第七讲 计算复杂性理论上

本讲提要

- 算法时空复杂性分析
- 确定图灵机与RAM模型

1.1算法时空复杂性分析

问题由两部分组成：一是对问题的答案所满足的特征的说明；二是对其所有参数的描述。而一个问题的实例则可通过指定问题中所有参数的具体取值来得到。问题可用Π表示，实例可用I表示。

例子**1** 判定奇素数。给出一个奇整数$m>2$，判定m是素数否。

1.1算法时空复杂性分析(续)

定义1 所谓算法是有限规则的集合，这些规则可用来求解某一问题，通过带有一般性有步骤的计算，并在有限步骤后终止计算，产生输出。

算法特征

（1）有穷性
（2）确定性
（3）可行性
（4）输入
（5）输出

1.1算法时空复杂性分析(续)

例子2 判定奇素数算法。

（1）输入正奇数m，令$n=3$;

（2）以n除m，得到余数r;

（3）如果$r=0$，算法结束，输出m不是素数；否则执行第（4）步；

（4）如果n大于$\sqrt{m}$，输出m是奇素数，否则执行$n \leftarrow n+2$，然后执行第（2）步。

1.1算法时空复杂性分析(续)

定义2 如果某一问题的规模是n，解决问题的某一算法所需要的时间为关于n的函数$T(n)$，称为该算法的时间复杂性。

类似可定义空间复杂性。

定义3 Θ记号：对一个给定的函数$g(n)$，用$\Theta(g(n))$表示如下特定函数的集合：$\Theta(g(n))=\{f(n)$：存在正常数c_1，c_2，n_0，使对所有的$n\geq n_0$，有$0\leq c_1 g(n)\leq f(n)\leq c_2 g(n)\}$。

1.1算法时空复杂性分析(续)

例子3 证明：$2n^2-3n=\Theta(n^2)$，$2n^3\neq\Theta(n^2)$。

例子3证明.

对于第一个式子，首先确定常数c_1，c_2，和n_0，使得对于所有$n\geq n_0$，有

$$c_1n^2\leq 2n^2-3n\leq c_2n^2$$

成立。用n^2除上述不等式得

$$c_1\leq 2-3/n\leq c_2。$$

这样$c_1=1/2$，$c_2=2$，以及$n\geq 2$可使得$c_1n^2\leq 2n^2-3n\leq c_2n^2$成立。因此，$2n^2-3n=\Theta(n^2)$。

而定义3要求对于所有任意大的$n\geq n_0$和某个常数c_2，有$2n^3\leq c_2n^2$，即$2n\leq c_2$，对于常数c_2这不可能。

1.1算法时空复杂性分析(续)

定义**4** O记号：对一个给定的函数$g(n)$，用$O(g(n))$表示如下特定函数的集合：$O(g(n))=\{f(n)$：存在正常数c，n_0，使对所有的$n \geq n_0$，有$0 \leq f(n) \leq cg(n)\}$。

#O记号在一个常数因子内给出某函数的一个渐进上界。

定义**5** Ω记号：给定的函数$g(n)$，用$\Omega(g(n))$表示如下特定函数的集合：$\Omega(g(n))=\{f(n)$：存在正常数c，n_0，使对所有的$n \geq n_0$，有$f(n) \geq cg(n)\}$。

#Ω记号给出某函数的一个渐进下界。

1.1算法时空复杂性分析(续)

定理1 对任意函数$f(n)$和$g(n)$，$f(n)=\Theta(g(n))$当且仅当$f(n)=O(g(n))$和$f(n)=\Omega(g(n))$ 。

定义6 o记号：给定的函数$g(n)$，用$o(g(n))$表示如下特定函数的集合：$o(g(n))=\{f(n)$：对任意正常数c，存在n_0，使对所有的$n \geq n_0$，有$0 \leq f(n) \leq cg(n)\}$。

在o表示中，当n趋于无穷大时，应该有：

$$\lim_{n\to\infty}\frac{f(n)}{g(n)}=0。$$

1.1算法时空复杂性分析(续)

定义7 w记号：给定的函数$g(n)$，用$w(g(n))$表示如下特定函数的集合：$w(g(n))=\{f(n)$：对任意正常数c，存在n_0，使对所有的$n \geq n_0$，有$f(n) \geq cg(n)\}$。

在w表示中，当n趋于无穷大时，应该有：

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty。$$

1.1算法时空复杂性分析(续)

定 义 **8** 多 项 式 时 间 算 法 (polynomial time algorithm)是指时间复杂性函数是$O(p(n))$的算法，这里$p(n)$是关于n的多项式。不能够这样限定时间复杂度的算法被称为指数时间算法。

例子4 设ε和c是任意常数,$0<\varepsilon<1<c$。以下是一些函数按阶的渐进增长率由小到大的排列：

$$1<\ln\ln n<\ln n<e^{\sqrt{\ln n \ln\ln n}}<n^{\varepsilon}<n^{c}<n^{\ln n}<c^{n}<n^{n}<c^{c^{n}}。$$

1.1算法时空复杂性分析(续)

例子**5** 时间复杂度为$O(n\log n)$、$O(n^3)$的算法都是多项式时间算法，时间复杂度为$O(2^n)$、$O(n!)$、$O(n^{\log n})$的算法都是指数时间算法。若一台PC一天算力是10^{12}基本计算次，对$O(n^3)$的算法可解决问题规模是10^4，$O(2^n)$的算法就是40，$O(n!)$的算法就是14；那么PC的速度提高200倍，这三种算法解决的问题规模可达$6\cdot 10^4$、48、16。

1.2 确定图灵机与RAM模型

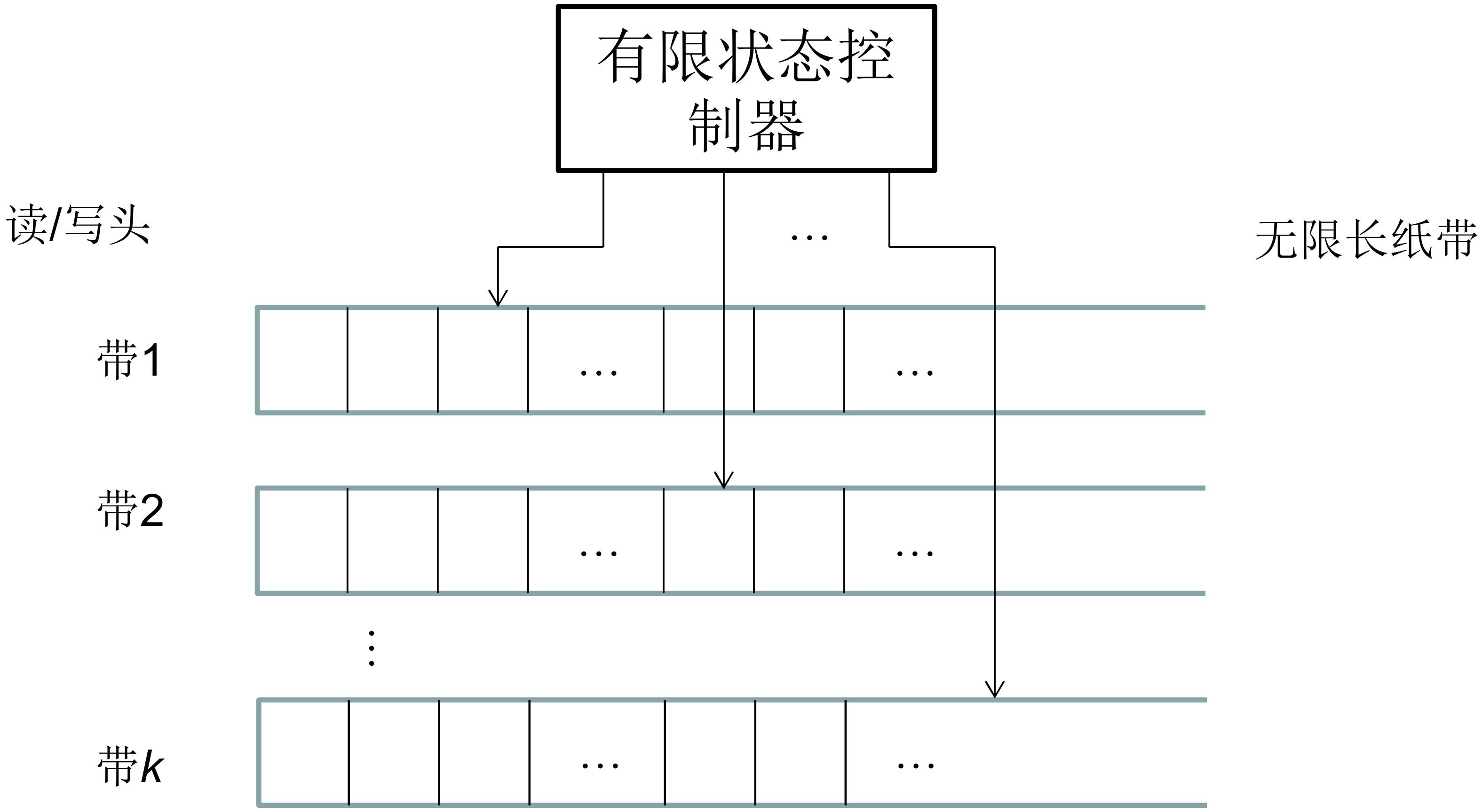

图 1 确定图灵机示意图

1.2 确定图灵机与RAM模型(续)

一个确定性k带图灵机程序(Program)可以详细描述为一个七元组：$TM=(Q, q_0, q_f, T, I, b, \delta)$,其中，$Q$是一个有限状态集合；$q_0 \in Q$称为初始状态；$q_f \in Q$称为终止状态或接收状态。$T$是带格中符号的有限集合；$I$是输入字符集，且$I \subset T$；$b$是$T$中的唯一空符，有$b \in T-I$；$\delta$称做图灵机的下移函数或有限状态控制函数，是从$Q \times T^k$的某一个子集到$Q \times (T \times \{L, R, S\})^k$的映射函数，即对由一个当前状态和$k$条带上扫描到的当前符号所构成的一个$k+1$元组，它唯一地给出一个新的状态和$k$个序偶，而每一个序偶由一个新的带符号和带头移动方向组成。

1.2 确定图灵机与RAM模型(续)

假定某台图灵机的下移函数表中有一个定义式为

$\delta(q_1, a_1, a_2, \ldots, a_k)=[q', (a'_1, d_1), (a'_2, d_2), \ldots, (a'_k, d_k)]$。

当图灵机处于状态q且对一切$1 \le i \le k$，第i条带的带头扫描当前方格中的符号正好是a_i时，图灵机就按这个下移函数定义式所规定的内容进行如下工作：

（1）把i条带头下当前方格中的符号a_i清除并写上新的带符号a'_i，$1 \le i \le k$。

（2）按d_i指出的方向移动各带的带头。这里，$d_i=L$表示带头往左移一格，$d_i=R$表示带头往右移一格，$d_i=S$表示带头不动。

（3）将图灵机的当前状态q改为q'。

1.2 确定图灵机与RAM模型(续)

定义9 被这台图灵机所接收的所有输入符号串的集合，称做这台图灵机识别的一个语言。

定义10 一台图灵机M的某一格局是一个k元组$(a_1, a_2, \ldots, a_k)$，其中a_i是一个形如xqy的字符串，这里q是M的当前状态，xy是在当前状态q下第i条带上的符号串(不计右端空白符)，第i个带头指向y的第一个符号，当y为空串时，第i个带头指向空白符。

1.2 确定图灵机与RAM模型(续)

如果图灵机计算一步后，它的格局由D_1变成了D_2，就记做$D_1 \to D_2$(读做“进入”)，对于$n \geq 2$，如果有

$D_1 \to D_2 \to \ldots \to D_n$

就记做

$D_1 \xrightarrow{*} D_n$

如果对某一k元组$(a_1, a_2, \ldots, a_k)$和某一输入串$(c_1, c_2, \ldots, c_t)$，$c_i \in I$，i=1，2，…，t，有

$(q_0\, c_1 c_2 \ldots c_t, q_0, \ldots, q_0) \xrightarrow{*} (a_1, a_2, \ldots, a_k)$

并且q_f在$(a_1, a_2, \ldots, a_k)$中，则称图灵机M接收输入串$c_1 c_2 \ldots c_n$。

1.2 确定图灵机与RAM模型(续)

定义11 一个确定图灵机程序的时间复杂性$T(n)$，是它处理所有长度为n的输入所需要的最大计算步数。

#当然，这是最坏情况下的时间复杂性。如果对某个长度为n的输入，图灵机不停机，则$T(n)$对这个n无定义。

定义12 一个确定图灵机程序的空间复杂性$S(n)$是处理所有长度为n的输入时，在k条带上所使用过的方格的总和。

#如果某个带头无限地向右移动而不停机，$S(n)$也无定义。

1.2 确定图灵机与RAM模型(续)

若存在一个多项式p，使得对所有$n \in Z^+$，有$T(n) \le p(n)$，则称程序为一个多项式时间确定图灵机程序。

定义**13**称满足如下形式化定义的集合为P类语言（问题）：

P={L：存在一个多项式时间确定性图灵机程序M，使得L能被M识别，即$L=L_M$}。

1.2 确定图灵机与RAM模型(续)

RAM的组件包括：

（1）一条只读输入带，由一系列的方格和带头组成，每格可以存放一个任意大小的整数。每当读出带头下的符号时，带头自动右移一个方格。

（2）一条只写输出带，由一系列的方格和带头组成，每格可以记录一个任意大小的整数。开始时所有方格都为空白，一旦执行一条指令，一个整数将打印在带头所对准的方格内，然后带头右移一个方格。

（3）存储器，由任意多个普通寄存器R_0，R_1，…组成，每个寄存器可以存放一个任意大小的整数。

（4）程序控制部件，用于控制一个程序。按程序的指向和每条指令的严格定义一步步执行。

（5）程序存储部件，用于存储RAM程序。

1.2 确定图灵机与RAM模型(续)

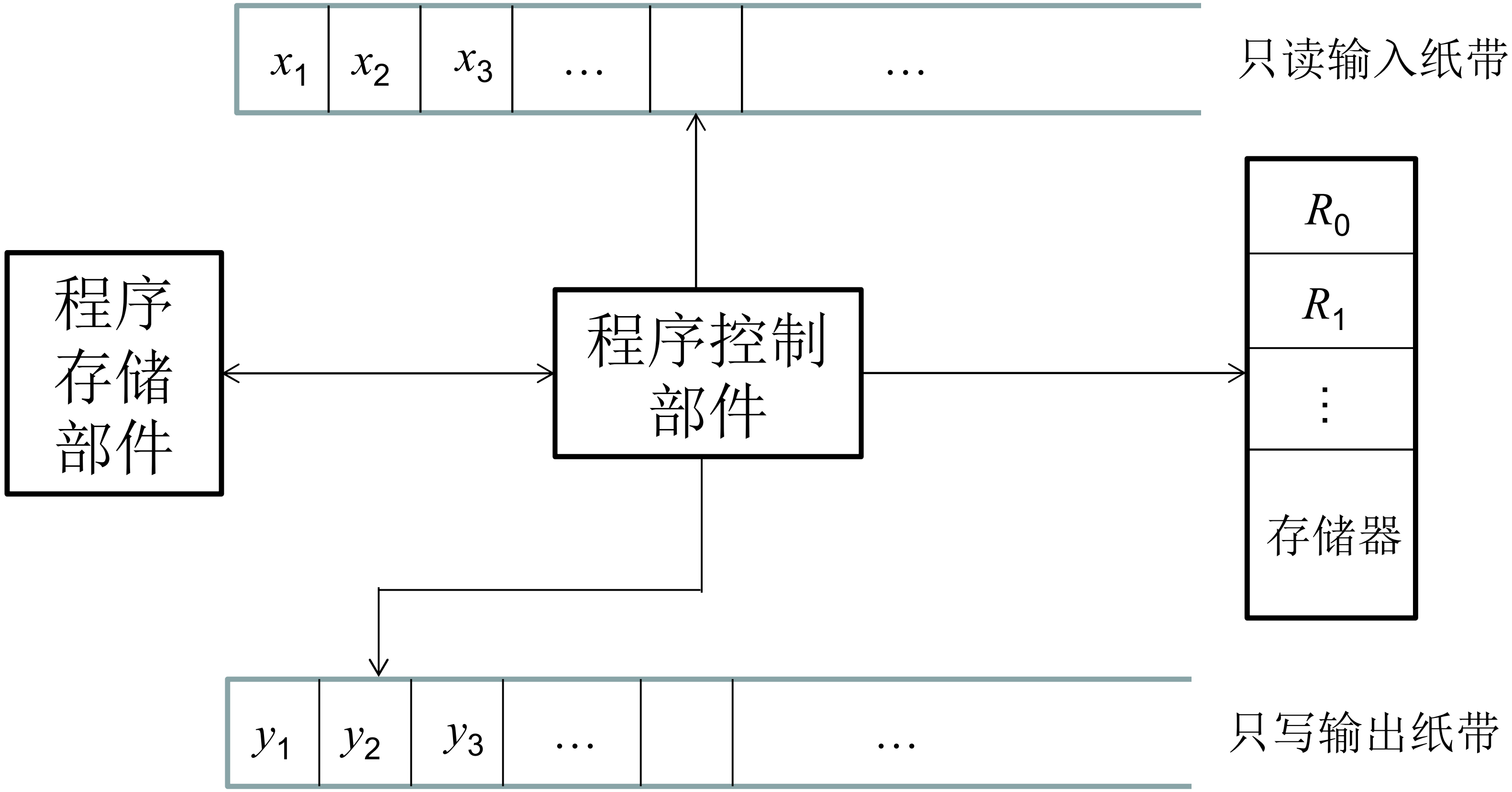

图 2 RAM模型结构示意图

1.2 确定图灵机与RAM模型(续)

RAM的解释

（1）RAM程序定义了从输入信息到输出信息的映射。可以解释为：RAM程序或计算函数，或识别一个语言。

（2）衡量一个算法好坏可以用RAM程序的执行所用的时间和消耗的内存来度量。

（3）RAM的均匀时间耗费是指被执行的指令和转移指令的总条数；均匀空间耗费是指计算中使用过的寄存器的总数。

（4）RAM的对数时间耗费是指被执行的每条指令的时间耗费之和；对数空间耗费是指计算中普通寄存器存过的自然数的最大长度之和。

1.2 确定图灵机与RAM模型(续)

定义**14**对于函数$f_1(n)$和$f_2(n)$，如果存在两个多项式$p_1(x)$和$p_2(x)$，使得除去有限多个n值外，不等式

$$f_1(n) \le p_1(f_2(n))\text{和}f_2(n) \le p_2(f_1(n))$$

成立，则称函数$f_1(n)$和$f_2(n)$是多项式相关的。

定理**2**设确定图灵机是求解某一问题Π的图灵机，对问题Π的任何长度为n的输入，确定图灵机处理它的时间复杂性为$T(n)$。那么，存在一个求解问题Π的RAM程序，它的时间复杂性不超过$O(T^2(n))$。

1.2 确定图灵机与RAM模型(续)

定理2证明.

通过构造一个RAM程序模拟求解问题Π的k带确定性图灵机的工作，来证明本定理。

将图灵机的第j条带上第i个单元与RAM的第$k(i-1)+j+c$单元对应起来，这建立了带方格与RAM内存单元的一一对应关系。这里c是一个常数，它给RAM留下了c个工作单元，其中有k个用于存放图灵机k个带头的当前位置。RAM程序可以借助这k个单元，用间接寻址方式读出（改写）任何带头下的当前符号。而图灵机的带头移动在 RAM中为修改记录带头的当前位置的那个单元内容。

1.2 确定图灵机与RAM模型(续)

定理2证明.（续）

在均匀耗费下，设图灵机的任何计算步时间耗费为1，RAM程序模拟它的时间耗费不超过ck，因此RAM程序能在$O(T(n))$时间内完成对图灵机程序的模拟。

在对数耗费下，RAM处理一个大小为n的整数需要的时间不超过$O(\log_2 n)$。因此，RAM模拟图灵机的时间耗费不超过$c_1 T(n)\log_2 n$，这里c_1是一个常数。因为$n \le T(n)$对图灵机是必然的，所以RAM模拟图灵机所需时间不超过$O(T(n)\log_2 T(n))$，也就是对一切自然数n，$T(n)\log_2 T(n)$不超过$T^2(n)$。

1.2 确定图灵机与RAM模型(续)

定理3设L是一个RAM程序所能识别的一个语言。在对数耗费标准下，这个RAM程序的时间复杂性是$O(T(n))$，存在一台能识别同一个语言L的多带确定图灵机，其时间复杂性为$O(T^2(n))$。

由定理2和定理3可得:

定理4 在对数耗费下，对于同一算法，采用RAM模型和确定图灵机模型的时间耗费是多项式相关的。

1.2 确定图灵机与RAM模型(续)

定理4的意义

图灵机模型比较原始，故要构造一台图灵机来描述一个算法比较困难。而计算问题的分类正是基于图灵机的。大多数情况下，采用RAM模型来描述算法较为容易。在RAM模型下，如果一个算法的复杂性不超过多项式，则这个算法在图灵机模型下也不超过多项式。

谢谢！