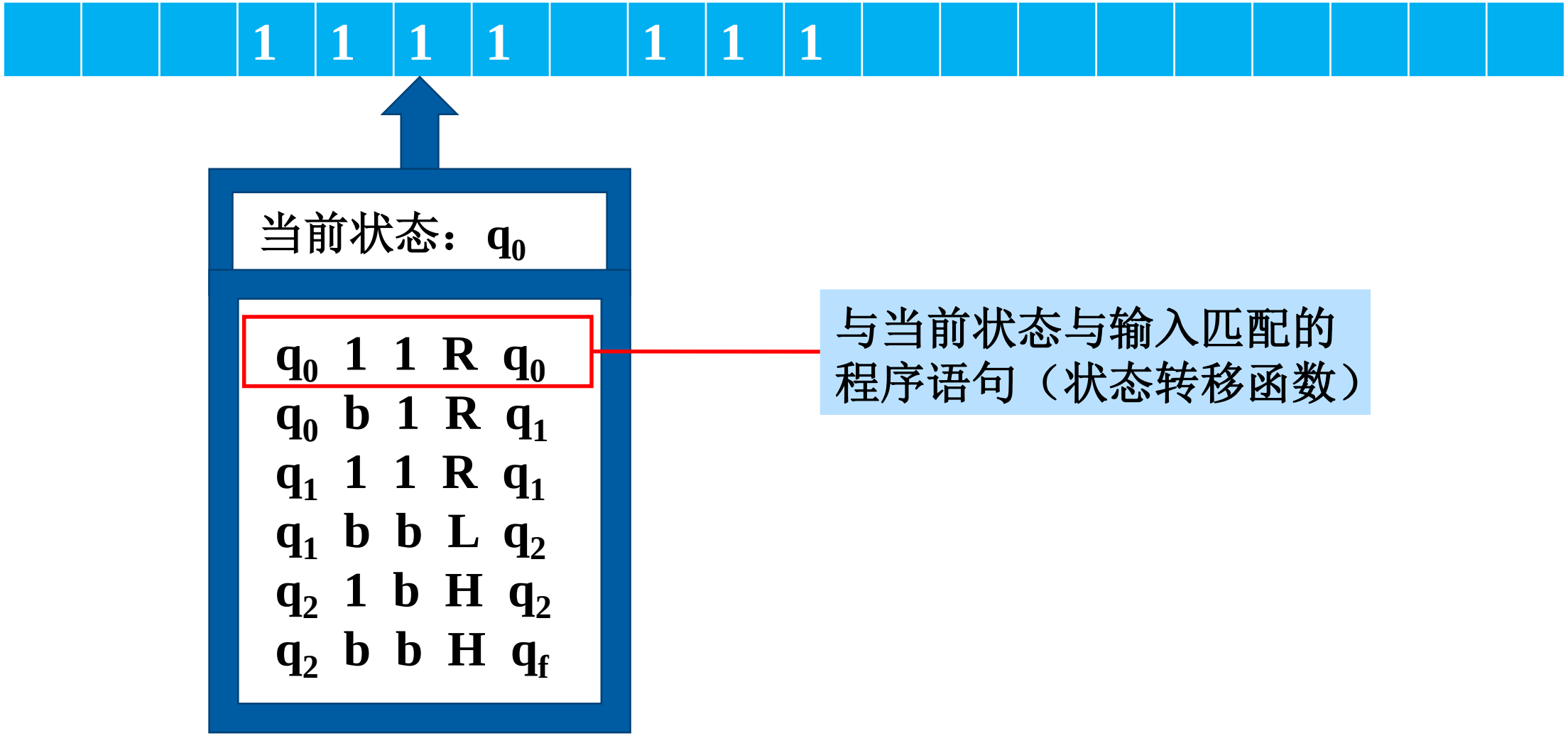
1.1 图灵机计算模型

图灵机示例1



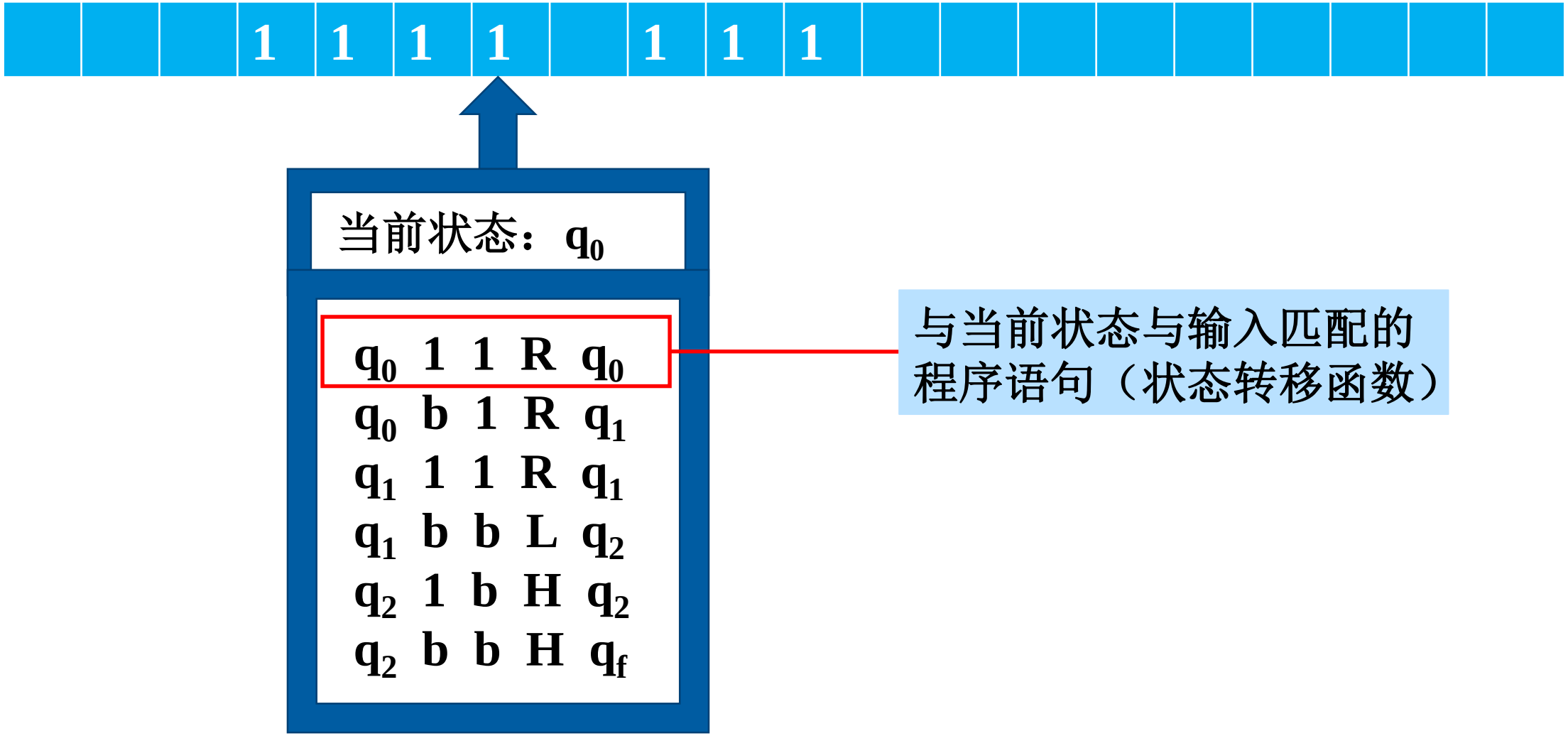
1.1 图灵机计算模型

图灵机示例1



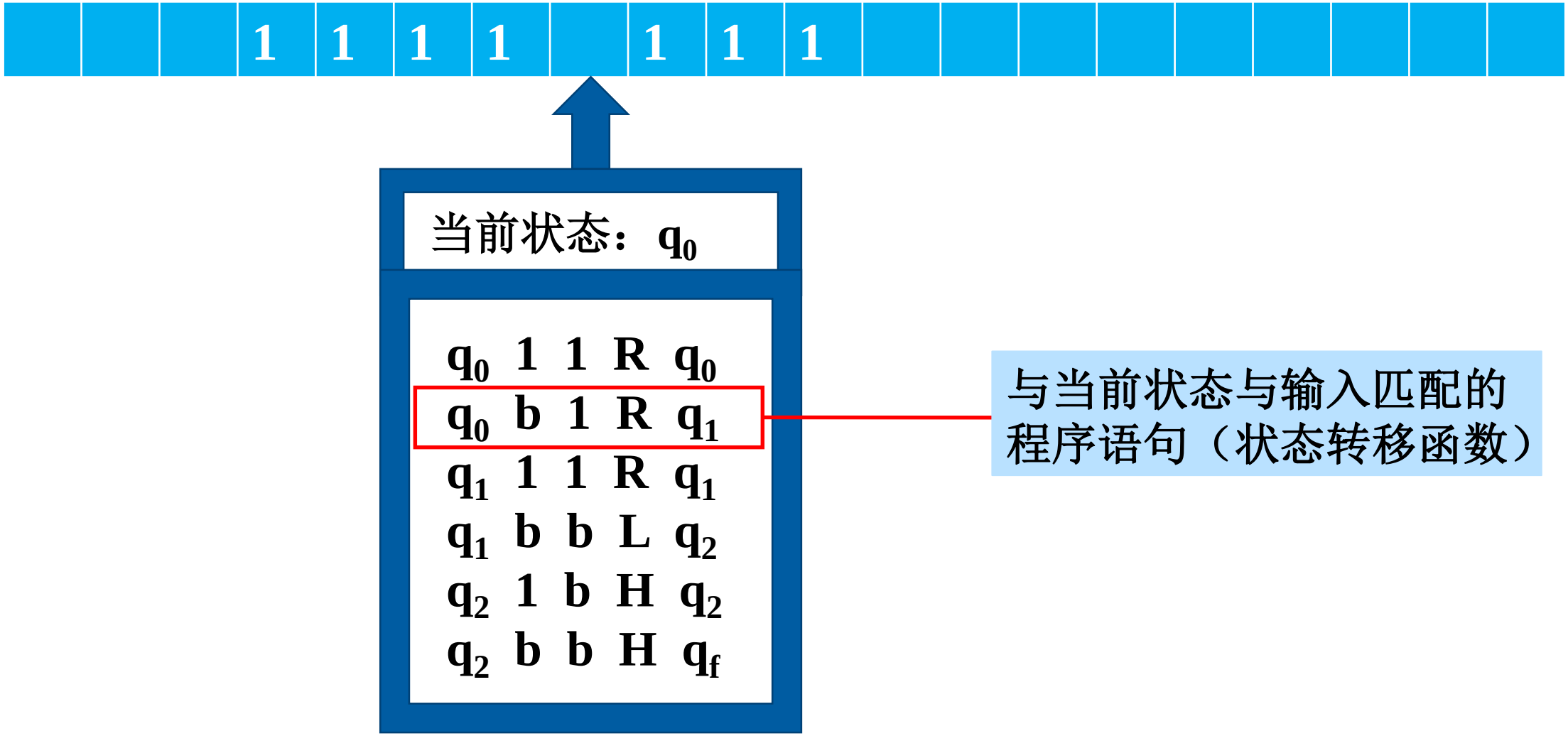
1.1 图灵机计算模型

图灵机示例1



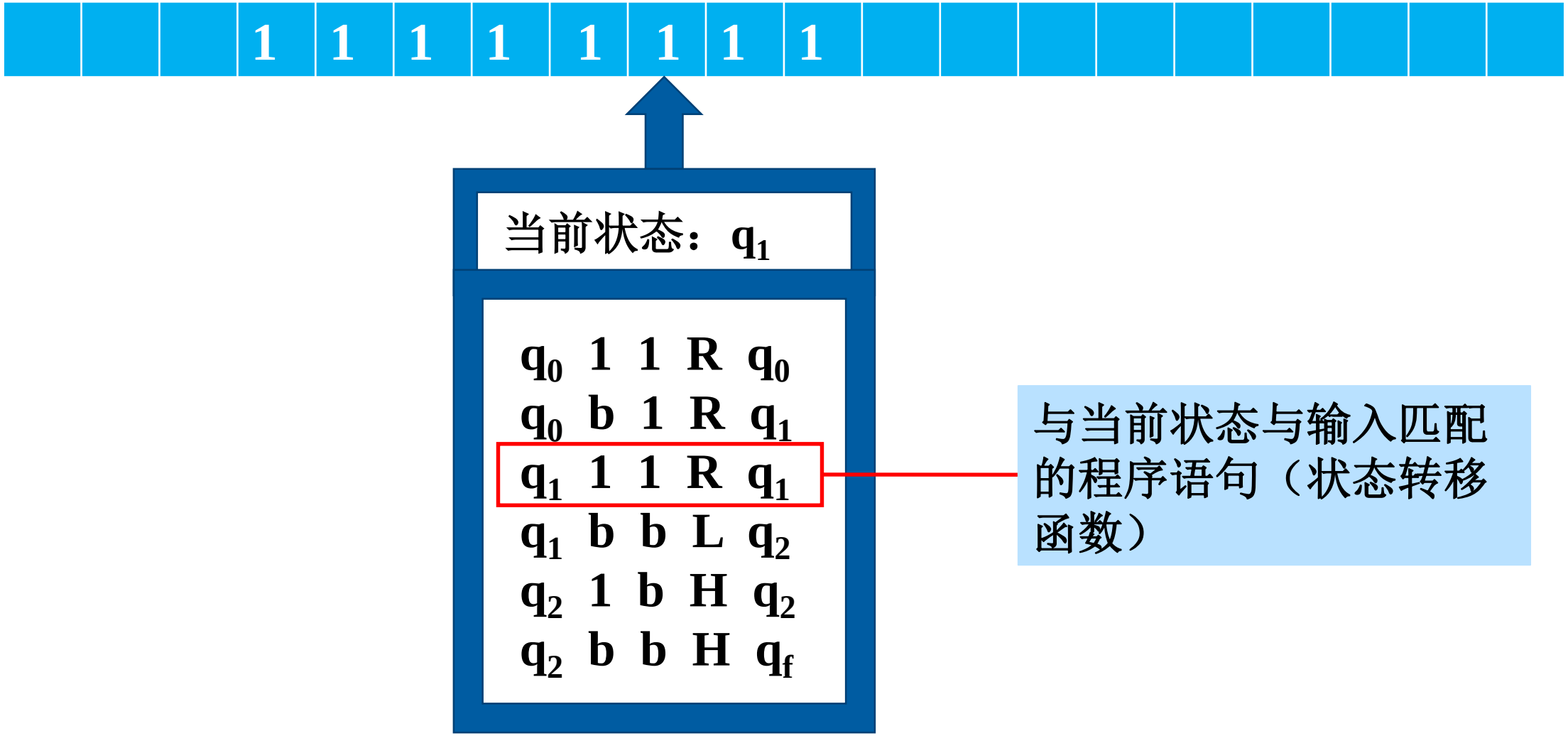
1.1 图灵机计算模型

图灵机示例1



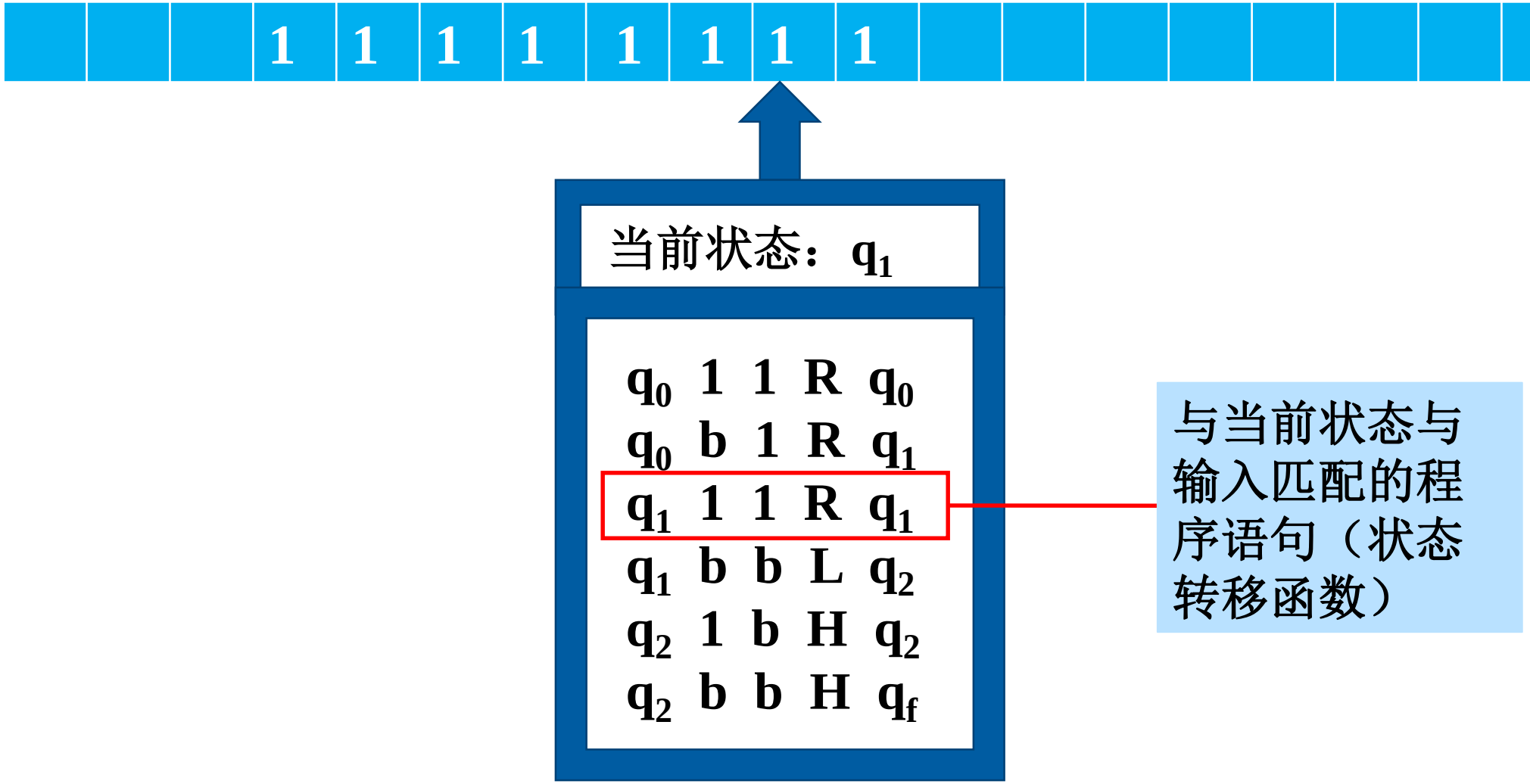
1.1 图灵机计算模型

图灵机示例1



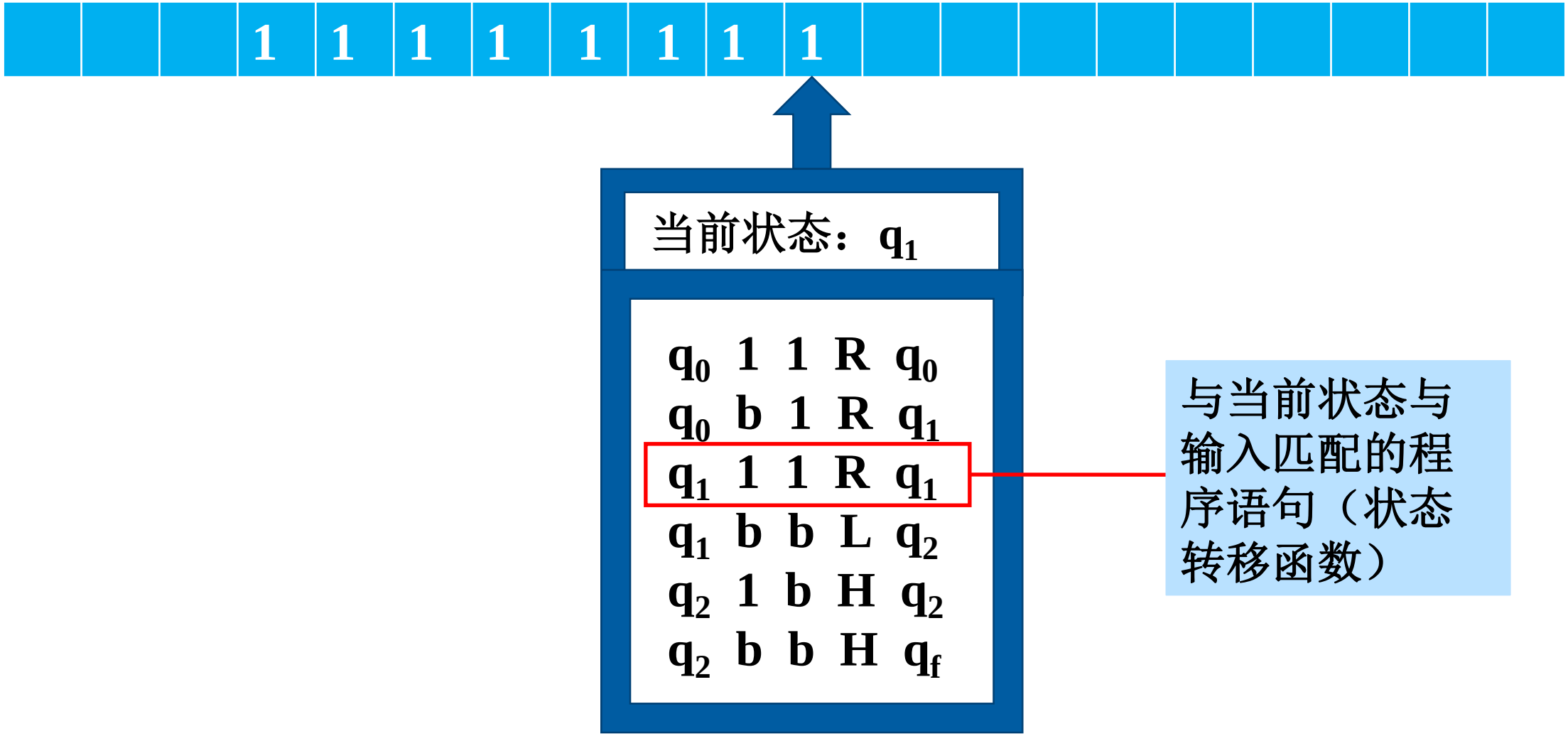
1.1 图灵机计算模型

图灵机示例1



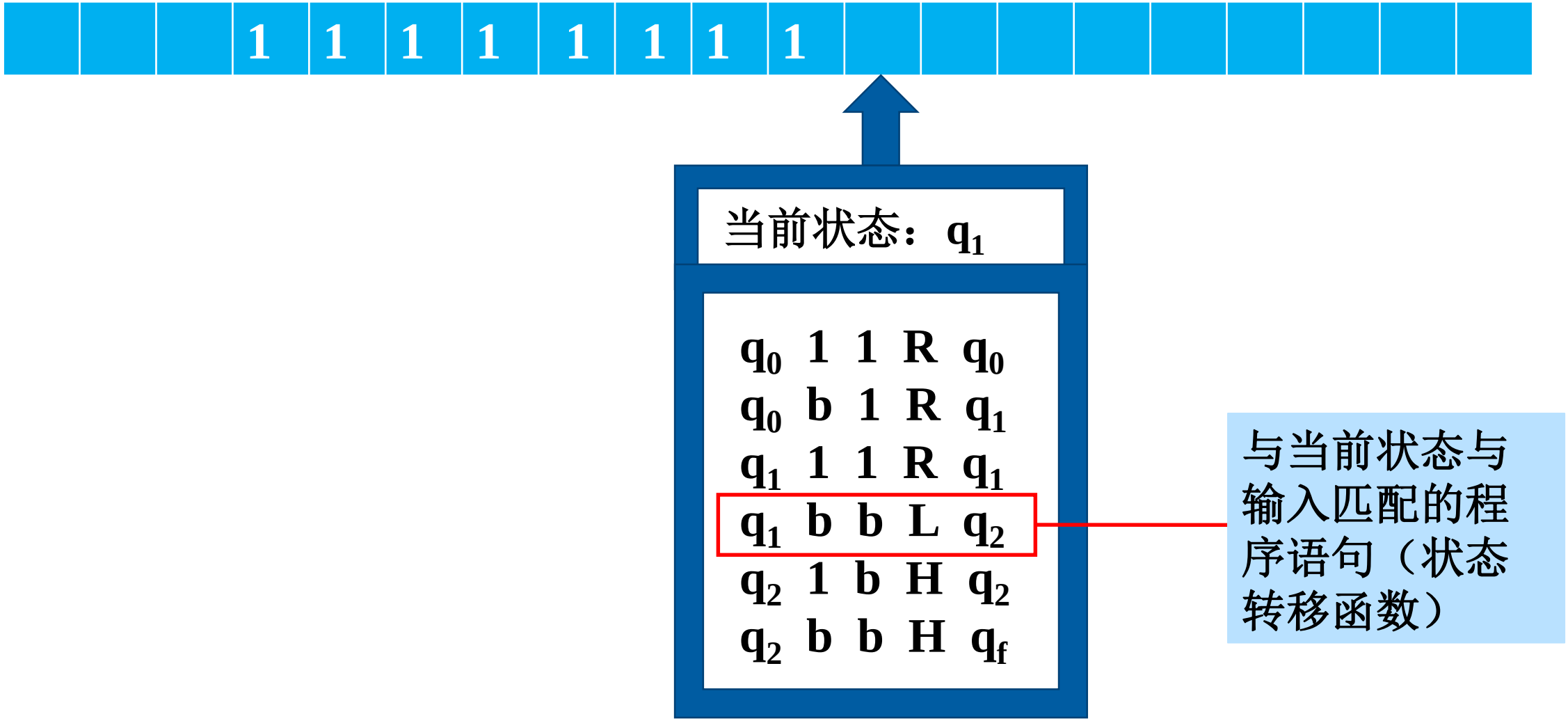
1.1 图灵机计算模型

图灵机示例1



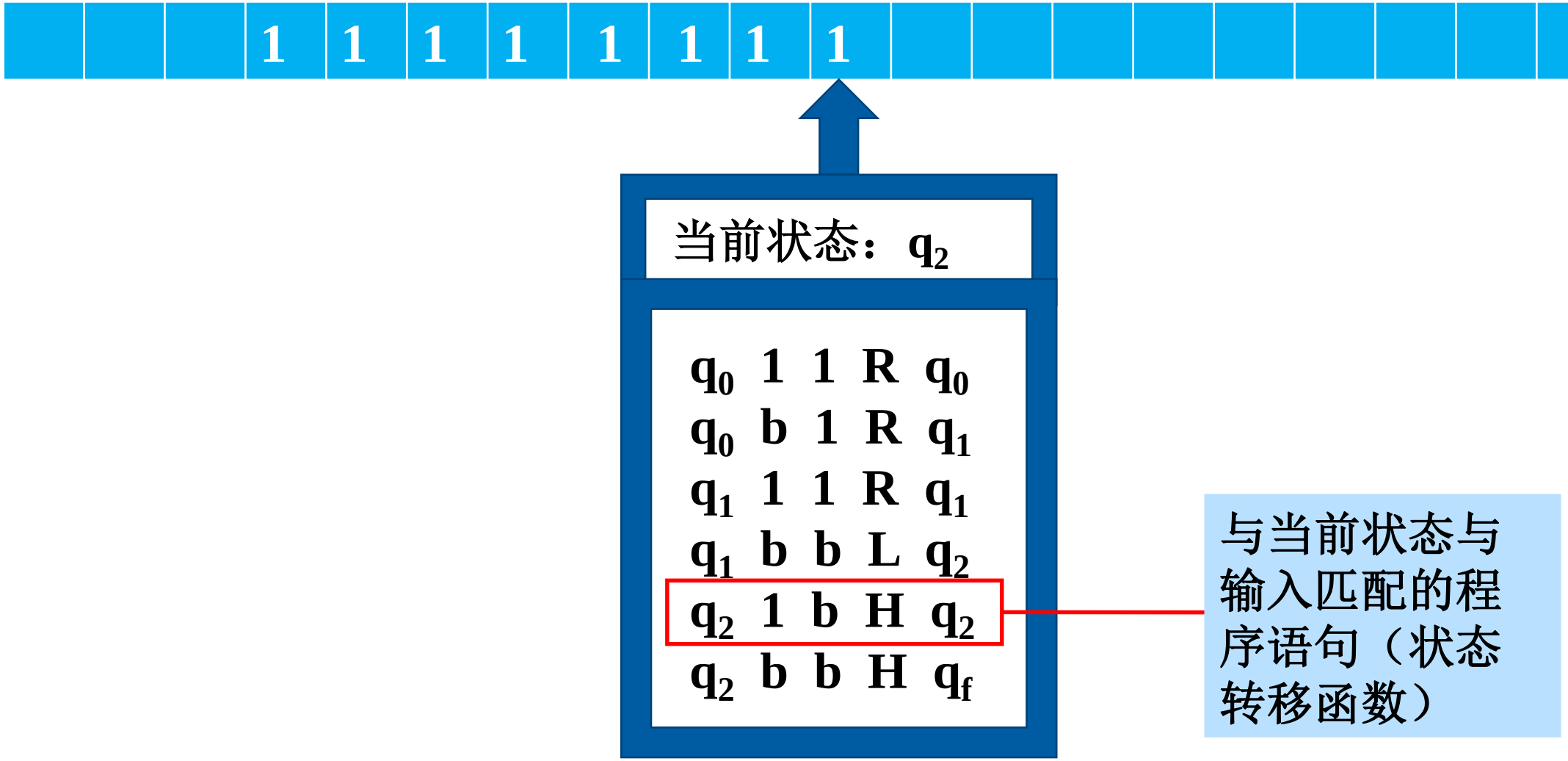
1.1 图灵机计算模型

图灵机示例1



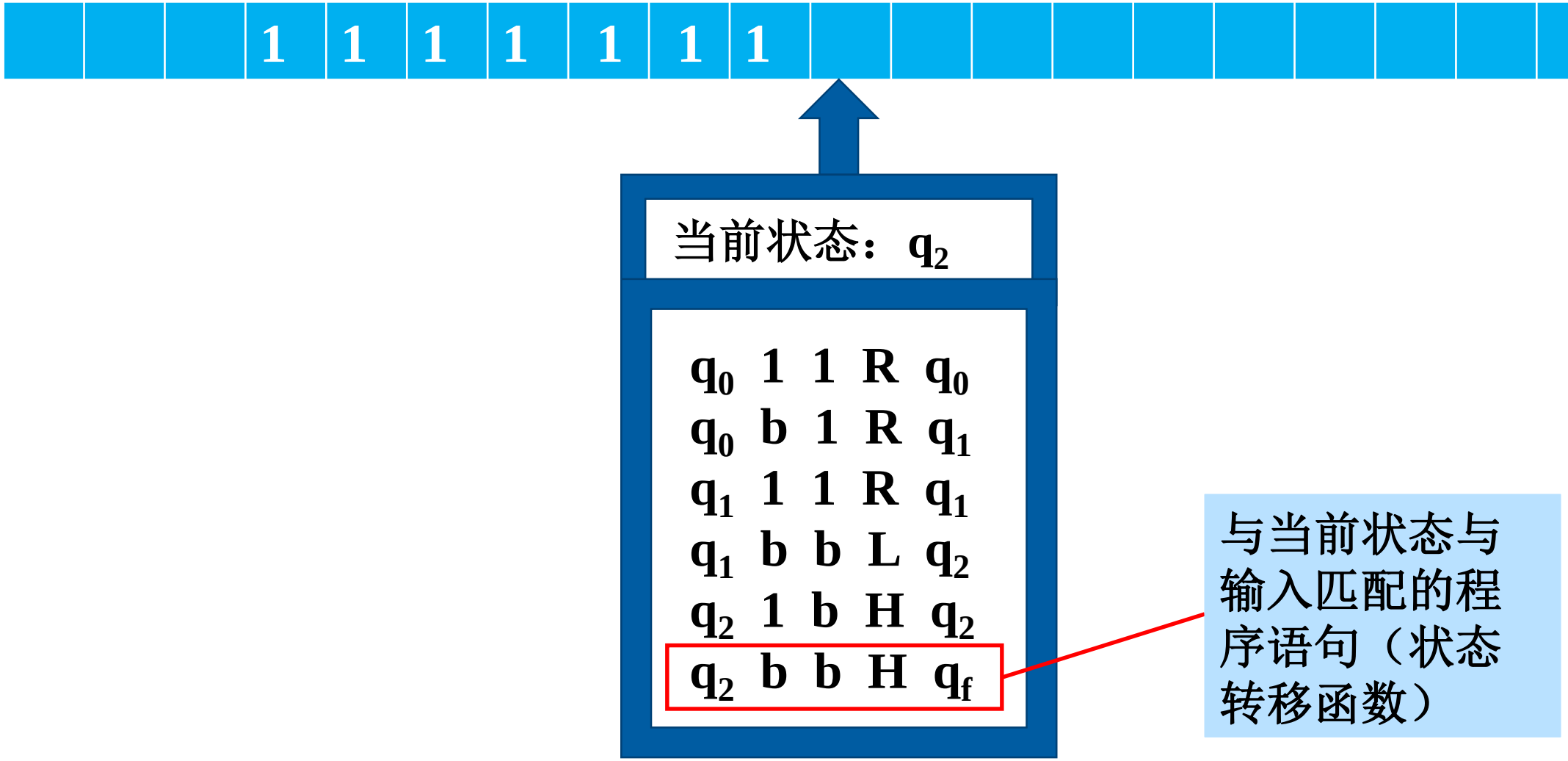
1.1 图灵机计算模型

图灵机示例1



1.1 图灵机计算模型

图灵机示例1



1.1 图灵机计算模型

图灵机示例1

- 这个过程做了一件什么事情？

4+3 =7

当前状态：q_f

q_0	1	1	R	q_0
q_0	b	1	R	q_1
q_1	1	1	R	q_1
q_1	b	b	L	q_2
q_2	1	b	H	q_2
q_2	b	b	H	q_f

图灵机停机意味着什么？

- 停机表示计算完毕，存储带上保留的是计算结果。
- 也就是说，停机意味着得出计算结果！

1.1 图灵机计算模型

图灵机示例2

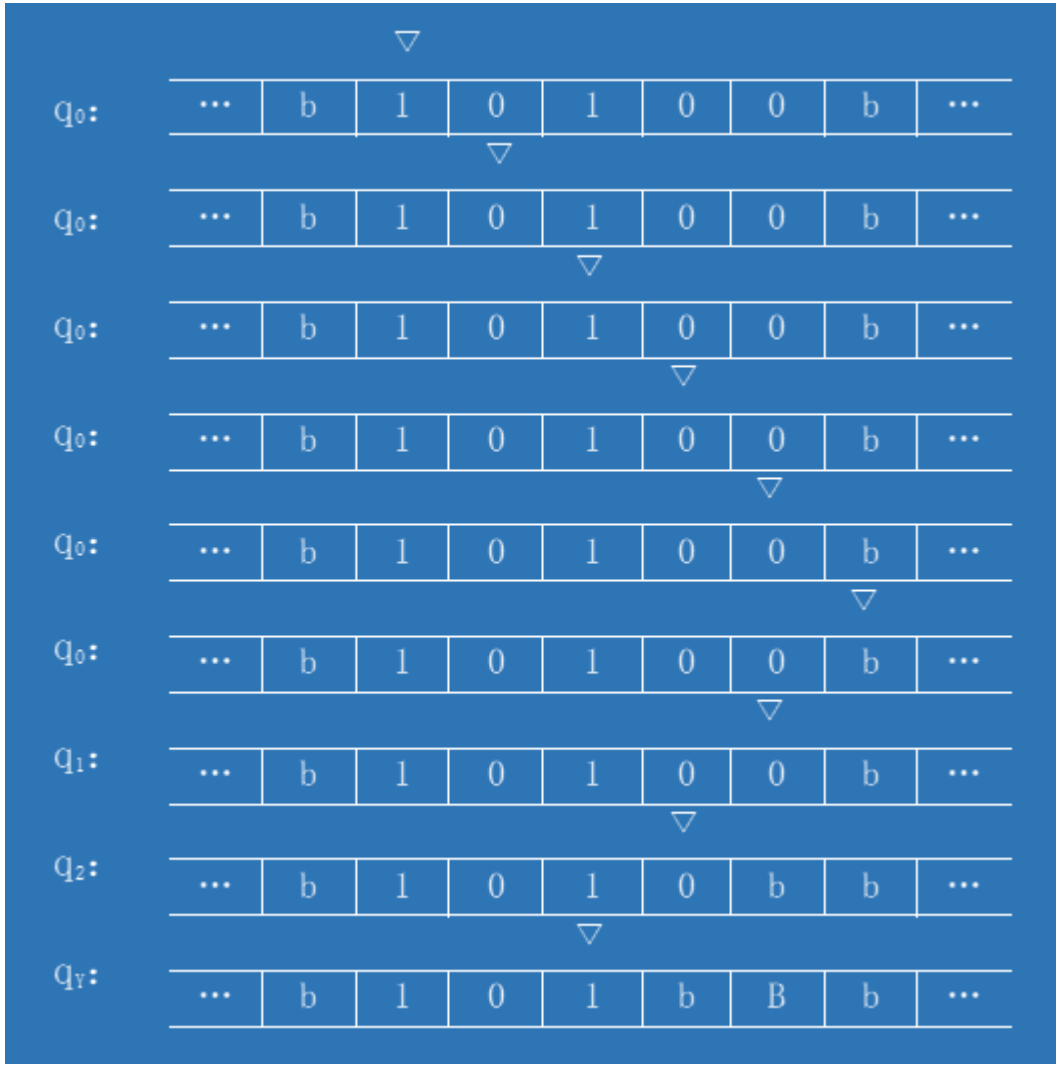
- 设有一带图灵机为M；
- 输入：是一个字符串x，放在从1到|x|的一段带方格中，每一个方格放一个符号。所有其它的方格开始时放着空白符；
- 运行：
  - 程序从状态q_0开始运行，这时读写头扫描方格1，然后计算一步一步地进行；
  - 如果当前状态q是q_Y或q_N，那末计算结束，并且当$q=q_Y$时答案为“是”，当$q=q_N$时答案为“否”；
  - 否则，当前状态q属于$Q - \{q_Y, q_N\}$，同时在被扫描的带方格中有一个符号s，$s \in T$，则$\delta(q,s)$为一个确定的值。假设$\delta(q,s)= (q', s', \Delta)$，那么读写头擦去s，并在这个方格里写入s’，然后移动一格。当$\Delta=-1$时，向左移动一格，当$\Delta=1$时，向右移动一格。与此同时，有限状态控制器从状态q变成q’。这就完成了一步计算，并为下一步计算做好了准备。

1.1 图灵机计算模型

图灵机示例2

- M的转移函数δ用表格形式描述。图中说明M在输入x=10100上的计算，给出了每一步前后的状态，读写头位置以及带的非空白部分的内容。

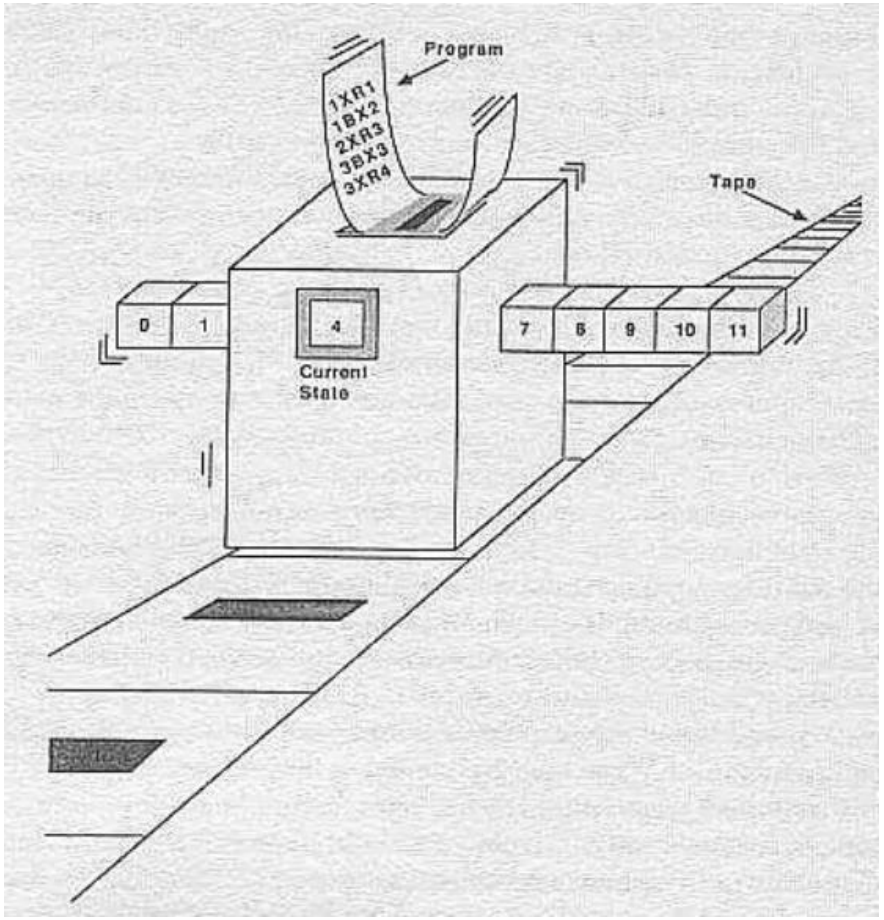
q	0	1	b
q_0	$(q_0,0,+1)$	$(q_0,1,+1)$	$(q_1,b,-1)$
q_1	$(q_2,b,-1)$	$(q_3,b,-1)$	$(q_N,b,-1)$
q_2	$(q_Y,b,-1)$	$(q_N,b,-1)$	$(q_N,b,-1)$
q_3	$(q_N,b,-1)$	$(q_N,b,-1)$	$(q_N,b,-1)$



1.1 图灵机计算模型

图灵机的作用

- 对于算法来说，图灵机是最惯用的理论模型。研究图灵机的主要目的是对“算法”、“有效过程”这样的直观概念给出精确的数学定义，从而精确刻画可计算性与可判定性等基本概念。
- 根据Church论题，图灵机在计算能力上等价于数字计算机，故利用图灵机可以研究计算机的能力和局限性。
- 对图灵机的研究集中在两个方面：第一，研究图灵机所定义的语言类，该语言类称为递归可枚举集合。第二，研究图灵机所计算的函数类，该函数类称为部分递归函数。



1.2 可计算问题与不可计算问题

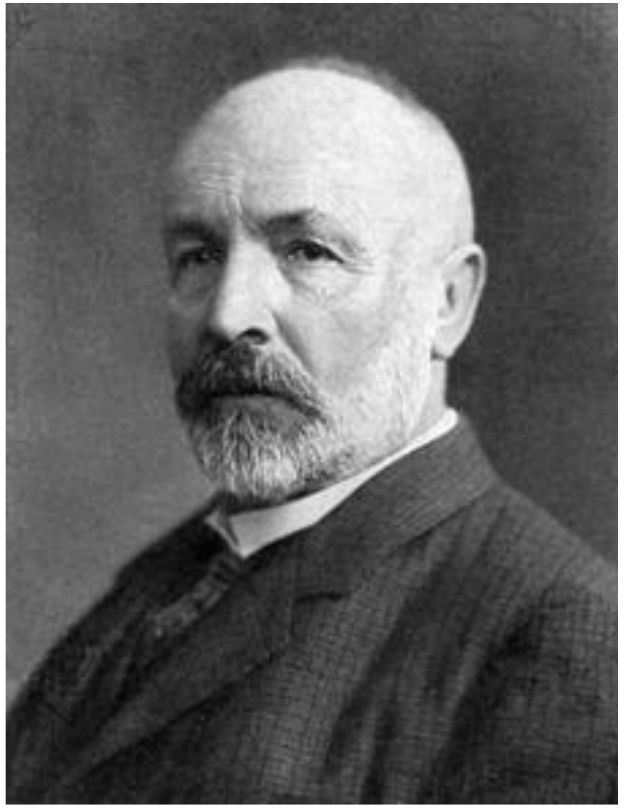
第三次数学危机

■集合论

◆十九世纪下半叶，康托尔创立了著名的集合论。刚产生时，曾遭到许多人的猛烈攻击，甚至使他产生了极度的抑郁。

◆后来数学家们发现，从自然数与康托尔集合论出发可建立起整个数学大厦。（法国著名数学家庞加莱，在1900年的国际数学家大会上宣称：“借助集合论概念，我们可以建造整个数学大厦。今天，我们可以说绝对的严格性已经达到了”）。

◆一切数学成果可建立在集合论基础上。



格奥尔格·康托尔
德国数学家，集合论的创始人

1.2 可计算问题与不可计算问题

第三次数学危机

■一个故事

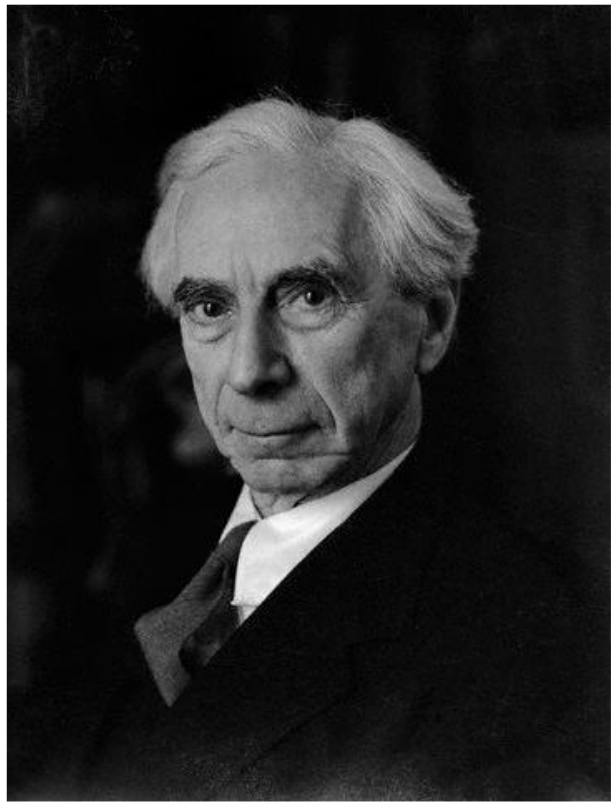
◆塞尔维亚有一位理发师：他只给所有不给自己理发的人理发，不给那些给自己理发的人理发。

◆罗素问：这个理发师要不要给自己理发呢？

■罗素悖论

◆S由一切不是自身元素的集合所组成。

◆罗素问：S是否属于S呢？



伯特兰·罗素（1872—1970）
英国哲学家、数学家、逻辑学家、
历史学家，无神论或者不可知论者，
和平主义社会活动家

1.2 可计算问题与不可计算问题

第三次数学危机

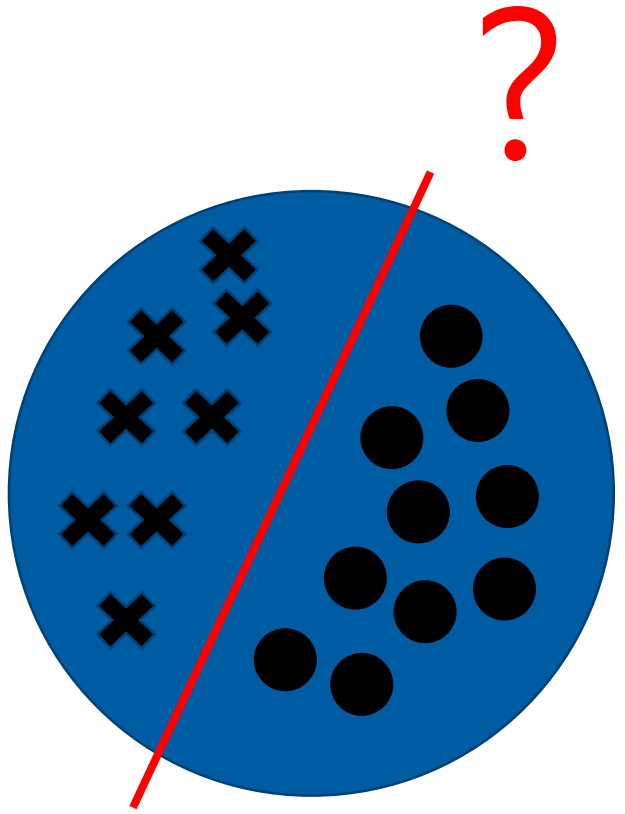
- 崩溃的数学家们想：那能不能找到一个完备的系统，从而构建整个数学大厦呢？这样的系统是否存在呢？哲学家哥德尔给出了结论。
- 哥德尔不完备性定理：1931年哥德尔成功证明了：任何一个数学系统，只要它是从有限的公理和基本概念中推导出来的，并且从中能推证出自然数系统，就可以在其中找到一个命题，对于它我们既没有办法证明，又没有办法推翻（既没有办法证“真”，又没有办法证“伪”）。
- 结论：不要试图寻找一个完美的系统？

库尔特 哥德尔（Kurt Godel）
1906年4月28日—1978年1月14日
美国数学家、逻辑学家和哲学家

1.2 可计算问题与不可计算问题

问题的可计算性

- 假设这个圆代表一个系统，在这个系统中，有些问题既不能被证真，也不能被证伪的，有些问题是可以被证真或证伪的，那么这个边界到底在哪里？
- 怎么判定一个未解的问题是否真的有解？
- 这是一个非常关键的问题，在计算机中我们把这个问题归结到可计算问题。



1.2 可计算问题与不可计算问题

问题的可计算性

- 设函数f的定义域是D，值域是R，如果存在一种算法，对D中任意给定的x，都能计算出f(x)的值，则称函数f是可计算的。否则，就是不可计算的。
- 在一个系统中，既存在可计算问题，也存在不可计算问题，当出现一个未解的问题时，那么如何判定它是否可计算呢？
- 数理学家们给出一个研究思路：为“计算”建立一个数学模型，称为计算模型，然后证明，凡是这个计算模型能够完成的任务，就是可计算的任务。不可完成的任务，就是不可计算的任务。

图灵提出的图灵机模型！

1.2 可计算问题与不可计算问题

问题的可计算性

- 哪些问题是计算机可计算的，是计算机科学的一个基本问题。
- **Turing论题：**一个问题是可计算的，当且仅当在图灵机上经过有限步骤后得到正确的结果。
- “有限步骤”相当宽松，因而，Turing论题中界定的可计算问题几乎包括了人类遇到的所有问题。

1.2 可计算问题与不可计算问题

停机问题

- 判断任意一个程序是否能在有限的时间内结束运行。
- 是否存在一个算法，对于给定程序P和它的任意输入I，能够判断P会在有限时间内结束或者死循环 。
- 不存在。证明：
- 假设算法A可以解决停机问题，也就是对于任意的程序 P 和输入 I：

$$A(P, I) = \begin{cases} 1, & \text{if P halts for input I} \\ 0, & \text{if P can't halt for input I} \end{cases}$$

1.2 可计算问题与不可计算问题

停机问题

- 程序本身也是一种数据，它可以作为输入。所以A可以判断P作为P的输入时，P是否会停机。
- 我们定义另一个程序Q：

$$Q(P)=\begin{cases}\text{停机，if } A(P,P)=0\\ \text{死循环，if } A(P,P)=1\end{cases}$$

Replace P by Q:

$$Q(Q)=\begin{cases}\text{停机，if } A(Q,Q)=0\\ \text{死循环，if } A(Q,Q)=1\end{cases}$$

```
int A(procedure, Input);
// A有两种返回值，死循环(0) 或 停机(1)
int Q(P)
{
   if (A(P,P) == 0){ // 如果H死循环
       return 1; // 此时会停机
    }
   else{ // 否则
       while(1){} // 此时会死循环
    }
}
```

1.2 可计算问题与不可计算问题

不可计算问题

- 判断一个程序中是否包含计算机病毒：能够检测现在已知的那些病毒，但不存在一个病毒检测程序，能够检测出所有未来的新病毒。
- 接受问题（acceptance problem）：如果给予计算机一段程序和一个input，计算机无法计算该程序能不能计算该input。
- 上帝能否制造一台能预言的机器：它具有红、绿灯，当问它的问题正确时，绿灯亮；不正确时，红灯亮。

1.3 易解问题与难解问题

多项式时间

- 一个算法时间复杂度是$O(n^3)$，它完全是可以接受的。因为它在一个非常大的输入下，仍然能在一个合理的时间内得到结果。
- 这里研究那些还没有找到合理快速算法的问题，但也无人证明该问题不存在快速算法。
- 那么问题的难易程度该如何定义呢？

1.3 易解问题与难解问题

多项式时间

- 我们采用多项式时间界限来定义一个问题的难易程度。
- 当我们判定一个问题是否可以找到一个多项式时间界限的解法，有以下三种情况：
  - **Yes**（找到多项式时间界限的解法）
  - **No**（证明不存在多项式时间界限的解法）
    - 证明该问题的所有解法是指数时间的
    - 证明根本不存在多项式时间界限的解法去解决该问题。
  - **Not sure**（目前没找到，不代表没有，也就是后面的NPC问题）

1.3 易解问题与难解问题

问题的易处理性（Tractablility）

- 一个问题可以在多项式时间内解决，称该问题是易解决的。
  - $\Theta(n^{100})$也很难处理，但是需要如此高次的多项式时间问题非常少；
  - 多项式时间具有很好的封闭性，无论加乘，多项式依然是多项式。
- 一个问题不能在多项式时间内解决，称该问题是不易解决的。

问题的复杂性分类小结

■图灵机模型：是人们普遍接受的计算理论的标准模型

◆它证明了通用计算理论，肯定了计算机实现的可能性，同时它给出了计算机应有的主要架构；

◆图灵机模型引入了读写、控制器和存储器等的概念，极大的突破了过去的计算机器的设计理念；

◆图灵机模型理论是计算学科最核心的理论，因为计算机的极限计算能力就是通用图灵机的计算能力，很多问题可以转化到图灵机这个简单的模型来考虑。

■可计算问题与不可计算问题

◆ 可计算（Computability）是研究什么样的问题可以被计算机系统或其他机器解决或计算的数学理论，它由图灵（Alan Turing）等人于1930年代提出。可计算是计算机科学、逻辑学、哲学等领域的基础，它揭示了计算的本质和限制，为计算机系统的设计和分析提供了理论指导和工具。

■易解问题与难解问题：是否能在多项式时间内求解的可计算问题。

PART 02

P类问题与NP类问题

2.1 判定问题

- 判定问题（decision problems）： 输出为“yes” or “no” (or, more formally, “1” or “0”)。
- 判断是否有一种能够解决某一类问题的能行算法的研究课题；
- 可以将搜索问题，生成问题，优化问题等转换成一个相关的判定问题。

2.1 判定问题

哈密顿回路-HC

- 判定问题：任给无向图G，问G是哈密顿图吗？
- 对应的搜索问题：
  - 为G找出一条哈密尔顿回路(如果存在)
  - 回答“No”(如果不存在)
- 对应的搜索问题不比判定问题更容易

2.1 判定问题

哈密顿回路-HC

- 如果多项式时间算法***A***能求解搜索问题
  - ◆当***G***是哈密尔顿回路时，***A***输出***G***的一条哈密尔顿回路；
  - ◆当***G***不是哈密尔顿回路时， ***A***输出“No”。
- 可基于***A***构造判定算法***B***
  - ◆如果***A***对***G***输出的是回路， ***B***输出“Yes”；
  - ◆如果***A***输出“No” ， ***B***输出“No”；
  - ◆则显然，***B***也是多项式时间算法。
- 于是HC搜索问题是易解的⇒HC判断问题是易解的；反之，HC判定问题是难解的⇒其搜索问题是难解的。

2.1 判定问题

旅行商问题-TSP

■优化问题：

任给n个城市，以及城市i与城市j之间的正整数距离$d(i, j)$，$i \neq j, 1 \leq i, j \leq n$。要求给出一条长度最短的巡回路线。

■判定问题：

给定一个带权图G和一个正数D，问是否有一条每一个城市恰好经过一次最后回到出发点且长度不超过D的巡回路线吗？即，是否存在$1, 2, \ldots, n$的排列σ，使得：

$$\sum_{i=1}^{n-1} d(\sigma(i), \sigma(i+1)) + d(\sigma(n), \sigma(1)) \leq D$$

2.1 判定问题

判定问题与优化问题

- 如果优化问题是易解问题，那么它相关的判定问题也是易解问题。
- 如果判定问题是难解问题，那么优化问题一定是难解问题；
- 将优化问题规约为判定问题，通过判定问题判断优化问题的下界。

2.2 P类问题

■**P(Polynomial)类问题：**是**确定性计算模型**下的**易解**的**判定**问题类，即在确定图灵机模型下多项式时间可解的判定问题类。

■**确定性（determinism）算法：**算法的整个执行过程中，每一步都只有一个确定的选择，并且对于同一输入实例运行算法，所得的结果严格一致。

2.3 NP类问题

■NP （Nondeterministic Polynomial，非确定性多项式时间算法）：在非确定图灵机模型下多项式时间内可解的判定问题，而在确定性图灵机下可验证的问题类。

◆所有多项式时间可验证的判定问题；

◆ 有可能多项式时间可解（P），也有可能多项式时间不可解。所以它是多项式时间非确定性算法可解。

2.3 NP类问题

非确定性图灵机

■非确定性图灵机(NDTM)：一个k带的非确定性图灵机M是一个7元组：(Q，T，I，δ，b，q_0，q_f)。与确定性图灵机不同的是：非确定性图灵机允许状态转移函数δ具有不确定性，即对于$Q \times T^k$中的每一个值$(q; x_1, x_2, \ldots, x_k)$，当它属于δ的定义域时，$Q \times (T \times \{L，R，S\})^k$中有唯一的一个子集$\delta(q; x_1, x_2, \ldots, x_k)$与之对应。可以在$\delta(q; x_1, x_2, \ldots, x_k)$中随意选定一个值作为它的函数值。

2.3 NP类问题

非确定性算法

■非确定性（nondeterminism）算法：由猜测和验证构成的算法。

◆猜测阶段：对问题的输入实例产生一个任意字符串ω，对于算法的每一次运行，串ω的值可能不同，因此，猜测以一种非确定的形式工作。

◆验证阶段：用一个确定性算法验证两件事。首先，检测猜测阶段产生的字符串ω是否合理；再次，验证串ω是否是问题的解。

2.3 NP类问题

非确定性算法示例

■哈密顿回路：一个图是否包含哈密顿会路。

◆猜测：一个包含所有点的序列p

◆验证：验证p是否为哈密尔顿回路，返回yes或者no。

2.4 P问题与NP问题的关系

- 显然所有P问题一定是NP问题，即$P \subseteq NP$ 。
  - 由于一台确定性图灵机可看作是非确定性图灵机的特例，所以可在多项式时间内被确定性图灵机接受的语言也可在多项式时间内被非确定性图灵机接受。故$P \subseteq NP$;
  - 在多项式时间内可解，那么一定在多项式时间内可验证。
- 那么所有NP问题都是P问题吗？
  - 目前没有人证明P=NP，这依然是一个待解决的问题。

P类问题与NP类问题小结

■判定问题：表现为寻求一种能行的方法、一种机械的程序或者算法，从而能够对某类问题中的任何一个在有穷步骤内确定是否具有某一特定的性质。选择判断问题，从而借助于图灵可判定（Turing Dicidable）理论分析问题复杂度。

■P类问题：所有可以在多项式时间内求解的判定问题构成P类问题。

■NP类问题：给定一个证书（certificate）也可以理解为一个解或结果，可以在多项式时间内验证此证书是否是问题的一个解的问题。比如，在哈密顿回路中，我们给出一个所有节点的序列，即证书，可以很容易在多项式时间内验证这个证书是否是一个哈密顿回路。所有的非确定性多项式时间可解的判定问题构成NP类问题。

PART 03

NP-hard问题与NPC问题

3.1 问题的归约

■规约(Reductions)：即问题变换。一个问题A可以规约为问题B的含义是，可以用问题B的解法解决问题A（或者说，问题A可以“变成”问题B）。

■例1：求解一元一次方程（问题A）可规约为求解一元二次方程（问题B）：

- ◆一元二次方程的二次项系数为0；
- ◆用求解一元二次方程解法来解一元一次方程。

■例2：哈密顿回路可以规约为TSP问题（旅行商问题）：

- ◆ TSP是带权图中求最小权的哈密顿回路。

3.1 问题的归约

■定义（归约）. 一个问题A可以归约为另外一个问题B，需要满足以下两个条件：

◆实例对应性： A的任意一个实例φ，通过函数f都可以转化成B的一个实例f(φ),且这个转化函数f必须为多项式时间；

◆输出一致性：归约后的输出和原来的输出一致，即在相同的输入下， A的输出结果和B的输出结果一致。

■问题A可归约为问题B，记为 $A \leq_P B$。

3.1 问题的归约

■规约的步骤：

（1）输入转换：将问题A的输入I_A转换为问题B的输入I_B；

（2）问题求解：对问题B应用它的一个算法产生一个输出O_B；

（3）输出转换：将问题B的输出O_B转换为问题A对应于输入I_A的正确输出O_A。

3.1 问题的归约

■规约的性质：

◆归约具有传递性：如果问题A可归约为问题B，问题B可归约为问题C，则问题A一定可归约为问题C。

◆假设$A \leq_p B$，则表示：A大于B的难度不会超过一个多项式时间因子：

- 如果B能够在多项式时间内求解，则A也能在多项式时间内求解，即：若果$B \in P$，则$A \in P$；
- 如果A不能在多项式时间内求解，则B也不能在多项式时间内求解；即如果$A \in NPC$，则 $B \in NPC$；
- 假设$A \leq_p B$且$B \leq_p A$，那么A和B是等价的。

3.1 问题的归约

■规约的证明：

◆实例对应性：

（1）A问题的任意一个实例都可以转化为B问题的一个实例；

（2）这个转化过程是多项式时间可以完成的，或者说转化函数f是个多项式函数。

◆输出一致性：

（1）A问题的任意一个实例，如果其输出为真，则对应B问题实例的输出也一定为真；

（2）B问题的任意一个实例，如果其输出为真，则对应A问题实例的输出也一定为真。

3.1 问题的归约

■规约证明实例：

- ◆Dantzig［1960］把一些组合最优化问题归约为一般的0-1整数线性规划问题；
- ◆Edmonds［1962］把图论问题“用最少的顶点覆盖所有边”和“寻找最大的顶点独立集”归约为一般的“集合覆盖问题”；
- ◆Gimple［1965］把一般的集合覆盖问题归约为逻辑设计的“素蕴涵覆盖问题”；
- ◆Dantzig，Blattner和Rao[1966]描述了一个“著名的”归约，把巡回推销员问题归约为带非负边长的“最短路径问题”。

3.2 NP-hard与NPC问题

■**NP-hard：** 所有NP问题都可以多项式归约到问题D，则D问题称为NP-hard问题。

■**NPC（NP-completeness）：** 所有NP问题都可以多项式归约到NP问题D，则NP问题D称为NPC问题。

■**NPC问题D满足两点性质：**

- ◆D问题是NP-hard问题；
- ◆D问题是NP问题。

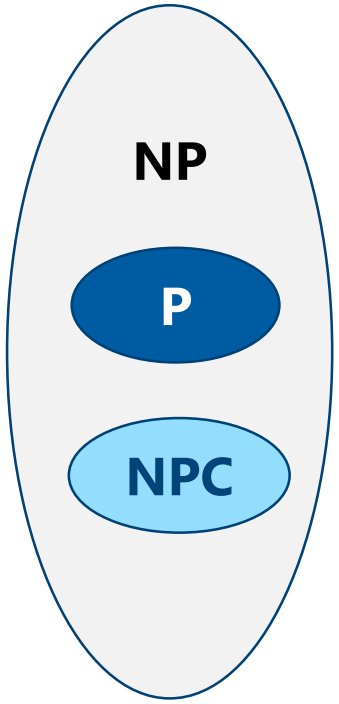
3.2 NP-hard与NPC问题

■NPC问题含义：

◆它和NP中任一问题一样难。也就是说如果任何问题都能通过多项式时间算法转换（规约）为这个NP问题，那么这个问题就是NP完全问题。也是最难的问题。

◆称之为完全，是因为NP完全问题归约了所有的NP问题。任意一个NP完全问题如能够在多项式时间内解决，那么NP中的每个一个问题都可用多项式时间解决，即 P=NP。

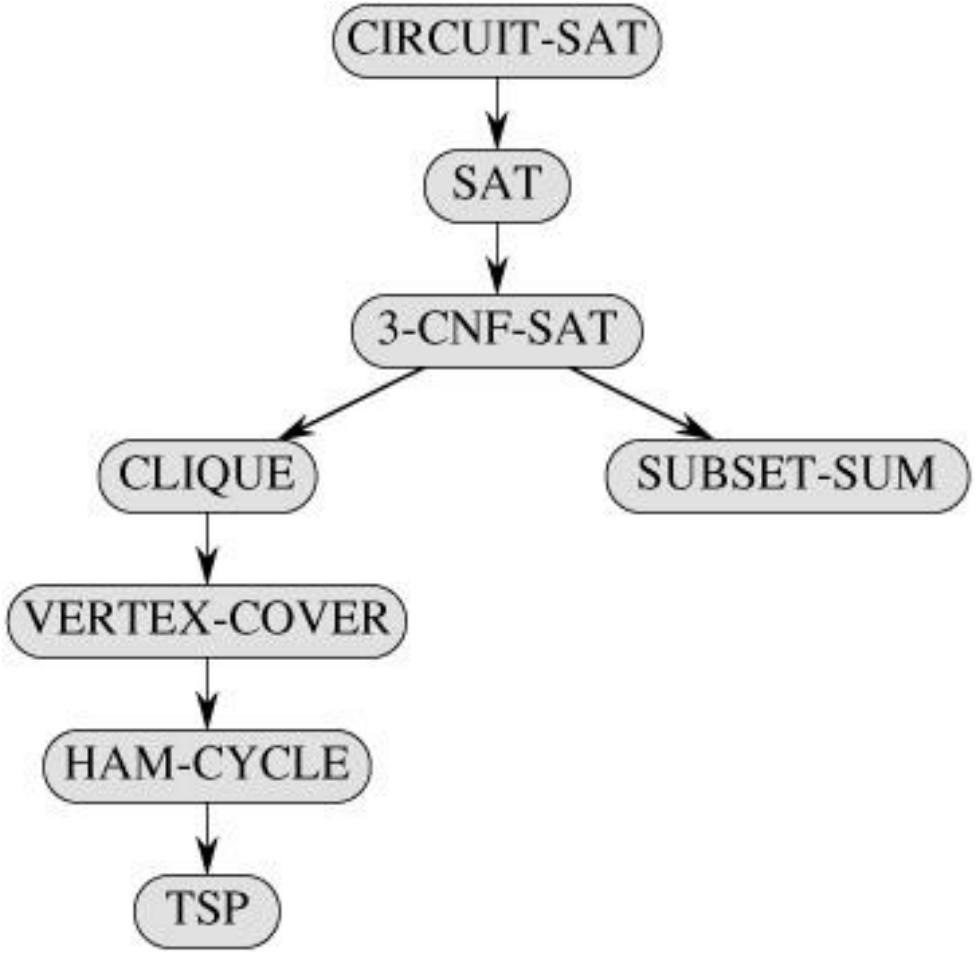
◆原因：NP完全问题实际上是最复杂的NP问题，即所有的NP问题都可以归为为一个NP完全问题，所以解决一个就解决所有。



3.2 NP-hard与NPC问题

■典型的NPC问题

- ◆团问题
- ◆顶点覆盖问题
- ◆哈密顿回路问题
- ◆旅行商问题
- ◆子集和问题



3.2 NP-hard与NPC问题

■ NPC问题的证明：

◆是一个NP问题；

◆一个已知的NPC问题可以规约成该问题。

NP-hard问题与NPC问题小结

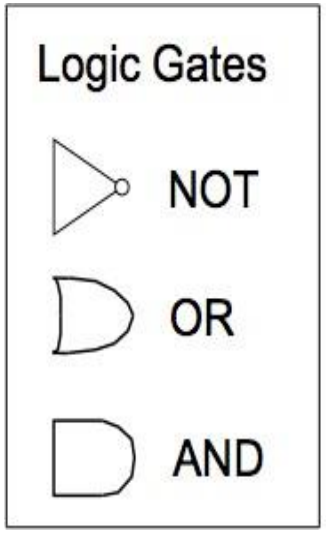
- **归约性**：通俗的讲，一个问题（如Q1）可以规约为另外一个问题（如Q2）是指问题Q1可以转换为问题Q2，之后可以通过求解Q2的方法来求解Q1。
- **NPC问题（NP完全问题）**：是NP类问题中最难的问题，包含两个条件：
  1. 是一个NP问题（首先限定了一个问题的难度范围，不能太难，至少可验证解）
  2. 所有的NP问题都可以‘转换’成此问题（确切的定义是：所有的NP问题都可以归约（reducibility）成此问题）
- **NP Hard问题**：相较于NPC问题，它没有要求必须是一个NP问题这个条件，也就是意味着其甚至在多项时间解都不可验证。

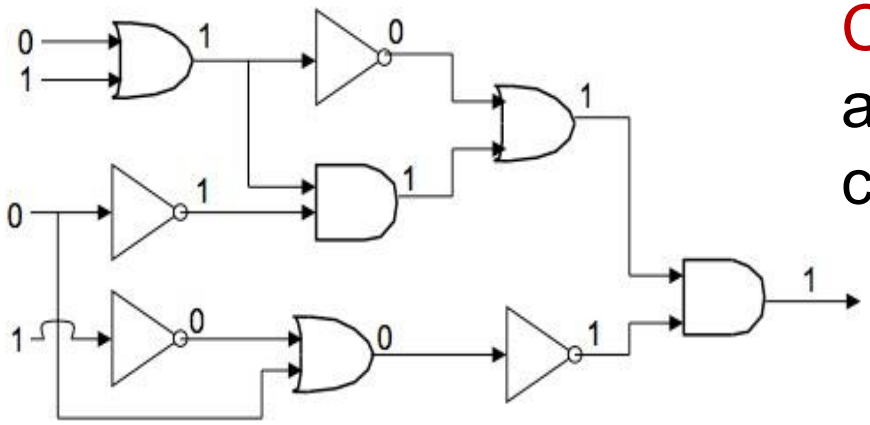
PART 04

几个典型NP完全问题证明

4.1 第一个NPC问题

- **电路可满足性问题（circuit-satisfiability problem）：** 给定一个逻辑电路，问是否存在一种输入使输出为True。
- 其它的NPC问题都是由这个问题归约而来的。因此，逻辑电路问题是NPC类问题的“鼻祖”。
- 有了第一个NPC问题后，一大堆NPC问题就出现了，因为再证明一个新的NPC问题只需要将一个已知的NPC问题约化到它就行了。





Circuit_SAT={<c>: c is a satisfiable boolean combinational circuit}

4.2 布尔公式可满足性问题

■布尔公式（boolean formula）φ包含：

- ◆ n个布尔变量$x_1, x_2, ..., x_n$;
- ◆ m个布尔连接词：布尔连接词是任何具有一个或两个输入和一个输出的布尔函数，如与、或、非、蕴含、当且仅当；
- ◆括号。

■布尔公式可满足性问题（亦称为命题可满足性问题，缩写为SAT），是确定是否存在满足给定布尔公式取值为TRUE的各变量的一组取值：

$$SAT = \{ \langle f \rangle, f \text{ is a satisfiable boolean formula} \}.$$

4.2 布尔公式可满足性问题

示例：

布尔公式：

$$\varphi = ((x_1 \rightarrow x_2) \vee \neg((\neg x_1 \leftrightarrow x_3) \vee x_4)) \wedge \neg x_2$$

具有可满足性取值：$\langle x_1 = 0, x_2 = 0, x_3 = 1, x_4 = 1\rangle$，因为：

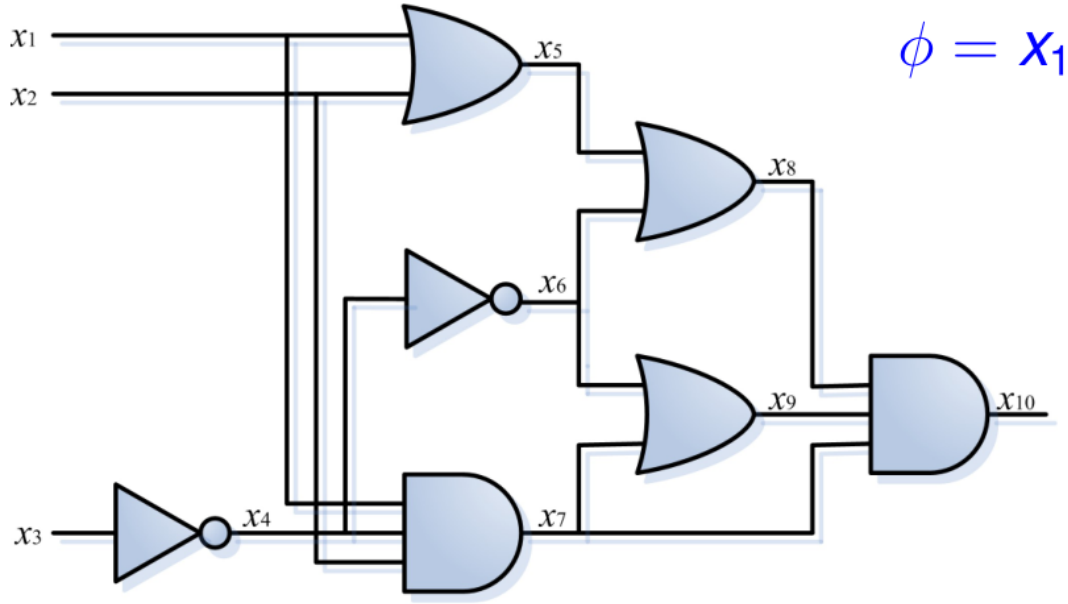
$$\begin{aligned}\varphi &= ((0 \rightarrow 0) \vee \neg((\neg 0 \leftrightarrow 1) \vee 1)) \wedge \neg 0 \\ &= (1 \vee \neg(1 \vee 1)) \wedge 1 \\ &= (1 \vee 0) \wedge 1 = 1\end{aligned}$$

4.2 布尔公式可满足性问题

■布尔公式可满足性问题是NPC问题，证明：

(1) 首先需证明SAT∈NP，为此，只要证明SAT的任意实例φ，φ的可满足指派组成的证据可以在多项式时间内验证。

（2）规约证明：电路可满足性规约到公式可满足性。



$$\phi = x_{10} \wedge (x_4 \leftrightarrow \neg x_3) \wedge (x_5 \leftrightarrow (x_1 \vee x_2)) \wedge (x_6 \leftrightarrow \neg x_4) \wedge (x_7 \leftrightarrow (x_1 \wedge x_2 \wedge x_4)) \wedge (x_8 \leftrightarrow (x_5 \vee x_6)) \wedge (x_9 \leftrightarrow (x_6 \vee x_7)) \wedge (x_{10} \leftrightarrow (x_7 \wedge x_8 \wedge x_9))$$

CIRCUIT-SAT $\leq_p$ SAT

4.3 3-CNF可满足性

- **3-合取范式（3-CNF）：**如果一个布尔公式可以表示为所有子句对“与”，且每个子句都是一个或多个文字的“或”，则为CNF；如果限制每个子句均有三个不同的文字，则为3-CNF。
- 示例：

$$(x_1 \vee \neg x_1 \vee \neg x_2) \wedge (x_3 \vee x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$$

4.3 3-CNF可满足性

■**3-CNF-SAT：3合取范式形式的布尔公式的可满足性问题是NP完全的**

■证明：

（1）这是一个NP问题

（2）公式可满足性可以归约到3-CNF，即$SAT \leq_p 3\text{-}CNF\text{-}SAT$

2.1 将布尔公式转换为子句的合取式

2.2 将子句转换为合取范式

2.3 将子句转为3个文字的合取范式

4.3 3-CNF可满足性

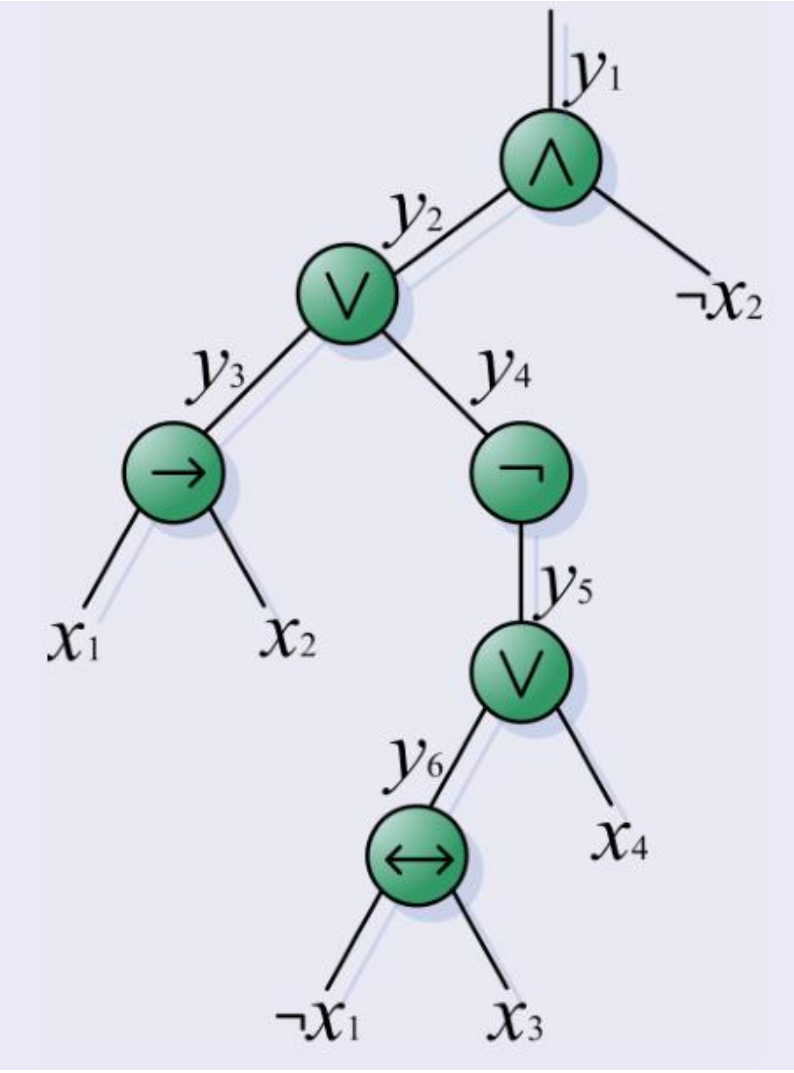
证明：

1. 将布尔公式转换为子句的合取式：

1.1 建立布尔公式的语法树；

示例：

$\varphi = ((x_1 \rightarrow x_2) \vee \neg((\neg x_1 \leftrightarrow x_3) \vee x_4)) \wedge \neg x_2$.

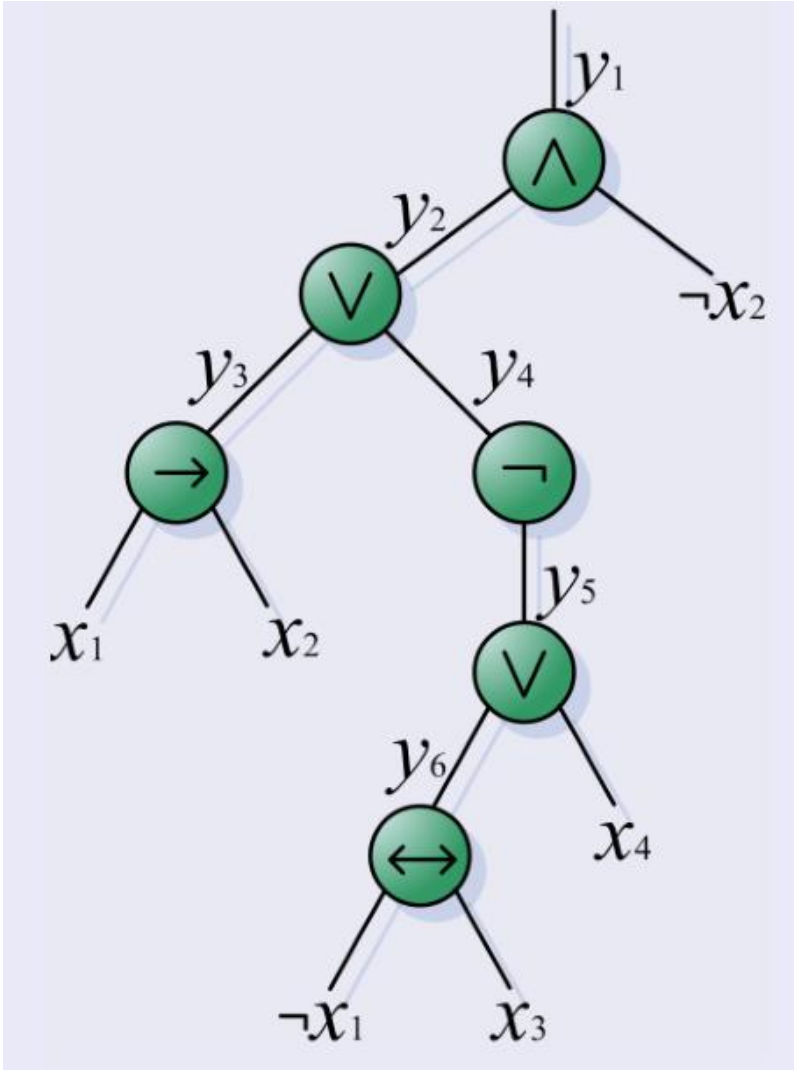


4.3 3-CNF可满足性

证明：

1.2 将语法分析树看成电路，得出归约的布尔公式，此布尔公式为合取式。

$$
\begin{aligned}
\phi' = y_1 &\wedge (y_1 \leftrightarrow (y_2 \wedge \neg x_2)) \\
&\wedge (y_2 \leftrightarrow (y_3 \vee y_4)) \\
&\wedge (y_3 \leftrightarrow (x_1 \rightarrow x_2)) \\
&\wedge (y_4 \leftrightarrow \neg x_5) \\
&\wedge (y_5 \leftrightarrow (y_6 \vee x_4)) \\
&\wedge (y_6 \leftrightarrow (\neg x_1 \leftrightarrow x_3))
\end{aligned}
$$



4.3 3-CNF可满足性

证明：

2. 将子句转换为合取范式

2.1 构造每个子句的真值表;

2.2 根据真值表中值为0的项，得出析取范式, 此析取范式等价于子句的否;

2.3 运用德摩根定律，得出合取子句（将析取范式再取否）。

$$\neg\phi_i' = (y_1 \wedge y_2 \wedge x_2) \vee (y_1 \wedge \neg y_2 \wedge x_2) \vee (y_1 \wedge \neg y_2 \wedge \neg x_2) \vee (\neg y_1 \wedge y_2 \wedge \neg x_2).$$

$$\phi_1'' = (\neg y_1 \vee \neg y_2 \vee \neg x_2) \wedge (\neg y_1 \vee y_2 \vee \neg x_2) \wedge (\neg y_1 \vee y_2 \vee x_2) \wedge (y_1 \vee \neg y_2 \vee x_2)$$

示例： $(y_1 \leftrightarrow (y_2 \wedge \neg x_2))$.

y_1	y_2	x_2	$(y_1 \leftrightarrow (y_2 \wedge \neg x_2))$
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	0
0	1	1	1
0	1	0	0
0	0	1	1
0	0	0	1

4.3 3-CNF可满足性

证明分析：

3. 将子句转为3个文字的合取取式

3.1 如果C_i中有3个不同的文字，则直接把C_i作为3合取范式中的一个子句；

3.2 如果C_i中有2个不同的文字 l_1 and l_2，则将其转换为：$(l_1 \vee l_2 \vee p) \wedge (l_1 \vee l_2 \vee \neg p)$

3.3 如果C_i中有1个不同的文字 l，则将其转换为：$(l \vee p \vee q) \wedge (l \vee p \vee \neg q) \wedge (l \vee \neg p \vee q) \wedge (l \vee \neg p \vee \neg q)$。

4.3 3-CNF可满足性

证明：

- 以上映射为多项式时间
  - ◆将布尔公式转换为子句的合取式
    同布尔电路转换为布尔公式
  - ◆将子句转换为合取范式
    每个子句至多变为8个子句（至多3个变量）
  - ◆将子句转为3个文字的合取取式
    至多引入4个子句
- 输出一致性：由以上步骤可知，3-CNF是可满足的当且仅当以上三个步骤的每一步都是可满足的。

=》以上转换为归约。

4.4 团问题

- **团(clique)：** 给定无向图G=(V, E)，团是G中的一个顶点子集V’ ⊆V，其中每一对顶点间都由E中的一条边连接。
- 一个团是G的一个完全子图。
- **团的规模：** 是指它所包含的顶点数。
- **团问题：** 是关于寻找图中规模最大的团的最优化问题。
- **团的判定问题：** 在图中是否存在一个给定规模为k的团？

CLIQUE={<G, k>: G是一个包含规模为k的团的图}

4.4 团问题

■团问题是NP完全的

■证明方法：

（1）这是一个NP问题：

假设团中的顶点集$V' \subseteq V$，对于任意一对顶点u，$v \in V'$，通过检查边(u, v)是否属于E，就可以在多项式内确定V'是否为团。

（2）可以从其他NPC问题规约得到：

3合取范式可以归约为团问题：

3-CNF-SAT $\leq_p$ CLIQUE

4.4 团问题

3合取范式归约为团问题证明：

■设 $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_k$是3-CNF形式中一个具有k个子句的布尔公式，对r = 1, 2, ..., k，每个子句恰好有3个不同的文字。

构造图

■φ中的每个子句$C_r=(l'_1, l'_2, l'_3)$，对应图中3个顶点；

■对于任意两个在不同组的顶点，如果满足“这两个顶点不是‘否’的关系”这一条件，就用一条边连接。

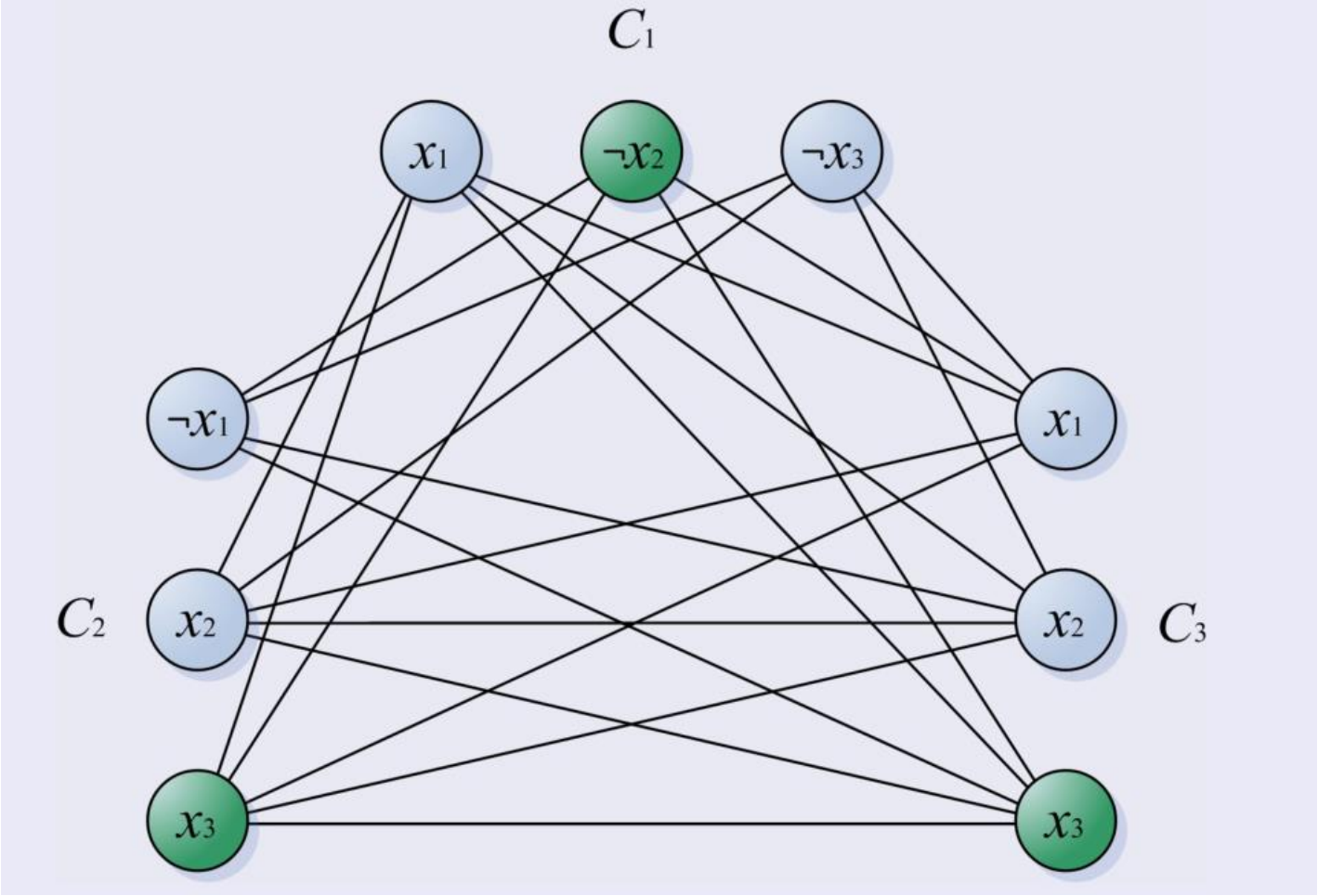
4.4 团问题

3合取范式归约为团问题证明：

■示例：

$\varphi =$

$(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$



4.4 团问题

3合取范式归约为团问题证明：

- 令φ是可满足的，即它有一个可满足的真值指派。 φ中每个子句至少有一个文字被指派1。每个这样的文字对应有一个顶点。从每个子句中选出一个这样的文字所对应的顶点形成一个规模为k的顶点子集，由G的构造，V是G中规模为k的团。
- (2)设G有一个规模为k的团。由于每个子句所对应的三个顶点彼此间无边连结。因此k个顶点分别来自k个子句所对应的三元组。因此这k个顶点对应的文字是彼此相容的，对这些文字赋值1，容易得到的一组可满足赋值。
- =》3合取范式和团问题具有输出一致性。

4.4 团问题

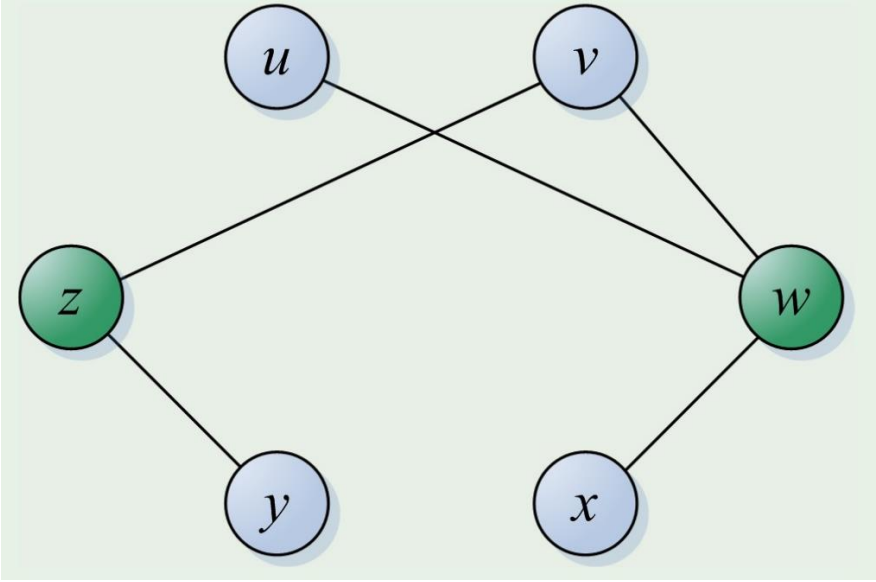
3合取范式归约为团问题证明：

- 另，由3-合取范式构建给定图可以在多项式时间完成，因而团问题是NPC问题。
- 疑问：归约为一个特殊的图，能说明 一般图的团问题也是NP完全的吗？
- 事实上仅证明了CLIQUE在这种特定图的情况下才是NP难的。但是，这一证明足以证明在一般的图中，CLIQUE也是NP难的。因为：如果有一个能在一般的图上解决CLIQUE问题的多项式时间算法，那么它就能在受限的图上解决CLIQUE问题。

4.5 顶点覆盖问题

- 顶点覆盖问题: 是指在一个给定的图中，找出具有最小规模的顶点覆盖。
- 顶点覆盖的判定问题：把这一最优化问题重新表述为一个判定问题，即确定一个图是否具有一个给定规模k的顶点覆盖。作为一种语言，定义为：

VERTEX-COVER={<G, k>: 图G有一个规模为k的顶点覆盖}.



规模为2的顶点覆盖{w, z}

4.5 顶点覆盖问题

■顶点覆盖问题是NP完全的

■证明方法：

（1）这是一个NP问题：

假定已知一个图G=(V, E)和整数k，将顶点覆盖V’⊆V作为证书。验证算法可证实|V’|=k；然后对每条边(u,v)∈E,检查是否有u∈V’或v∈V’。这一问题很容易在多项式时间内验证。

（2）可以从其他NPC问题规约得到：

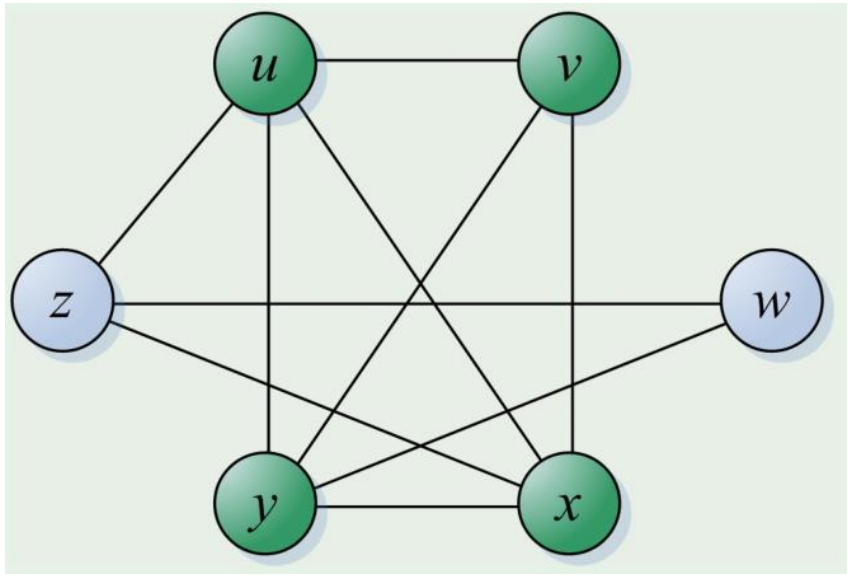
团问题可以归约为顶点覆盖问题：

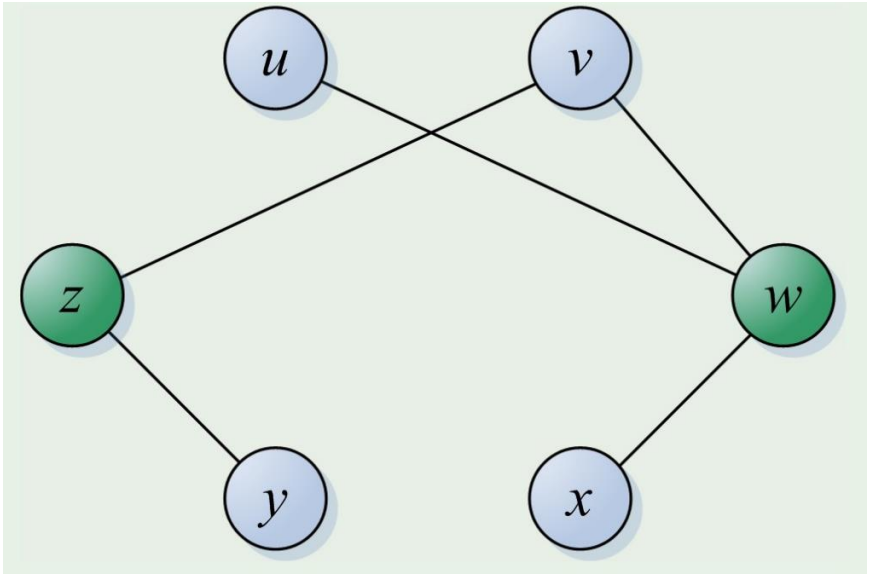
$\text{CLIQUE} \leq_P \text{VERTEX\_COVER}$

4.5 顶点覆盖问题

团问题归约为顶点覆盖问题的证明：

- 给定一无向图G = (V, E), 定义G的补图$\overline{G} = (V, \overline{E})$。其中：
  $\overline{E}$= {(u,v) : u, v∈V, and (u,v) ∉ E}。（补图+原图=完全图）
- 规约过程：图G具有一个规模为k的团，当且仅当$\overline{G}$有一个规模为|V|-k的顶点覆盖。





4.5 顶点覆盖问题

团问题归约为顶点覆盖问题的证明：

- 如果G有规模为k的团V′，则$\overline{G}$必然有规模为|V|-k的顶点覆盖：
  - 设(u, v)为$\overline{G}$的任意边，则此边肯定不属于图G，则(u, v)其中至少一个点不属于团V′，即(u, v)至少有一个属于V-V′， 即V-V′覆盖边(u, v)；
  - 因为边为任意的，所以V-V′可以覆盖所有边，即V-V′为规模为|V|-k 的顶点覆盖。

4.5 顶点覆盖问题

团问题归约为顶点覆盖问题的证明：

■如果$\overline{G}$有规模为|V|-k的顶点覆盖S′，则G必有规模为k的团：

◆对于(u, v)为$\overline{G}$的任意边，则u和v至少一个属于$\overline{G}$的顶点覆盖S′：

因为：如果有两个点都不属于S′，则这两个点在$\overline{G}$里是没有边，则在G里一定是有边的

◆即对于任意(u, v)，如果都不属于S′，则u, v之间在G图一定有边，所有这些顶点V-S′组成了规模为k的团。

■ =》团问题和顶点覆盖问题具有输出一致性。

■另，由原图构建补图可在多项式时间完成，因而顶点覆盖问题为NPC问题。

4.6 哈密顿回路问题

- 哈密顿回路问题：在一个图（通常为无向图）中，从某个顶点出发，经过所有其他顶点一次且仅一次的回路称为哈密顿回路。
- 其判定问题为：给定一个图，判断这个图是否包含哈密顿回路。

4.6 哈密顿回路问题

■哈密顿回路问题是NP完全的。

■这是一个NP问题

- 已知一个图G=(V,E),将形成哈密顿回路的|V|个顶点所组成的序列作为证书，容易在多项式时间内验证这个证书是否真是一个哈密顿回路。

■可以从其他NPC问题规约得到：

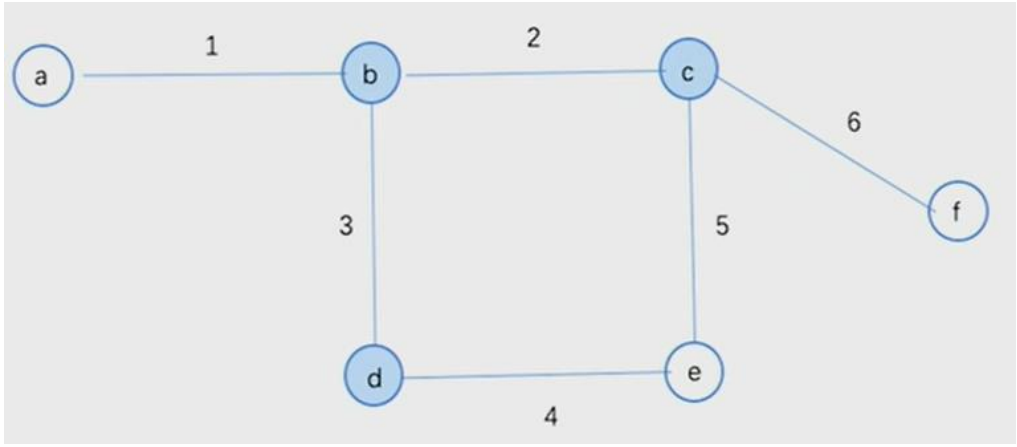
- 顶点覆盖问题可以归约为哈密顿回路问题

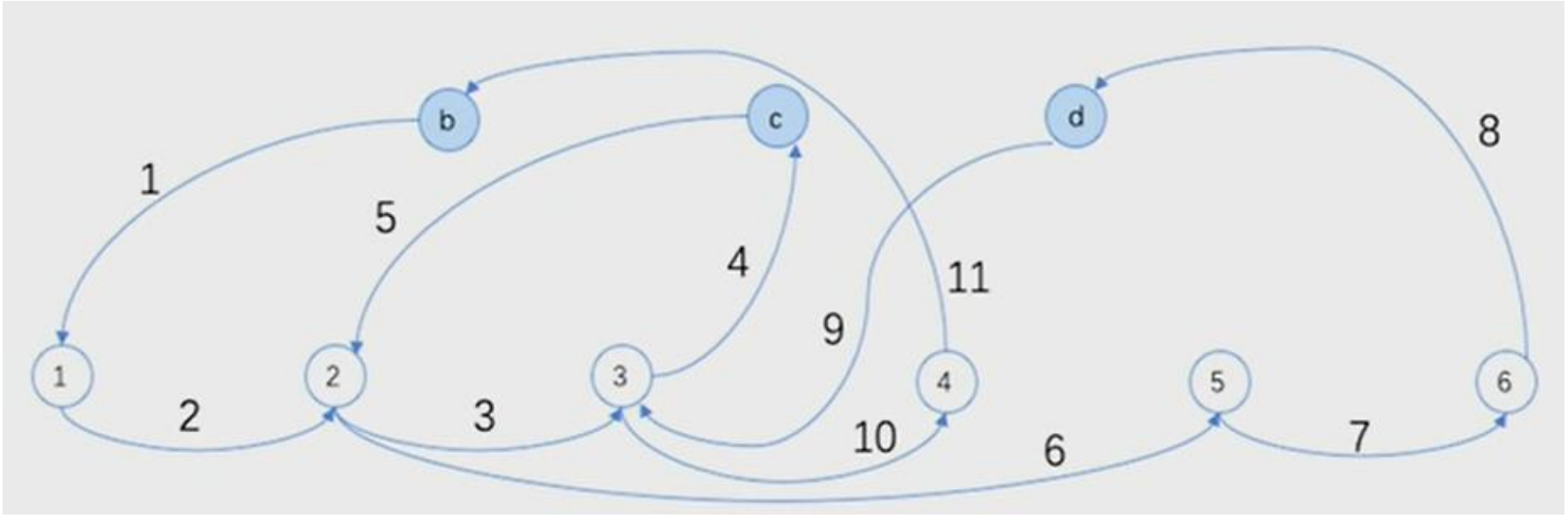
$$\text{VERTEX-COVER} \leq_p \text{HAM-CYCLE}$$

4.6 哈密顿回路问题

顶点覆盖问题归约为哈密顿回路问题的证明：

■顶点覆盖问题是覆盖一个图中所有的边，而哈密顿回路是需要遍历所有的点，所以从顶点覆盖规约到哈密顿回路应该是一个可行的思路。

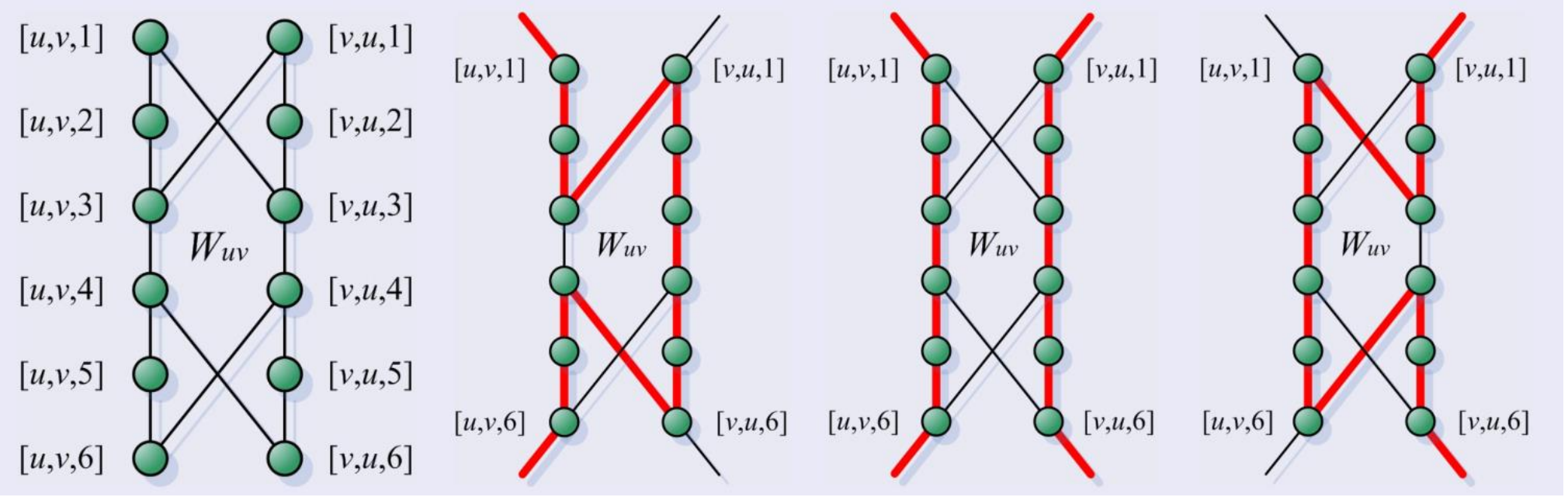




4.6 哈密顿回路问题

顶点覆盖问题归约为哈密顿回路问题的证明：

- 将给定无向图G=(V,E),和整数k，构建一个无向图G′ =(V′, E′)，G中有一个规模为k的顶点覆盖，当且仅当G′中有一条哈密顿回路。
- 附件图：给定原图中任意一条边(u, v), 转换为以下含有12个顶点和14条边的附件图。



4.6 哈密顿回路问题

顶点覆盖问题归约为哈密顿回路问题的证明：

■除了G中每条边对应的附件图，在图G′中再添加k个选择器顶点$s_1, s_2, ..., s_k$，这些选择器与G中顶点覆盖中的k个顶点相对应。

■再在图中添加两类边：

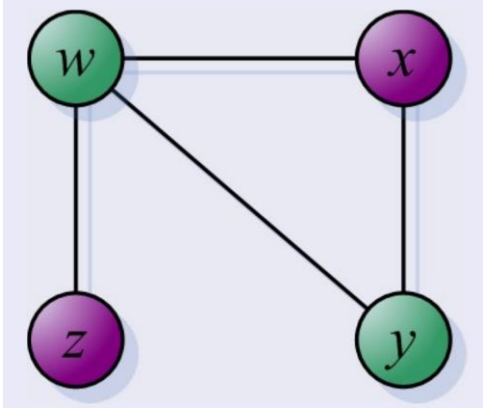
（1）针对图G中任一顶点$u \in V$，将与u关联的各条边对应的附件图，按照u的邻接顶点的任一顺序进行收尾相连；

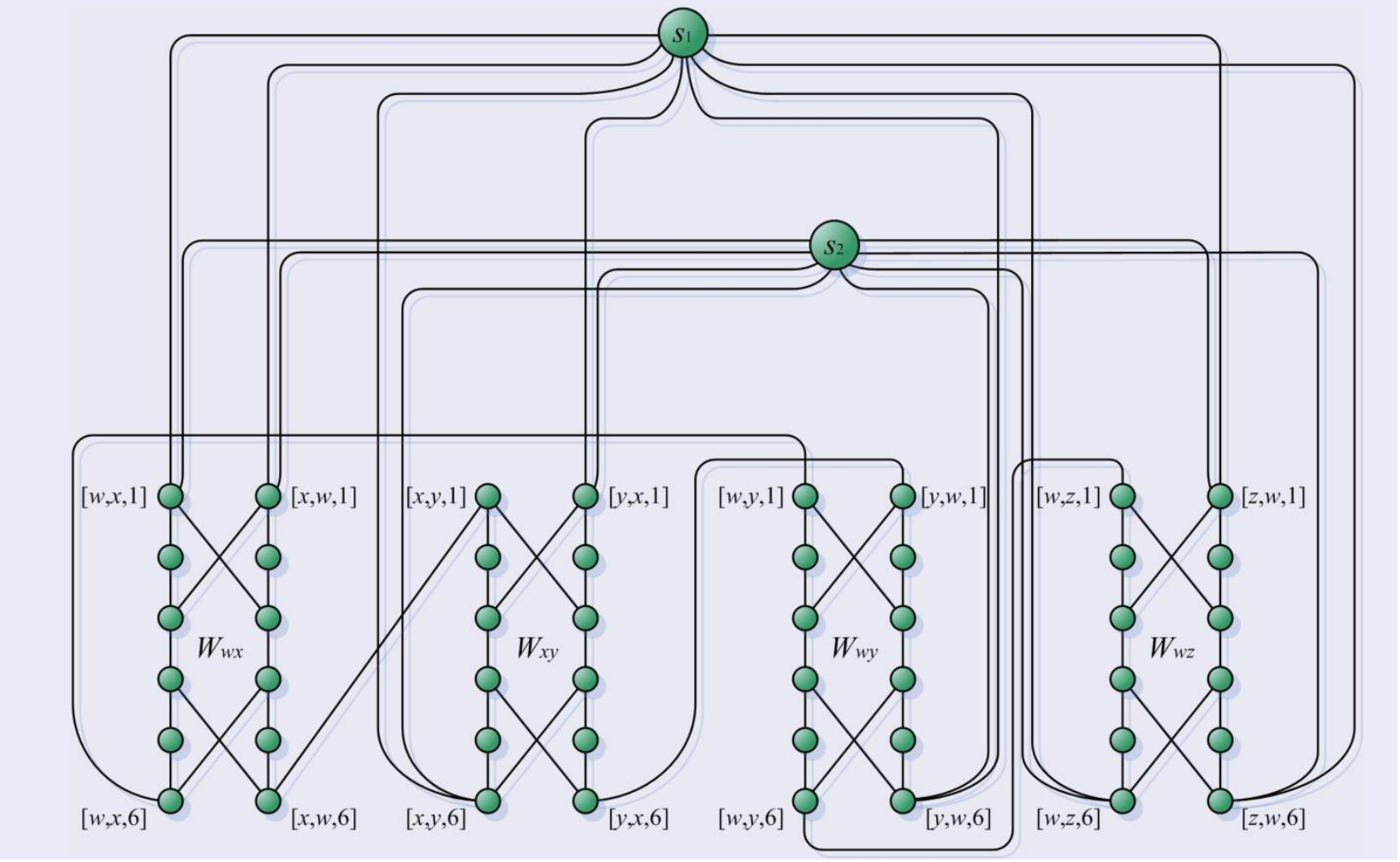
（2）针对图G中任一顶点u，将与其对应的第一个和最后一个附件图，分别链接到个G′中各选择器顶点上。

4.6 哈密顿回路问题

顶点覆盖问题归约为哈密顿回路问题的证明：

■顶点覆盖对应哈密顿回路问题中图G的构建示例





4.6 哈密顿回路问题

顶点覆盖问题归约为哈密顿回路问题的证明：

■可以在多项式时间基于G生成G′：

$$|V'| = 12|E| + k \leq 12|E| + |V|$$

$$\begin{aligned} |E'| &= (14|E|) + (2|E| - |V|) + (2k|V|) \\ &= 16|E| + (2k-1)|V| \\ &\leq 16|E| + (2|V| - 1)|V| \end{aligned}$$

4.6 哈密顿回路问题

顶点覆盖问题归约为哈密顿回路问题的证明：

■输出一致性：

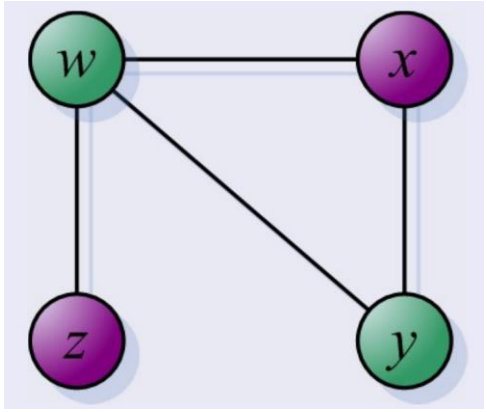
- ◆设G中有规模为k的顶点覆盖，将G′各选择器顶点与G中顶点覆盖中的顶点相对应，必在图G′中形成一条哈密顿回路。
- ◆若在图G′中有一条哈密顿回路，则回路中选择器顶点所对应的顶点集合，必形成图G中的一个顶点覆盖。

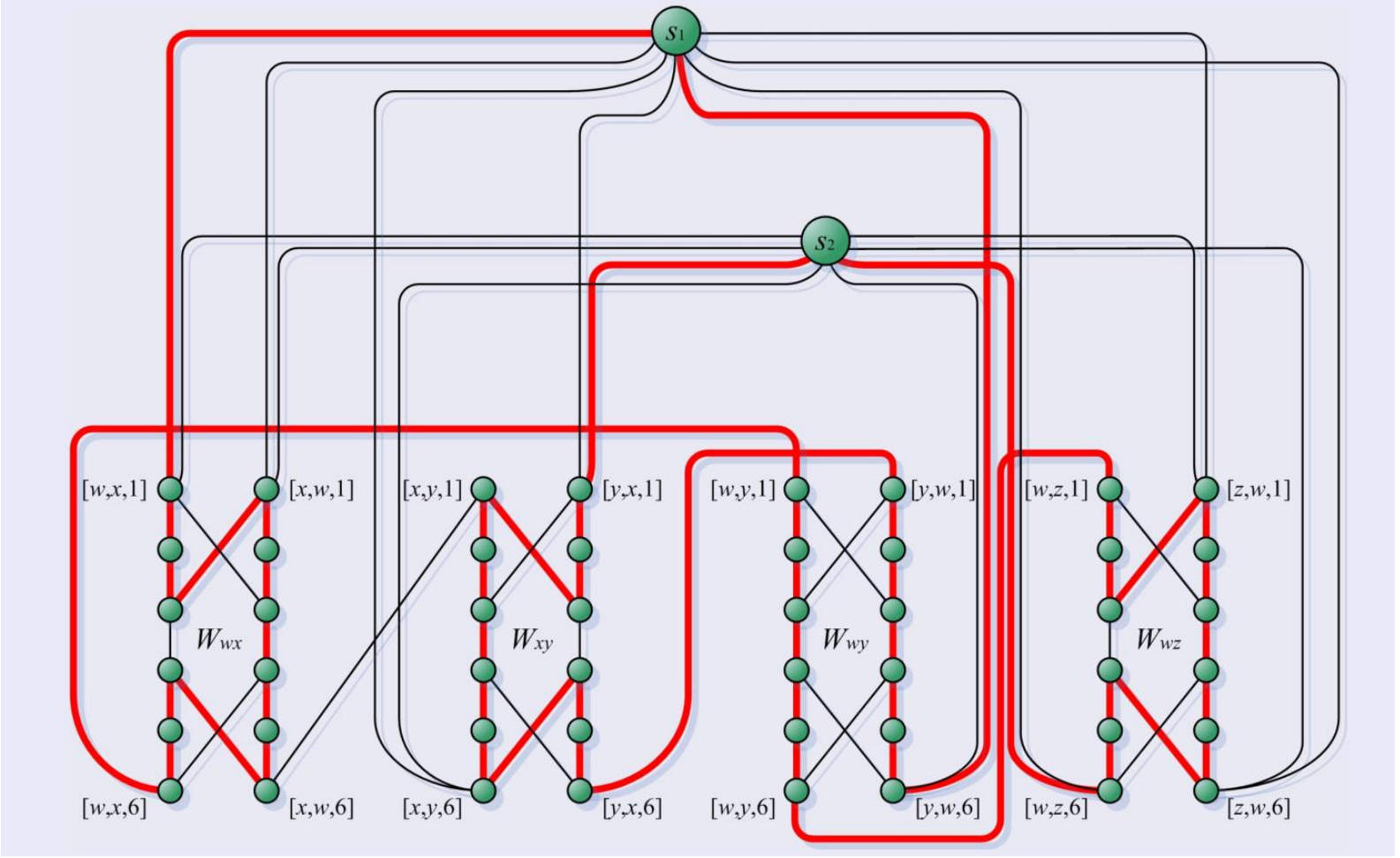
■ =》顶点覆盖问题是NP完全的。

4.6 哈密顿回路问题

顶点覆盖问题归约为哈密顿回路问题的证明：

■顶点覆盖与图G中哈密顿回路对应关系的示例





几个典型NP完全问题证明小结

- 第一个NPC问题：电路可满足性问题；
- 布尔公式可满足性问题：可由电路可满足性问题归约证明；
- 3-CNF可满足性问题：可由布尔公式可满足性问题归约证明；
- 团问题：可由3-CNF可满足性问题归约证明；
- 顶点覆盖问题：可由团问题归约证明；
- 哈密顿回路问题：可由顶点覆盖问题归约证明；
- 旅行商问题：可由哈密顿回路问题归约证明。

THANK YOU FOR YOUR WATCHING

Teacher：张明卫
@东北大学 软件学院