第9章 难解性

前言

- 某些计算问题在理论上虽然是可解的，但是获得其解需要大量的时间或空间，其难以在实践中得到应用，这样的问题称为难解的。
- 在第7、8章中，介绍了几个被认为是难解的问题，但是还没有一个得到证明。例如，虽然还不知道怎样证明，但大多数人相信SAT问题和其他所有NP完全问题都是难解的。本章给出几个证明是难解的问题。
- 为了给出这些例子，先证明几个定理，将图灵机的能力与计算可获得的时间或空间数量关联起来。在本意的最后，进一步讨论了证明NP中的问题是难解的可能性，也就是解决P问题与NP问题的可能性。首先，介绍相对化技术，利用它来证明采用某些方法并不会达到这一目标。然后，讨论一种被研究者采用，并已显示出一定前景的方法——电路复杂性理论。

难解性的含义：

若某一计算问题在理论上是可解的，但解法需要耗费大量的时间或空间，以至于无法在实践中应用，则称该问题是难解的。

难解性的形式

难解性可以有多种形式：
例如：

1、一个大部分时间容易计算但偶尔很难算的问题，仅在最坏情况下是难解的；

2、在超级计算机上是易算的，但在PC机上可能需要过量时间的问题

本章的目标

证明存在理论上可判定但实际中不可判定的问题即可判定而难解的问题。

总之：难解性问题指的是在最坏情况下复杂性太大，以至于在任何所能想象建造出来的计算机上所耗费的时间都将超过宇宙的余生。

例如：***SAT***问题和所有的***NP***完全问题。

主要内容

9.1 层次定理

9.2 相对化

9.3 电路复杂性

9.1 层次定理

通常的直觉是给图灵机更多的时间或空间就能扩大它所能求解的问题类。

例如，在时间n^3内，图灵机应能比在时间n^2内判定更多的语言。

在一定条件下，这种直觉是正确的。**层次定理(hierarchy theorems)**证明了这种直觉在满足某些条件下的正确性。

采用术语层次定理，是因为这些定理中的每一个都证明了时间和空间复杂性类不全相同—它们形成一个层次结构，其中时空界限较大的类比时空界限较小的类包含更多的语言。

层次定理的分类

- 空间复杂性层次定理（简单）
- 时间复杂性层次定理（复杂）

9.1 层次定理

空间复杂性层次定理比时间复杂性层次定理稍简单一些，故首先介绍它。在实际陈述定理之前，引入下面的定义。

定义 9.1

函数$f: N \to N$，$f(n)$到少为$O(\log n)$，如果函数f把1^n映射为$f(n)$的二进制表示，并且该函数在空间$O(f(n))$内是可计算的，称该函数为空间可构造的。

含义：如果存在某个图灵机M，在$O(f(n))$空间内运行，并且在输入1^n后总能停机，停机时$f(n)$的二进制表示出现在带子上，则f是空间可构造的

9.1 层次定理

例9.2　通常出现的复杂度至少为O(log*n*)的函数都是空间可构造的，包括$\log_2 n$、$n\log_2 n$和n^2。

例如，n^2是空间可构造的，因为机器以1^n为输入，通过数1的数目得到*n*的二进制开式，采用标准的方法将*n*自乘，输出n^2。全部空间消耗是O(*n*)，当然也是O(n^2)。

9.1 层次定理

定理 9.3

空间层次定理。对于任何空间可构造函数$f:N\to N$，存在语言A，在空间$O(f(n))$内可判定，但不能在空间$o(f(n))$内可判定。

证明思路：必须显示一个语言A具有两个性质：

第一，A在空间$O(f(n))$内可判定；

第二，A不能在空间$o(f(n))$判定。

9.1 层次定理

证明思路：

通过给出判定算法*D*来描述*A*。算法*D*将在 $O(f(n))$ 空间内运行，从而保证了第一个性质。进而，*D*将保证*A*不同于任何在$o(f(n))$空间内可判定的语言，从而保证了第二个性质。

几个问题：

1、为保证*A*不能在$o(f(n))$空间内判定，设计*D*用以实现定理4.9中证明停机问题A_{TM}不可解时所采用的对角线法。如果*M*是在$o(f(n))$ 空间内判定一个语言的机器，则*D*保证*A*与*M*的语言至少存在一点不同的地方。是哪个地方？就是对应于描述*M*自己的地方。

2、看一看*D*的运算方式。简单地讲，*D*把它的输入看作是TM *M*的描述。(如果输入不是任何TM的描述，则*D*在该输入上的动作是无意义的，所以武断地让*D*拒绝即可。)然后*D*在同一输入，即＜*M*＞上，在空间界限 $f(n)$ 内运行*M*。如果*M*在这么大空间内停机,则*D*接受的充分必要条件是*M*拒绝。如果*M*不停机，则*D*拒绝。

9.1 层次定理

证明思路：

几个问题：

3、D与M不能相同。如果相同，D有可能没有足够的空间在输入＜M＞上把M运行完（因为需要额外的处理）。

4、对D的输入进行修改$<M>10^{*}$。如果M在$o(f(n))$空间内运行，则由于渐近行为最终肯定是要消亡的，所以对于某个大的k值，D将有足够的空间在输入$<M>10^{k}$上把M运行完。

5、当D在某个字符串上运行M时，M可能陷入死循环而占用有穷空间。但是D应该是一个判定机，所以必须保证D在模拟M时不会循环。任何在空间 $o(f(n))$ 内运行的机器只消耗时间 $2^{o(f(n))}$ 。修改D，使它计算在模拟M中用掉的步数。如果计数超过 $2^{o(f(n))}$ ，则D拒绝。

9.1 层次定理

定理 9.3

空间层次定理。对于任何空间可构造函数$f:N\to N$，存在语言A，在空间$O(f(n))$内可判定，但不能在空间$o(f(n))$内可判定。

下面的$O(f(n))$空间算法D判定的语言A不能在$o(f(n))$空间内判定。

D="对输入w：

1) 令n是w的长度。

2) 利用空间可构造性计算$f(n)$，并划分出这么多带空间。如果后面的步骤企图使用更多的空间，就拒绝。

3) 如果w不是形如$\langle M\rangle 10^*$，其中M是某个TM，就拒绝。

4) 在w上模拟M，计数模拟过程中使用的步数。如果计数超过$2^{f(n)}$，则拒绝。

5) 若M接受，则拒绝。若M拒绝，则接受。"

O(f(n))

保证D判定性

可在o(f(n))内判定

9.1 层次定理

推论 9.4

对于任意两个函数f_1、f_2：N→N，基中$f_1(n)$等于$o(f_2(n))$，f_2是空间可构造的，有$SPACE(f_1(n)) \subsetneq SPACE(f_2(n))$。

推论 9.5

对于任意两上实数$0 \leqq \varepsilon_1 \leqq \varepsilon_2$，有

$SPACE(n^{\varepsilon 1})$不属于$SPACE(n^{\varepsilon 2})$

9.1 层次定理

推论 9.6

NL⊊ SPACE。

萨维厅定理说明NL不属于SPACE($\log_2 n$)，空间层次定理说明SPACE($\log_2 n$)不属于SPACE(n)，所以推论成立。正如8.5节末所说的那样，就对数空间可归约性而言，TQBF是PSPACE完全的，所以TQBF∉NL。

9.1 层次定理

推论 9.7

PSPACE $\subsetneq$ EXPSPACE。

就判定过程必须消耗多于多项式的空间这一意义而言，该推论证明存在难解的但可判定的问题。语言本身有一些不太自然——它们只是为了分享复杂性类才有意义。在讨论时间层次定理以后，得用这些语言来证明其他更加自然的语言的难解性。

9.1 层次定理

定义 9.8

函数t：$N \to N$，$t(n)$至少为$O(n \log n)$，如果函数把1^n映射为$t(n)$的二进制表示，并在时间$O(t(n))$内可计算，则称该函数为是时间可构造的。

含义：如果存在某个TM M，在时间 $O(t(n))$ 内运行，而且在输入1^n上启动后总能停机，停机时$t(n)$的二进制表示出现在带子上，则t是时间可构造的。

9.1 层次定理

例9.9 通常出现的不小于$n\log n$的函数都是时间可构造的，包括函数$n\log n$, $n\sqrt{n}$, n^2以及2^n。

例如，为了证明$n\sqrt{n}$是时间可构造的，首先设计一个TM，以二进制计算1的个数。为此该TM沿着带子移动一个二进制计数器，每到一个输入位置把它加1，直至输入的末端。因为对于n个输入位置的每一个都需要消耗$O(\log n)$步，所以这部分工作消耗$O(n\log n)$步。然后，从n的二进制表示中计算出$\lfloor n\sqrt{n} \rfloor$的二进制形式。因为涉及的数的长度是$O(\log n)$，所以任何合理的计算方法都将消耗$O(n\log n)$时间。

9.1 层次定理

定理 9.10

时间层次定理。对于构造函数t: $N \to N$，存在语言A，在时间O(t(*n*))内可判定，但在时间o(*t*(*n*))/log*t*(*n*)内不可判定。

证明思路：

本定理的证明类似于定理9.3的证明。构造一个TM *D*在时间O(t(*n*))内判定语言*A*，而*A*不能在时间o(*t*(*n*))/log*t*(*n*)内被判定。这里，*D*读取一个形如<M>10*的输入*w*，模拟*M*在输入*w*上的运行，确保时间消耗不超过t(*n*)。如果*M*停机的时间在这之内，则*D*给出相反的输出。

证明中重要的差别涉及到模拟*M*的开销，及计算模拟所使用的步数的开销。机器*D*必须有效地执行这种定时的模拟，以使*D*在时间O(t(*n*))内运行的同时，避开所有在时间o(*t*(*n*))/log*t*(*n*)内可判定的语言。

9.1 层次定理

定理 9.10

时间层次定理。对于构造函数t: $N \to N$，存在语言A，在时间O(t(*n*))内可判定，但在时间o(*t*(*n*))/log*t*(*n*)内不可判定。

下面的O(*t*(*n*))时间算法D所判定的语言A不是在o(*t*(*n*))/log*t*(*n*)时间内可判定的。

D="对输入*w*:

1) 令*n*是*w*的长度。

2) 利用时间可构造性计算*t*(*n*)，把 $\lceil t(n)/\log(n) \rceil$ 存放在一个二进制计数器中每一次执行步骤3、4、5之前，把该计数器减1。如果计数器减到0，就拒绝。

3) 若*w*的形式不是<m>10*，其中M是某个TM，就拒绝。

4) 在 *w*上模拟M .

5) 若M 接受，则拒绝；若M 拒绝，则接受。"

9.1 层次定理

考察该算法的每一步以确定运行时间。显然，步骤1、2、3能够在为O(t(n))时间内完成。在步骤4，每次D模拟M的一步，它都要读取M的当前状态以及M读写头下的带符号，在M的转移函数中查找M的一下个动作，以使它能够适当地更新M的带内容。所有这三个对象都存放在D的带上某处。如果它们彼此分开很远，D每次模拟M的一步都需要走许多步来收集这些信息。所以，D总是把这些信息放在一起。

可以把D的单带组织成轨道。得到两条轨道的一种方法是以奇数位置存储一条轨道，以偶数们置存储另一条轨道。另一种获得两条轨道效果的方法是扩大D的带字母表，使它包括每一对符号，其中，一个符号来自上轨道，另一个符号来自下轨道。更多的轨道效果也可以类似独。注意，如果只使用固定数目个轨道，多轨道主演唱会增加一数倍的时间开销。这里，D采用三条轨道。

9.1 层次定理

推论 9.11

对于任意两个函数t_1, t_2：N→N，其中$t_1(n)$等于$o(t_2(n))/\log t_2(n)$而且t_2是时间可构造的，有$TIME(t_1(n))$不属于$TIME(t_2(n))$。

推论 9.12

对于任意两个实数$1 \leqq \varepsilon_1 \leqq \varepsilon_2$，有$SPACE(n^{\varepsilon_1})$不属于$SPACE(n^{\varepsilon_2})$。

推论 9.13

$P \subsetneq EXPTIME$。

9.1 层次定理

指数空间完全性

下边将证明，如果允许正则表达式采用比通常的正则运算更多的运算，则分析表达式的复杂性将急剧上升。设↑是**指数运算**(exponentiation operation)，若R是一个正则表达式，*k*是一个非负整数，则写法R↑k等价于*R*自身连接*k*次。也可把R↑k缩写为R^k。换言之，

$$R^k = R \uparrow k = \overbrace{R \circ R \circ \ldots \circ R}^{k}$$

广义正则表达式允许指数运算，也允许通常的正则运算。显然，广义正则表达式仍然产生正则语言，跟标准正则表达式一样。通过重复基本的正则运算表达式，可以除去指数运算。令

$EQ_{REX\uparrow}$ ={<*Q*，*R*>| *Q*和*R*是等价的带指数运算的正则表达式}

9.1 层次定理

为了证明$EQ_{REX\uparrow}$是难解的，可以证明它对于类EXPSPACE是完全的。任何EXPSPACE完全问题都不可能在PSPACE中，更不用说P了。否则EXPSPACE将等于PSPACE，与推论9.7矛盾。

定义 9.14

语言B是EXPSPACE完全的，如果

1) B∈EXPSPACE,并且

2) EXPSPACE中的每个A都多项式时间可归约到B。

定理 9.15

EQREX↑是EXPSACE完全的。

9.1 层次定理

证明思路：

在度量 $EQ_{REX\uparrow}$ 的复杂性时，假定所有指数都写成二进制整数。表达式的长度是它包含的所有符号的总数。

接下来概略地叙述判定 $EQ_{REX\uparrow}$ 的**EXPSPACE**算法。为了判定两个带指数的表达式是否等价，首先用重复表达式的办法删除指数，然后把得到的表达式转化为**NFA**。最后再利用一个类似于例**8.4**中判定 $\overline{ALL_{\mathrm{NFA}}}$ 的算法的**NFA**等价性判定过程。

9.1 层次定理

N＝“对输入＜N_1，N_2＞，其中N_1和N_2是NFA。

1)给N_1和N_2的起始状态打上标记。

2)重复下面操作$2^{q_1+q_2}$次，其中q_1和q_2是N_1和N_2的状态数。

3) 非确定地选择一个输入符号，改变在N_1和N_2的状态上标记的位置，以模拟诗篇这个符号。

4）若在任何时刻，标记放在一个有穷自动机的接受状态上，而没有放在另一个有穷自动机的接受状态上，则接受；否则，拒绝。“

9.1 层次定理

如果自动机N_l和N_2是等价的，N显然拒绝，因为它只在确定—台机器接受某个串而另一台机器不接受时才能接受。如果这两个自动机不等价，则存在某个字符串被一台机器接受而不能被另——台接受。这样的字符串中必定有长度小超过$2^{q_1+q_2}$的。若个然，考虑用这样的串中最短的一个作为非确定选择的序列。因为只存在$2^{q_1+q_2}$种不同的方式把标记放在N_1和N_2的状态上，所以在更长的中标记的位置必定重复。把介于重复之间的那部分字我们字符串删除，就得到更短的字符串。因此算法N在它的非确定选择中会独到这个串并接受。所以N运算正确。

算法N在非确定线性空间内运行，于是根据萨维奇定理，可以得到判定该问题的确定型$O(n_2)$空间算法。下面用该算法的确定形式设计判定$EQ_{REX}\uparrow$的算法E。

9.1 层次定理

E＝“对输入＜R_1，R_2＞．其中R_1，和R_2是带指数的正则表达式；

1) 把R_l和R_2转化为等价的正则表达式B_l和B_2，其中B_l和B_2利用重复代替指数。

2)利用引理2．29的证明中给出的转化过程，把B_1和B_2转化为等的NFA N_1和N_2。

3)利用算法N的确定版来判定讨N_l与N_2是否等价。”

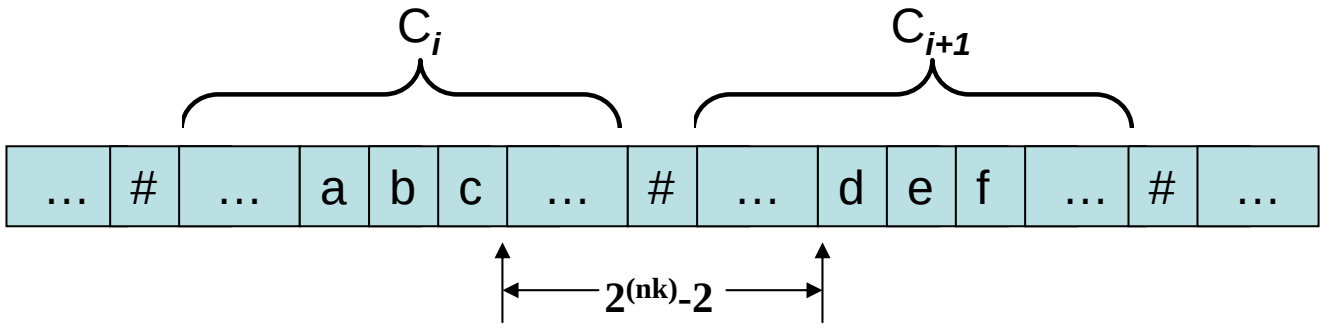
显然算法E是正确的。为了分析它的空间复杂性，注意到用重复代替指数可能会把表达式的长度增加2^l倍，其中l是指数的长度和。于是表达式B_1和B_2的长度最大为$n2^n$，其中n是输人的长度。引理2．29的转化过程使长度线性增加，因此NFA N_1和N_2最多有$O(n2^n)$个状态。因为输人长度是$O(n2^n)$，所以算法N的确定版消耗空间$O((n2^n)^2)＝O(n^2 2^{2n})$.于是$EQ_{REX}$↑是指数空间可判定的。

9.1 层次定理

- 下面证明$EQ_{REX}\uparrow$是EXPSPACE难的。设TM M在空间$2(n^k)$，内判定语言“A，k是某个常数。归约把输人w映射为一对正则表达式R_1和R_2。表达式R_1就是△*,若用Γ和Q表示M的带字母表和状态集，则 ＝Γ ∪ Q ∪ {#}是计算历史中可能出现的所有符号组成的字母表。构造表达式R_2，使它产生不代表M在w上的拒绝计算历史的所有字符串。当然，M接受w，当且仅当M在w上没有拒绝计算历史。因此这两个表达式等价当是仅当M接受w。构造过程如下。
- M在w上的一个拒绝计算历史是由符号#分隔的一系列格局。采用标准的格局编码，其中代表当前状态的符号放在当前读写头位置的左边。假定所有格局的长度为$2^{(n^k)}$，如果长度不够，就用空白符填在右边。拒绝计算历史的第 一个格局是M在w上的初始格局，最末格局是一个拒绝格局。每一个格局都必须根据转移函数中指定的规则从前一个格局转变而来。

9.1 层次定理

- $R_{bad\text{-}start}=S_0 \cup S_1 \cup \ldots \cup S_n \cup S_b \cup S_{\#}$
- $S_b=\triangle^{n+1}(\triangle \cup \varepsilon)2^{(n^k)-n-2}\triangle_{-\square}\triangle^*$
- $S_{\#}=\triangle^{(2(n^k))}\triangle_{-\#}\triangle^*$
- $R_{bad\text{-}reject}=\triangle_{-}^{*}{}_{qreject}$



主要内容

9.1 层次定理

9.2 相对化

9.3 电路复杂性

9.2 相对化

用对角化法证明一个非确定型多项式时间TM能够判定一个不在P中的语言？

在相对化方法中，将修改计算模型，给图灵机一些本质上是“免费”的信息。依据实际提供给它的信息，图灵机就可能比以前更轻松地解决某些问题。

例如，假定对任何长度的布尔公式，给图灵机以在一步内解决可满足性问题的能力。不管怎样实现这一奇迹——想象一个附带的“黑匣子”给了机器这种能力。可以把这个黑匣子称为**谕示(oracle)**，以强调它没必要对应于任何物理设备的性质。显然．不管P是否等于NP，机器都可以利用谕示在多项式时间内解决任何NP问题，因为每一个NP问题都可多项式时间归约到可满足性问题。这样的图灵机被称为相对于可满足性问题进行计算，因此才有术语相对化。

9.2 相对化

定义 9.16

针对一个语言A的谕示是一个能够判断任何串w是否在该语言中的设备。谕示图灵机M^A就是给通常的图灵机附加一条带子，称为谕示带。每当M^A在谕示带上写下一个字符串时，它就能在一步计算内得知这个字符串是否属于A。

例9.17 如前面提到的，相对于可满足性问题的多项式时间计算包含了 NP的全部．换言之，$NP \subseteq P^{SAT}$。进而，$coNP \subseteq P^{SAT}$，因为P^{SAT}是一个确定性复杂性类，在补运算下封闭。

9.2 相对化

例 9.18 正如P^{SAT}包含我们认为不属于P的语言一样。NP^{SAT}包含我们认为不属于NP的语言。例如，两个在变量x_1，…，x_t上的布尔公式 ϕ 和 ψ，如果它们在任何变量赋值上都有相同的值，则称它们为等价的。如果一个公式没有更小的公式和它等价，则称它为极小的。

令：

NONMIN—FORULA={$<\phi>|\ \phi$不是极小布尔公式}

9.2 相对化

- 下一个定理表明存在谕示A和B，使得可以证明P^A与NP^A不同，而P^B与NP^B相同。这两个谕示很重要，因为它们的存在表明不太可能用对角化方法解决P与NP问题。
- 对角化方法的核心是一台图灵机对另一台图灵机的模拟。模拟是这样完成的：模拟机器能够确定另一台机器的行为，从而以不同的方式动作。假定给这两台机器以同样的谕示，那么每当被模拟机器询问谕示时，模拟机也询问，所以模拟过程可以象以前一样进行下去。因此，凡是仅用对角化方法证明的关于图灵机的定理，当给两台机器以相同谕示的时候，将仍然成立。
- 特别地讲，如果能用对角化方法证明P与NP不同，就能下结论说它们相对于任何谕示也是不同的。但是P^B与NP^B相等，所以结论不成立。因此对角化方法不足以分开这两个类。类似地，依据简单模拟的证明不能说明这两个类相等，因为那将证明它们相对于任何谕示都是相等的，但实际上P^A与NP^A不相等。

9.2 相对化

定理 9.19

1)存在谕示A使得$P^A \neq NP^A$。

2)存在谕示B使得$P^B = NP^B$。

证明思路：展示谕示B是容易的，只需令B是任意的PSPACE完全问题，如$TQBF$即可。

构造谕示A。设计A使得NP^A中的某个语言L_A在证明它时需要蛮力搜索，因而L_A不可能属于P^A。因此可以得出$P^A \neq NP^A$的结论。构造过程依次考察每一个多项式时间谕示机器，保证每一个都不能判定语言L_A。

9.2 相对化

证明：令B是TQBF。有下面一系列包含关系：

$$NP^{TQBF} \subseteq NPSPACE \subseteq PSPACE \subseteq P^{TQBF}$$

包含关系1成立，是因为可以把非确定型多项式时间谕示机器转变为非确定型多项式空间机器，该机器不使用谕示，而是计算出对TQBF的查询的答案。由萨维奇定理知包含关系2成立。包含关系3成立是因为TQBF是PSPACE完全的。因此下结论$P^{TQBF}=NP^{TQBF}$。

下面说明怎样构造谕示A .对于任意的谕示A，令L_A是所有这样的字符串的集合：A中有一个长度相同的字符串。于是

$$L_A=\{w|\exists x\in A[|x|=|w|]\}$$

显然，对于任何A，语言L_A属于NP^A。

9.2 相对化

- 为了证明L_A不属于P^A，如下设计A。设$M_1, M_2,\ldots$，是所有多项式时间谕示TM的列表。为了简单，假定Mi在时间n^i内运行。构造过程分步骤进行。第i步构造A的一部分，保证M_i^A不判定L_A。通过声明某些字符串在A中，而另外一些字符串不在A中来构造A。每一步仅确定有穷个字符串的状态(指在或者不在A中)。开始，没有关于A的任何信息。从步骤1开始.
- 步骤i　迄今为止，已经声明有穷个字符串在或不在A中。选择n，让它比所有这些串的长度都长，而且n充分大使得2^n比n^i大，n^i是M_i的运行时间。现在说明怎样扩展关于A的信息，使得只要l^n不在L_A中，M_i^A就接受它。
  - 在输入1^n上运行M_i，如下回应它的谕示查询。如果M_i查询—个状态已经确定的字符串y，回应与它的状态一致。如果y的状态还未确定，就对查询回应NO．并声明y不在A中。继续模拟M_i直到它停机。

9.2 相对化

- 现在从M_i的角度考虑问题。如果M_i发现一个长为n的字符串在A中，它就应该接受，因为它知道1^n在L_A中。如果Mi确定所有长为n的字符串都不在A中，它就应该拒绝，因为它知道1^n不在L_A中。但是，它没有足够的时间来询问所有长为n的字符串，而且我们已经对它所做的每一个查询都回答NO。所以当取停机时必须决定是接受还是拒绝，而它还没有足够的信息保证它的决定是对的。
- 现在的目标是确保它的决定是不对的。为此，观察它的决定，然后扩展A使其反面成立。于是，如果M_i接受1^n.就声明所有剩下的长为n的字符串都不在A中，从而确定1^n不属于L_A. 如果M_i拒绝1^n,我们就找到一个M_i未查询过的长为n的字符串，并声明该字符串属于A，从而保证l^n属于L_A。这样的字符串必定存在，因为M_i运行ni步，少于长为n的字符中总数2^n,这样就达到了步骤i的目标，该步完成。从步骤i+l继续进行下去。
- 在完成所有步骤以后，把那些经过所有步骤以后状态还没有确定下来的字符串都声明为不在A中，从而完成对A的构造。没有多项式时间谕示机器能够以谕示A判定L_A，定理得证。
- 总之，相对化方法告诉我们；为了解决P与NP问题，必须分析计算，而不仅仅是在模拟它们。

主要内容

9.1 层次定理

9.2 相对化

9.3 电路复杂性

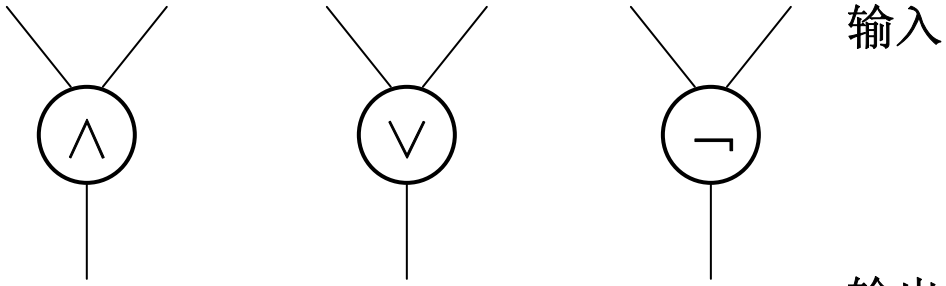
9.3 电路复杂性

将按照数字电路设计的电子器件用导线连在一起就构成了计算机。也可以用对应数字电路的理论模型，称为布尔电路，来模拟如图灵机这样的理论模型。

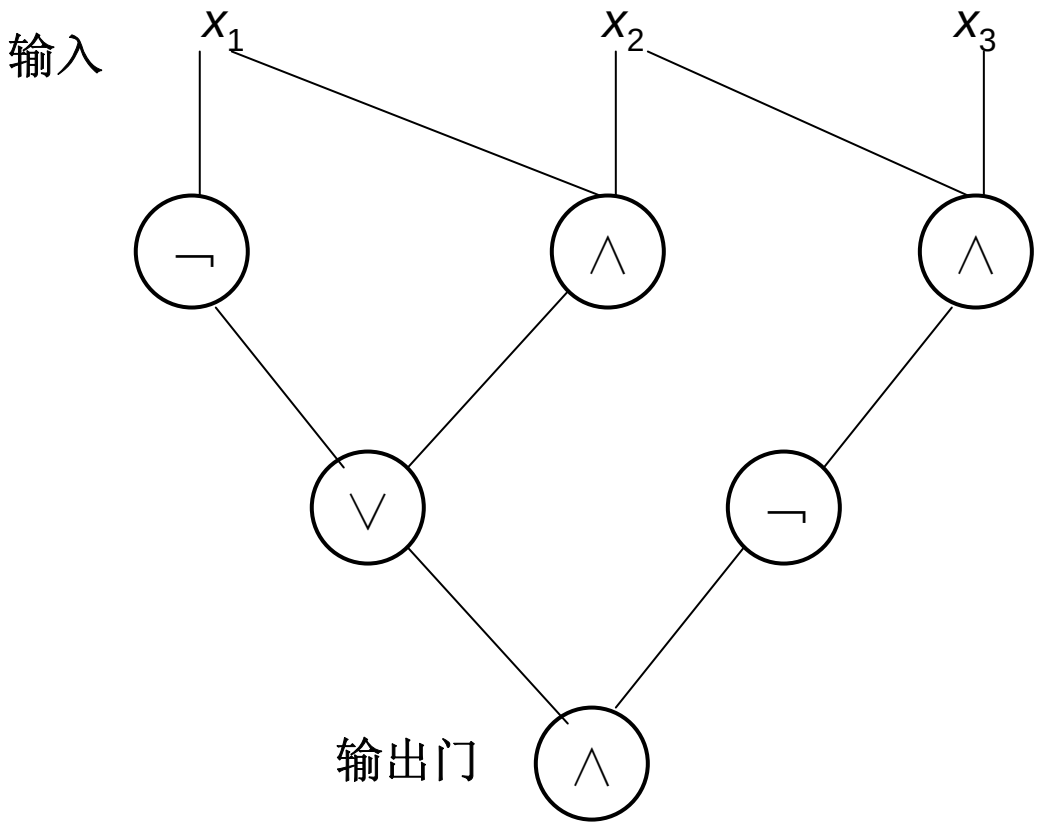
建立图灵机与布尔电路之间的联系有两个目的。
第一，研究者相信电路提供了一种方便的计算模型，用以处理P与NP问题及其相关问题。
第二，电路为有关*SAT*是NP完全的库克一列文定理提供了另一种证明方法。本节讨论这两个主题。

9.3 电路复杂性

定义 9.20 一个布尔电路是由导线连接的门和输入的集合。不允许有循环。门有三种形式：与门、或门和非门，如图所示。



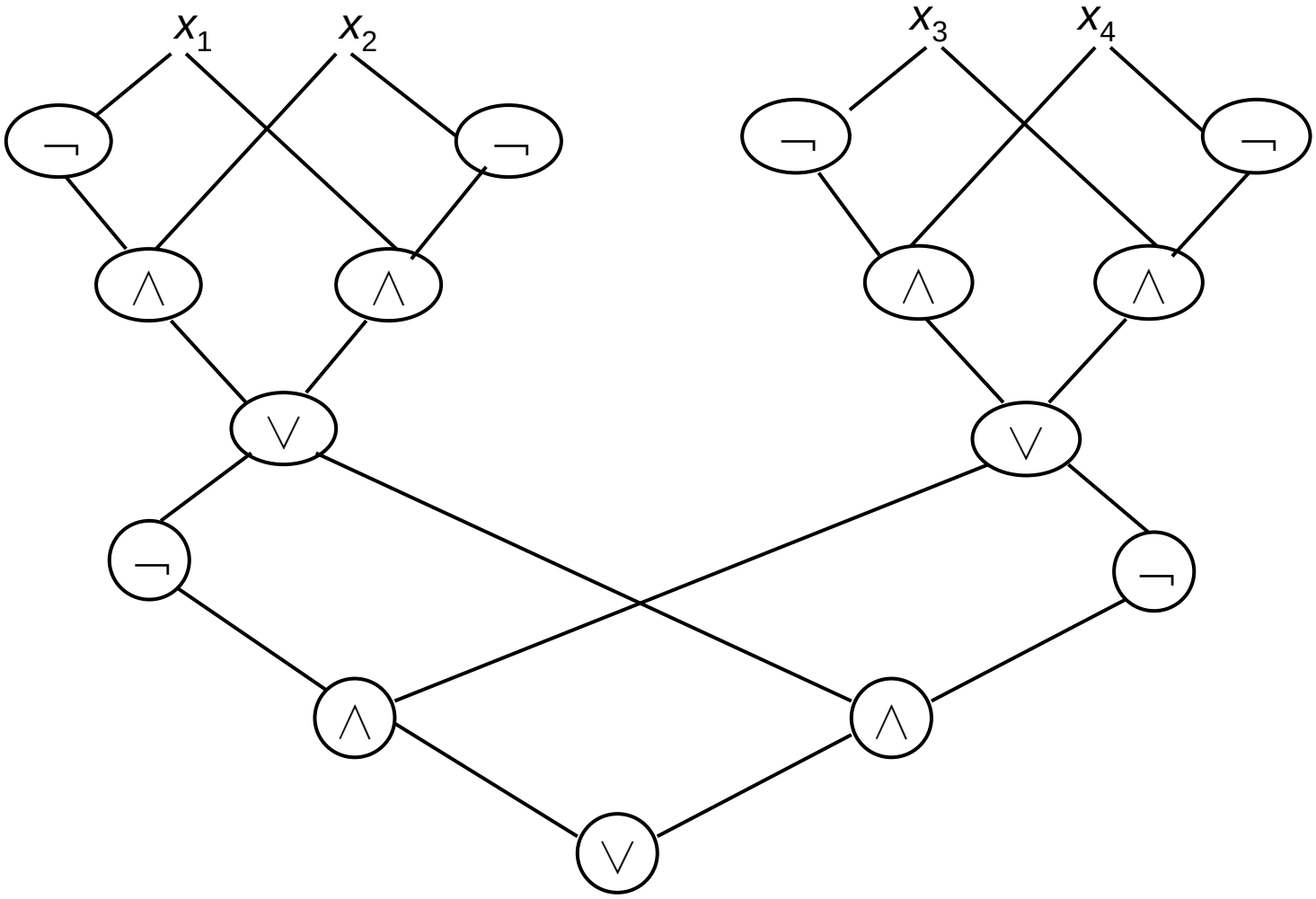
9.3 电路复杂性



9.3 电路复杂性

用函数来描述布尔电路的输入/输出行为。给带 n 个输入变量的布尔电路 C 关联一个函数 $f_C:\{0,1\}^n \to \{0,1\}$，其中，如果当输入 x_1，…，x_n 设置为 a_1，…，a_n 时，C 的输出为 b。则写 $f_C(a_1, \cdots, a_n) = b$。称 C 计算函数 f_C。有时考虑有多个输出门的布尔电路。带 k 个输出位的函数计算一个值域为 $\{0,1\}^k$ 的函数。

9.3 电路复杂性



9.3 电路复杂性

例 9.21 n-输入奇偶函数$parity_n$：$\{0,1\}^n \to \{0,1\}$输出1，当且仅当输入变量中有奇数个1。下面的电路计算有4个变量的奇偶函数$parity_4$。

实际上，可以用电路来检查适当地编码为{0,1}上的语言的成员资格。存在的问题是任何具体的电路只能处理某一固定长度的输入，而语言可能包含不同长度的字符串。所以，不是用单一电路来检查语言的成员资格，而是用整个一族电路来完成这一任务，每个输入长度有一个电路。

定义 9.22

一个电路族C是电路的一个无穷列表$(C_0, C_1, C_2, \ldots)$，其中C_n有n个输入变量。称C在{0，1}上判定语言A，如果对于每个字符串w，$w \in A$当且仅当$C_n(w)=1$，其中n是w的长度。

9.3 电路复杂性

电路的规模是它所包含的门的数目。当两个电路有同样的输入变量，并且在每一输入赋值上都输出同样值的时候，称它们是等价的。如果一个电路没有与之等价的更小的电路，则它是(规模)极小的。极小化电路问题在工程上有明显的应用，但是一般难以解决。甚至判定一个具体的电路是否是极小的，也不见得是在P或NP内可解的。一个语言的电路族中每一个电路C_i如果是极小的，则称该族是极小的。一个电路族($C0$，$C1$，$C2$，…)的规模复杂度是一个函数f：$N \to N$，其中$f(n)$是Cn的规模。

电路的深度(depth)是从输入变量到输出门的最长路径的长度(导线的数目)。如同定义电路的规模一样，可以定义深度极小的电路和电路族，以及电路族的深度复杂度(depth complexity)。

定义 9.23

语言的电路(规模)复杂度是该语言的极小电路族的规模复杂度。语言的电路深度复杂度类似定义，只是把规模换成深度。

9.3 电路复杂性

例 9.24 容易推广例9．21，给出计算n个变量的奇偶函数的有O(*n*)个门的电路。一种办法是构造一棵由计算XOR函数的门组成的二叉树．这里XOR函数就是2-奇偶函数，然后如前面的例子那样，用2个NOT、2个AND和1个OR实现每个XOR门。

- 令A是包含奇数个1的字符串组成的语言，那么A的电路复杂度是O(*n*)。
- 语言的电路复杂度与它的时间复杂度有关。任何时间复杂度小的语言，其电路复杂度也小，如下面的定理所述。

9.3 电路复杂性

定理 9.25

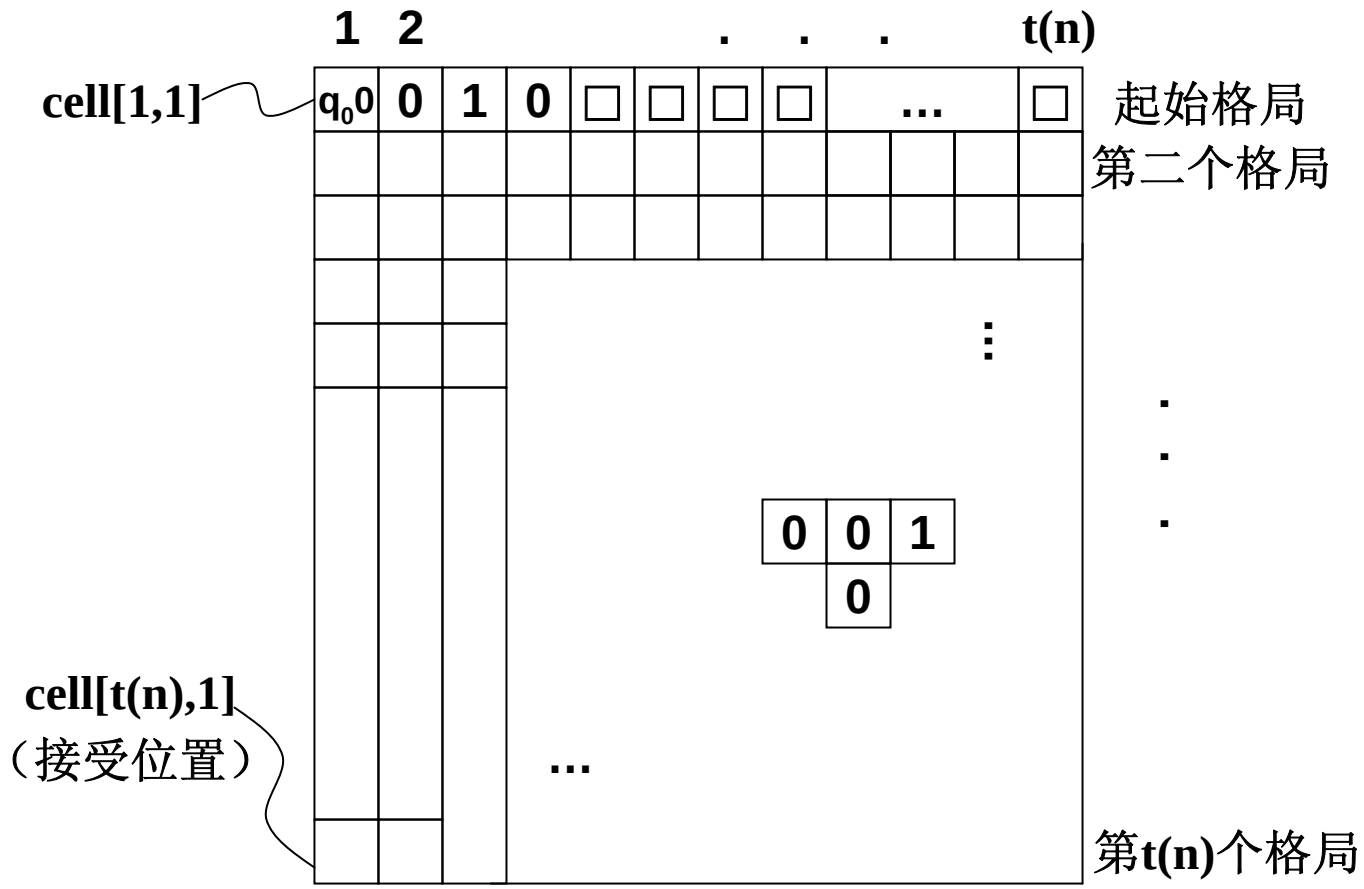
设t：N→N是一个函数，t(*n*)≧n 。若A ∈ TIME(t(*n*))，则A的电路复杂度为O(t²(*n*))。

证明思路： 设 M 是在时间 $t(n)$ 内判定 A 的 TM。(为了简单，忽略 M 的实际运行时间 $O(t(n))$ 中的常数因子。对每一个 n，构造电路 C_n 模拟 M 在长为 n 的输入上的运算。C_n 的门分成行，每一行对应 M 在长为 n 的输入上进行运算的 $t(n)$ 步之一。门的每一行代表 M 在相应步骤上的格局(状态，读写头位置，带内容)。每一行用导线连到上一行，以使它从上一行的格局能够计算本格局。修改 M 使得输入编码为{0,1}。另外，当 M 即将接受时，它把读写头移到最左单元，在进入接受状态之前先在那里写下符号␣。这样一来，就可以指定电路的最后一行的一个门为输出门。

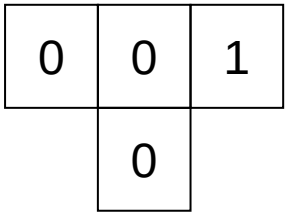
9.3 电路复杂性

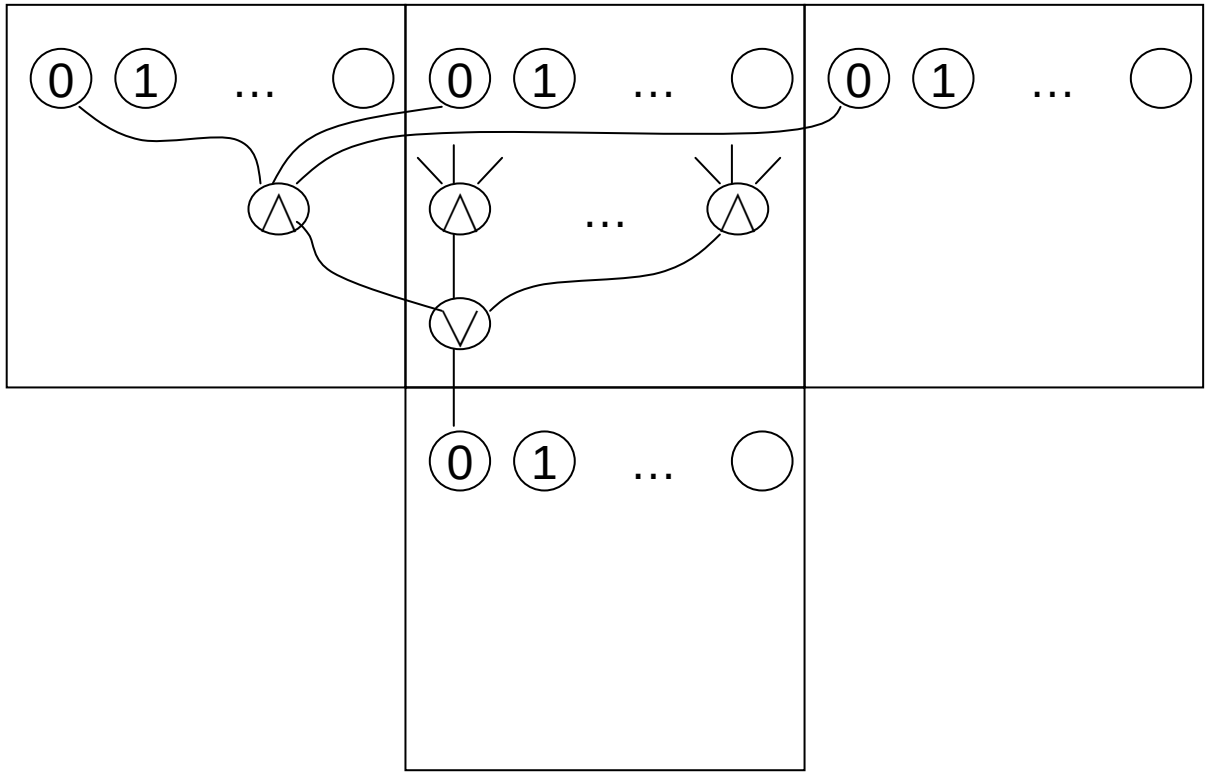
- 该定理提供了一条证明P≠NP的途径，即试图证明NP中的某个语言的电路复杂性超过多项式。
- 证明：设M＝(Q，Σ，Γ,，δ，q0，q_{accept}，q_{reject})在时间t(n)内判定A，设*w*是长为n的输入。定义M在w上的画面为一张t(n)*t(n)的表格，其行是M的格局。画面的顶行包含M在*w*上的起始格局。第i行包含计算历史的第i步的格局。
- 为了方便，本证明修改了格局的表示形式。在3.1.1节描述的老的形式中，状态出现在读写头正在读取的符号的左边。与之不同，现在把状态和读写头下的符号表示为一个单一的复合字符。例如，若M在状态q，带上包含字符串1011，读写头正读取左起第二个符号，老的形式为1q011，新形式为1q011．其中复合字符则q0同时表示状态q和读写头下的符号0。

9.3 电路复杂性



9.3 电路复杂性





9.3 电路复杂性

定理 9.26

CIRCUIT-SAT是NP完全的。

为了证明该定理，必须证明CIRCUIT-SAT属于NP，并且NP中的任何语言A都归约到*CIRCUIT-SAT*。第一点是显然的。为了证明第二点，必须给出多项式时间归约*f*，把字符串映射为电路，其中

$$f(w)=<C>$$

蕴涵

$W \in A \equiv$布尔电路C是可满足的

因为A属于NP，所以它有多项式时间验证机V，其输入的形式为<*x*,*c*>，这里C可以是证明x属于A的证书。为了构造f，用定理9.25的方法得到模拟V的电路。把w的符号赶往对应于x的电路的输入。剩下的电路输入都对应证书C。把这个电路称为C，并输入它。若C是可满足的，则存在一个证书，所以*w*属于A。反之，若*w*属于A，则存在一个证书，所以C是可满足的。

9.3 电路复杂性

定理 9.27

3SAT是NP完全的。

作业

- **9.4、9.7、9.11、9.25、9.26**