Time Complexity

- **Section 7.1 Measuring Complexity**
  - Big O / Little o notation, asymptotic(渐进）
  - Time/space complexity
- **Section 7.2 polynomial time (P),**
  - Strong Church-Turing Thesis
- **Section 7.3 NP Class**
  - Polynomial time reductions
  - Reduction from 3SAT to k-CLIQUE
- **Section7.4 NP-Completeness**
  - Cook-Levin Theorem

同濟大學 TONGJI UNIVERSITY

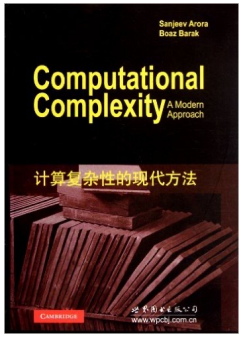
Chapter 7 Time Complexity

Objective

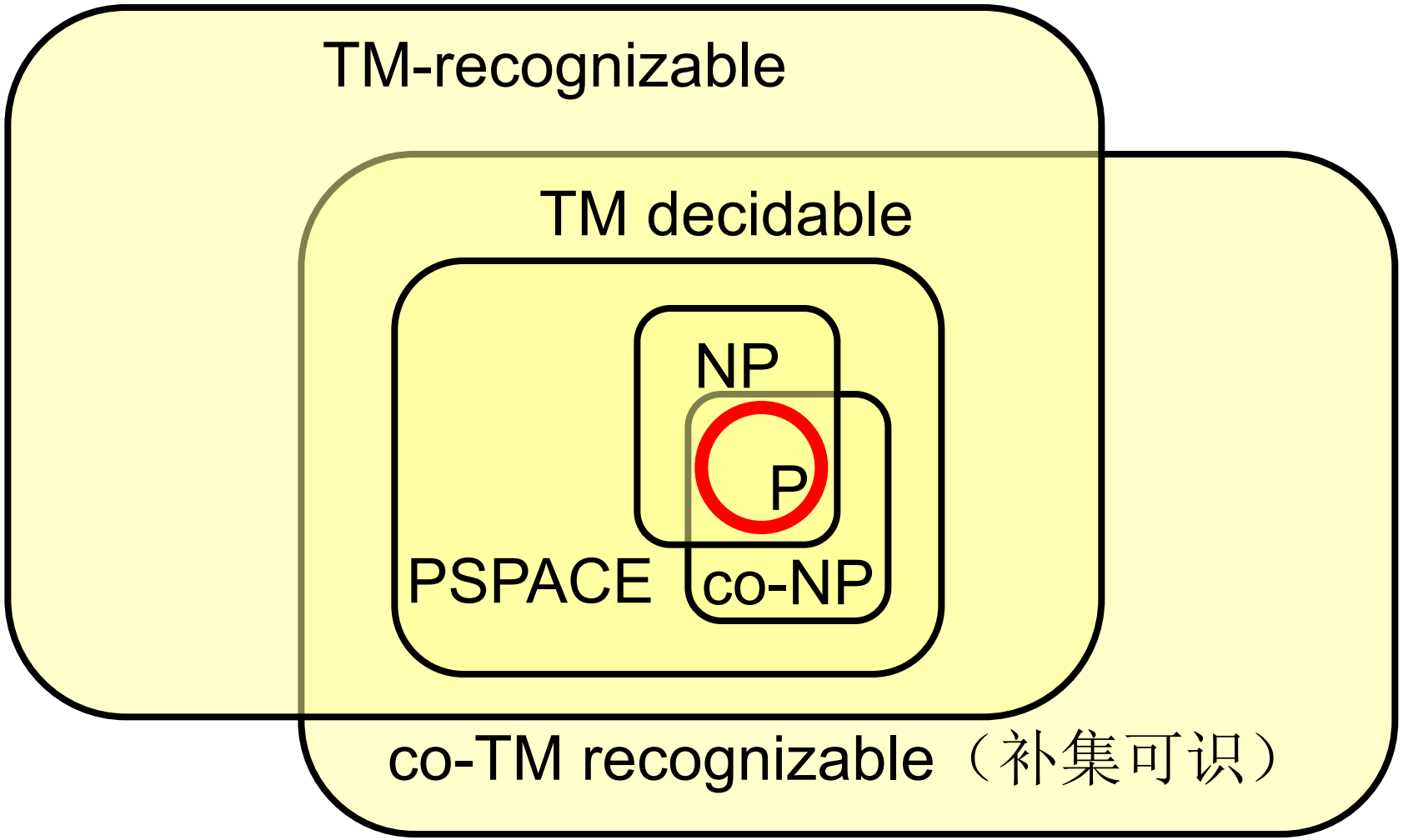
- In complexity theory, the objective is to classify problems as easy and hard ones;
- In computability theory, the classification of problems is by those that are solvable and those that are not.

Recommend:

Arora, Sanjeev. Computational complexity : a modern approach. 世界图书出版公司北京公司, 2012.



Refining Decidability



同济大学 TONGJI UNIVERSITY

7.1 Measuring Complexity

EXP: **Consider the prime decision problem**

PRIME = { x | x∈N, x is a prime }

The (perceived) difficulty of PRIME is illustrated by the fact that the x∈PRIME? question seems to become much more difficult as x gets bigger.

感性认识： 当输入的**x**变大时, 判定素性就变难

We will study the relation between the bit-size of a problem instance and the required time/ space complexity of the solution for such an instance (worst case).

所以，要考虑 问题难度与 输入大小的关系

同济大学 TONGJI UNIVERSITY

Running Time / Time complexity

Definition 7.1: Let M be a deterministic Turing machine that halts on all in- puts. The ***running time*** or ***time complexity*** of M is the function f: N→N, where f(n) is the maximum number of **steps** that M uses on any input of length **n**.

Note:

$$f(n) = \max_{|x|=n}(\text{no. of time-steps of M on x})$$

Worst case and size of the input x in bits.

we say: *"f(n) is the running time of M"*, or *"M is an f(n) time Turing machine"*.

同济大学 TONGJI UNIVERSITY

Asymptotic Analysis

Definition 7.2: Let f and g be two functions $N \to R^+$. We say and write f(n) = O(g(n)) if and only if there are two positive constant c and n_0 such that $f(n) \le c \cdot g(n)$ for all $n \ge n_0$.

We say that:

"g(n) is an (asymptotic) upper bound on f(n)"

渐进上界.

Note: you may think of O as representing a suppressed constant.

同济大学 TONGJI UNIVERSITY

Some big O examples

看长远，比谁大

Let $f(n) = 15n^2 + 7n$ and $g(n) = ½ \cdot n^3$.

We have $f(n)=O(g(n))$ because for $n_0=16$ and $c=2$, we see indeed for all $n \geq n_0$:

$f(n) = 15n^2 + 7n \leq 16n^2 \leq n^3 = c \cdot g(n)$.

长远趋势≤， 渐进上界

More tight is $5n^4 + 27n = O(n^4)$.

Take $n_0=1$ and $c=32$.

(But $n_0=3$ and $c=6$ works also.)

同济大学 TONGJI UNIVERSITY

Polynomials vs Exponentials

Let f be a k-th degree polynomial $f(n) = a \cdot n^k + \ldots$,
then $f(n) = O(n^r)$ for all $r \geq k$. 多项式看最高项目
当 $n \to \infty$, $\lim (f(n) / n^K) \to c$

Exponential functions like 2^n always ‘overpower’ polynomials.
For all constants a and k,
the function $f(n) = a \cdot n^k + \ldots$, will obey: $f(n)=O(2^n)$.
罗比达法则 可以证明 多项式 低于指数

同济大学 TONGJI UNIVERSITY

Logarithms 对数 低于 多项式

$n = O(n \cdot \log(n))$ but $n \cdot \log(n) = O(n^d)$ for which d?

Answer: For every d>1.

Polynomials dominate (powers of)-logarithms, just like exponentials overpower polynomials.
对数 低于 多项式

Note that because $a \cdot \log(n) = O(\log(n))$ for all a, we do not have to indicate the base b of $\log_b$.
对数换底相当于 乘以常数因子，可忽略

同济大学 TONGJI UNIVERSITY

The O-Ordering，变化趋势 ≤

We can also put the O in the exponent:

$f(n) = 2^{O(\log n)}$ thus implies (for big enough n):

$f(n) \leq 2^{c \cdot \log(n)} = n^c$ for some c.

同济大学 TONGJI UNIVERSITY

Little o-Notation 小o

Definition 7.5: Let f and g be two functions $N \to R^+$.
We say and write f(n) = o(g(n)) if and only

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

Where big-O is about “less-or-equal-than” <=,
little o is about “strictly less than” <.
可用罗比达法则求极限验证

同济大学 TONGJI UNIVERSITY

Analyzing Algorithms---*single-tape DTM*

Consider a 1-tape TM that decides the language
$A = \{ 0^k1^k \mid k=0,1,2,\ldots \}$.

M1 = "On input string w:

1. Scan across the tape and reject if a 0 is found to the right of a 1.
2. Repeat if both 0s and 1s remain on the tape:
3. Scan across the tape, crossing off a single 0 and a single 1.
4. If 0s still remain after all the 1s have been crossed off, or if 1s still remain after all the 0s have been crossed off, reject . Other- wise, if neither 0s nor 1s remain on the tape, accept ."

分析结果：

1. 判断一个串，用 $O(n^2)$时间；
2. 使用的资源：单带图灵机。

同济大学 TONGJI UNIVERSITY

Analyzing Algorithms---*single-tape DTM*

M2 = “On input string w:

1.Scan across the tape and reject if a 0 is found to the right of a 1.

2.Repeat as long as some 0s and some 1s remain on the tape:

3. Scan across the tape, checking whether the total number of 0s and 1s remaining is even or odd. If it is odd, reject .
4. Scan again across the tape, crossing off every other 0 starting with the first 0, and then crossing off every other 1 starting with the first 1.

5.If no 0s and no 1s remain on the tape, accept. Otherwise, reject .”

分析结果：

1. 判断一个串，用 O(nlogn)时间；
2. 使用的资源：单带图灵机。
3. **This result can not be further improved on single-tape Turing machines. In fact, any language that can be decided in o(nlogn) time on a single-tape Turing machine is regular.**

同济大学 TONGJI UNIVERSITY

Analyzing Algorithms---*multi-tape DTM*

M3 = “On input string w:

1.Scan across tape 1 and reject if a 0 is found to the right of a 1.

2.Scan across the 0s on tape 1 until the first 1. At the same time, copy the 0s onto tape 2.

3.Scan across the 1s on tape 1 until the end of the input. For each 1 read on tape 1, cross off a 0 on tape 2. If all 0s are crossed off before all the 1s are read, reject .

4.If all the 0s have now been crossed off, accept . If any 0s remain, reject .”

分析结果：

1. 判断一个串，用 O(n)时间；
2. 使用的资源：多带图灵机。

同济大学 TONGJI UNIVERSITY

Analyzing Algorithms

Discussion

1. The complexity of A depends on the model of computation selected.
2. In complexity theory, we classify computational problems according to their time complexity. But with which model do we measure time?
3. Definition of the time complexity class.

DEFINITION 7.7

Let t: N→R^+ be a function. Define the ***time complexity class***, TIME(t(n)), to be the collection of all languages that are decidable by an O(t(n)) time Turing machine.

Recall the language $A = \{0^k1^k | k \geq 0\}$. The preceding analysis shows that $A \in TIME(n^2)$, because M_1 decides A in time $O(n^2)$ and $TIME(n^2)$ contains all languages that can be decided in $O(n^2)$ time.

同济大学 TONGJI UNIVERSITY

Model Dependency?

The previous result gives an O(n·log(n)) versus O(n) difference between 1 and 2-tape TMs.

上述问题在不同的机器上用不同的时间

Like with computability, we want the complexity of a problem to be independent of the specific TM-model that we use.

希望问题的复杂度 由问题本质而不是由TM确定

What to do?

Answer: We group $O(n^2)$, $O(n \cdot \log(n))$, $O(n)$, et cetera together in a ‘polynomial-time class’.

同济大学 TONGJI UNIVERSITY

Complexity relationships among models

THEOREM 7.8（多带与单带）

Let t(n) be a function, where t(n) ≥ n. Then every t(n) time multi-tape Turing machine has an equivalent $O(t^2(n))$ time single-tape Turing machine .

DEFINITION 7.9

Let M be a nondeterministic Turing machine that is a decider. The ***running time*** of M is the function f: N→N, where f(n) is the maximum number of steps that M uses on **any branch** of its computation on **any input** of length n.

THEOREM 7.11 （非确定性与确定性）

Let t(n) be a function, where t(n) ≥ n. Then every t(n) time nondeterministic single-tape Turing machine has an equivalent $2^{O(t(n))}$ time deterministic single-tape Turing machine.

同濟大學 TONGJI UNIVERSITY

The ‘Polynomial Time Class’ P

Definition 7.12: The class of languages that can be decided by a single-tape TM in polynomial time is denoted by **P**:

$$\mathbf{P} = \bigcup_{k=1,2,\ldots} \mathbf{TIME}(n^k)$$

P： 可用多项式时间在单带机上解决的问题的集合

Many, many problems are in **P**.
The problems in **P** are efficiently solvable.
一般认为 P 是易计算的一大类
Practically computable

同济大学 TONGJI UNIVERSITY

Robustness of P

In general, every kind of TM (k-tape, r-heads, etc.) can be solved by a standard, single tape TM with only polynomial time/space overhead.

As a result, the poly-time class **P** does not depend on the specific computational model that you use.

If some TM can solve A in poly-time, then $A \in \mathbf{P}$.

We can extend the TM-model in many ways…

同濟大學 TONGJI UNIVERSITY

Strong Church Turing Thesis

All reasonable computational models are polynomial-time/space equivalent:
It is always possible to simulate one model with a machine from another model with only polynomial time/space overhead.

The answer to the question "A∈**P**?" does not depend on the model that we favor.

P 和 Non P 是两个本质性的大类，与编程艺术无关

目前计算机上合理的可计算的问题：P类

同济大学 TONGJI UNIVERSITY

定理 7.12 $PATH \in P$

PATH={<G,s,t>|G is a directed graph that has a directed path from s to t }

思路

- 给出判定PATH的多项式时间算法。蛮力算法不够快。
- PATH的蛮力算法考察G中所有可能路径，来确定是否存在从s到t的有向路径。
- 可能路径就是G中长度最多为m的节点序列，m是G中的节点数。（如果从s到t存在有向路径，就存在长度不超过m的有向路径，因为路径上不需要重复节点。）
- 可能路径数是m^m，蛮力算法消耗指数时间。
- 多项式时间算法 设法避免蛮搜。
- 宽度优先搜索。连续标记G中从s出发，长度为1，2，3，直到m的有向路径可达的所有节点。很容易用多项式来界定该策略的运行时间。

同济大学 TONGJI UNIVERSITY

定理7.12 $PATH \in P$

证明： 思路：找亲戚—由近到远，作记号—避免重复

M = “对输入<G, s, t>，G是包含节点 s 和 t 的有向图：

1）在节点s上做标记。

2）复重下面步骤3，直到不再有节点被标记。

3）扫描G的所有边。如果找到一条边（a，b），a被标记而b没有标记，那么标记b。

4）若t 被标记，则接受；否则，拒绝。”

- 多项式时间。步骤1和4只执行一次。步骤3最多执行m次，除最后一次外，每一次执行都要标记G中的一个未标记的节点。所以用到的总步骤数最多是1+1+m，是G的规模**的多项式**。
- 步骤1和4很容易用任何合理的确定型模型在多项式时间内实现。步骤3需要扫描输入，检查某些节点是否被标记，这也容易在多项式时间内实现。所以M是$PATH$的多项式时间算法。

同济大学 TONGJI UNIVERSITY

定理7.13 $RELPRIME \in P$

RELPRIME = {<x, y> | x and y are relatively prime}.

思路　搜遍这两个数的所有可能的公因子，如果没有大于1的公因子，则互素 。

- 用k进制（k≥2）表示数，数字的大小是长度的指数倍。（如10进制的3位数，表示的最大数是10^4-1）蛮力算法需搜遍指数多个可能因子，消耗指数的运行时间。
- 改用**欧几里德算法(Euclidean algorithm)**，算最大公因子。gcd(x,y)，x和y互素的充要条件 gcd(x,y)=1。
- 欧几里德算法 用函数mod，x mod y　等于用y去整除x所得的余数。

同济大学 TONGJI UNIVERSITY

定理7.13 REL-PRIME∈P

证明：

先定义 子程序 E

E=“对输入<x,y>，x和y是二进制表示的自然数：

1）重复下面的操作，直到y=0

2） 赋值x←x mod y。

3） 交换x和y的值。

4）输出x。”

算法R以E为子程序求解REL-PRIME。

E的时间复杂性为多项式时间，

1. 步骤2的每一次执行（第一次可能例外），都把x的值至少减少一半。
2. 步骤2和3执行最大次数是min($2\log_2 x$, $2\log_2 y$)，与表示长度成正比，步骤的执行次数是O(n)。

同济大学 TONGJI UNIVERSITY

定理7.13 REL-PRIME∈P

总程序

***R*=“对输入<*x*,*y*>，*x*和*y*是二进制表示的自然数：**

1）在<*x*，*y*>上运行*E*。

2）若结果为1，就接受；否则，就拒绝。”

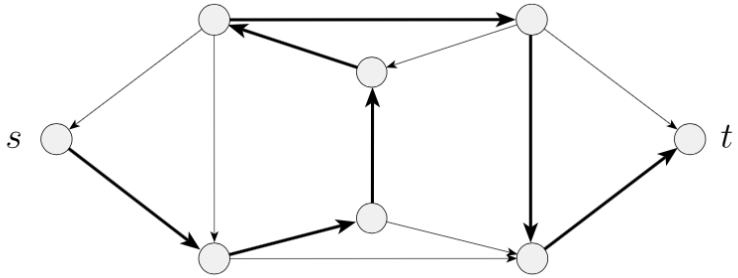
- *因为 **E** 多项式时间*
- 所以***R***也在多项式时间

同济大学 TONGJI UNIVERSITY

7.3 The Class NP

polynomial verifiability

HAMPATH ={⟨G, s, t⟩| G is a directed graph with a Hamiltonian path from s to t}.



Can it be solved in polynomial time ? better than Brute-force algorithm.

The *HAMPATH* problem has a feature called ***polynomial verifiability*** that is important for understanding its complexity.

怎么验证呢？凭什么验证？**Certificate(证书)**

In other words, *verifying* the existence of a Hamiltonian path may be much easier than *determining* its existence.

同济大学 TONGJI UNIVERSITY

7.3 The Class NP

Definition 7.18: A verifier for a language A is a program V such that we can write A as
A = {< ω > | V accepts <ω,c> for some c}
(The string c is the certificate that proves ω ∈A.)
验证机— 验证程序(如strcmp)，c — 证书(相当于摇奖结果）

EXP：***COMPOSITES* = {x| x = pq, for integers p, q > 1}.**

- For the *COMPOSITES* problem, a certificate for the composite number x simply is one of its divisors.
- For the *HAMPATH* problem, a certificate for a string ⟨G, s, t⟩ ∈ *HAMPATH* simply is a Hamiltonian path from s to t.
- In both cases, the verifier can check in polynomial time that the input is in the language when it is given the certificate.

同济大学 TONGJI UNIVERSITY

7.3 The Class NP

Definition 7.19: **NP** is the class of languages that have polynomial time verifiers.
彩票中奖：猜难，证易.

The term NP comes from ***Nondeterministic Polynomial time*** . Problems in NP are sometimes called NP-Problems.

Languages in **P** have poly-time deciders. 判定器
Languages in **NP** have poly-time ‘verifiers’. 验证器

同济大学 TONGJI UNIVERSITY

7.3 The Class NP

HAMPATH is NP ?

NTM N1 = "On input ⟨G, s, t⟩, where G is a directed graph with nodes s and t:

1. Write a list of m numbers, $p_1,...,p_m$, where m is the number of nodes in G. Each number in the list is nondeterministically selected to be between 1 and m.
2. Check for repetitions in the list. If any are found, *reject*.
3. Check whether $s = p_1$ and $t = p_m$. If either fail, *reject*.
4. For each i between 1 and m-1, check whether (p_i, p_{i+1}) is an edge of G. If any are not, *reject*. Otherwise, all tests have been passed, so *accept* ."

同濟大學 TONGJI UNIVERSITY

7.3 The Class NP

THEOREM 7.20

A language is in NP iff it is decided by some nondeterministic polynomial time Turing machine.

PROOF IDEA We show how to convert a polynomial time verifier to an equivalent polynomial time NTM and vice versa. The NTM simulates the verifier by guessing the certificate. The verifier simulates the NTM by using the accepting branch as the certificate.

实质：NTM与DTM之间的转换

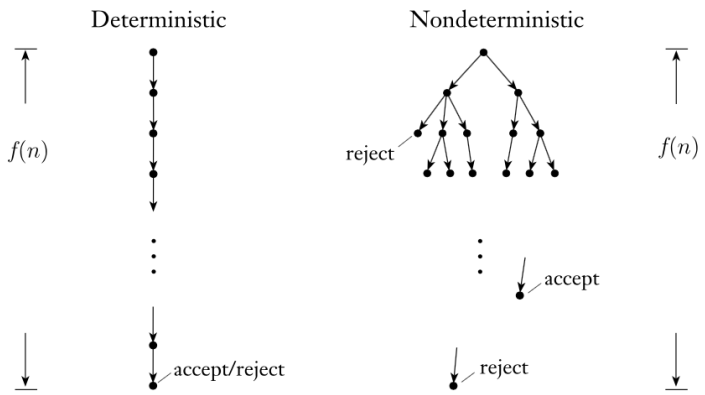


FIGURE 7.10
Measuring deterministic and nondeterministic time

同济大学 TONGJI UNIVERSITY

7.3 The Class NP

PROOF let A ∈ NP , Let V be the polynomial time verifier. Assume that V is a TM that runs in time n^k and construct NTM N (calls V)

长度为n^k的串只有有限个。排序后易枚举

N = "On input w of length n:
1. Nondeterministically select string c of length at most n^k.
2. Run V on input ⟨w, c⟩.
3. If V accepts, accept ; otherwise, reject ."

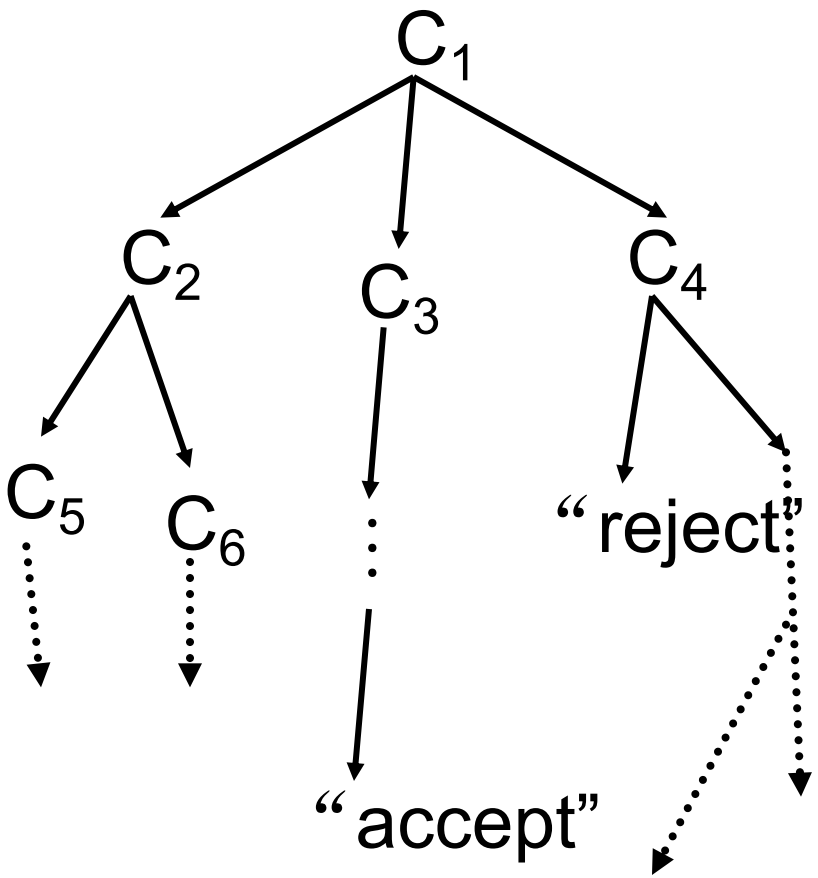
To prove the other direction of the theorem, assume that A is decided by a polynomial time NTM N and construct a polynomial time verifier V as follows.

V = "On input ⟨w, c⟩, where w and c are strings:
1. Simulate N on input w, treating each symbol of c as a description of the nondeterministic choice to make at each step (as in the proof of Theorem 8.2 NTM==DTM**).**
2. If this branch of N 's computation accepts, accept ; otherwise, reject ."

DTM模拟NTM中的Adress磁带

同济大学 TONGJI UNIVERSITY

Variants of T（复习）



Proof IDEA: DTM D simulates NTM N;

1.N’s computation on input w as a tree;
2.A branch of tree is one branch of the N’s computation;
3.A node of the tree corresponds to a configuration.
4.TM D search the tree for an accepting configuration, in the breadth-first search rather than depth-first search.
5.Every nondeterministic TM has an equivalent 3-tape Turing machine, which –in turn– has an equivalent 1-tape Turing machine.

同济大学 TONGJI UNIVERSITY

Variants of T（复习）

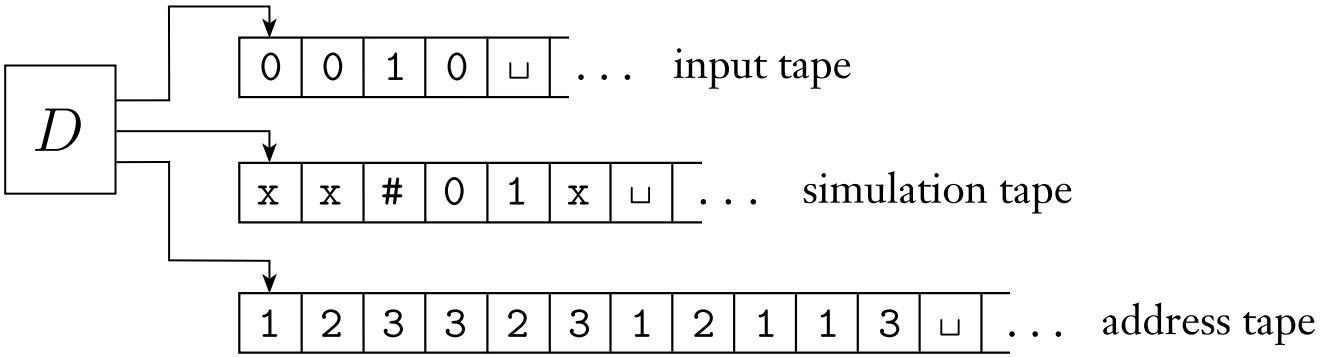
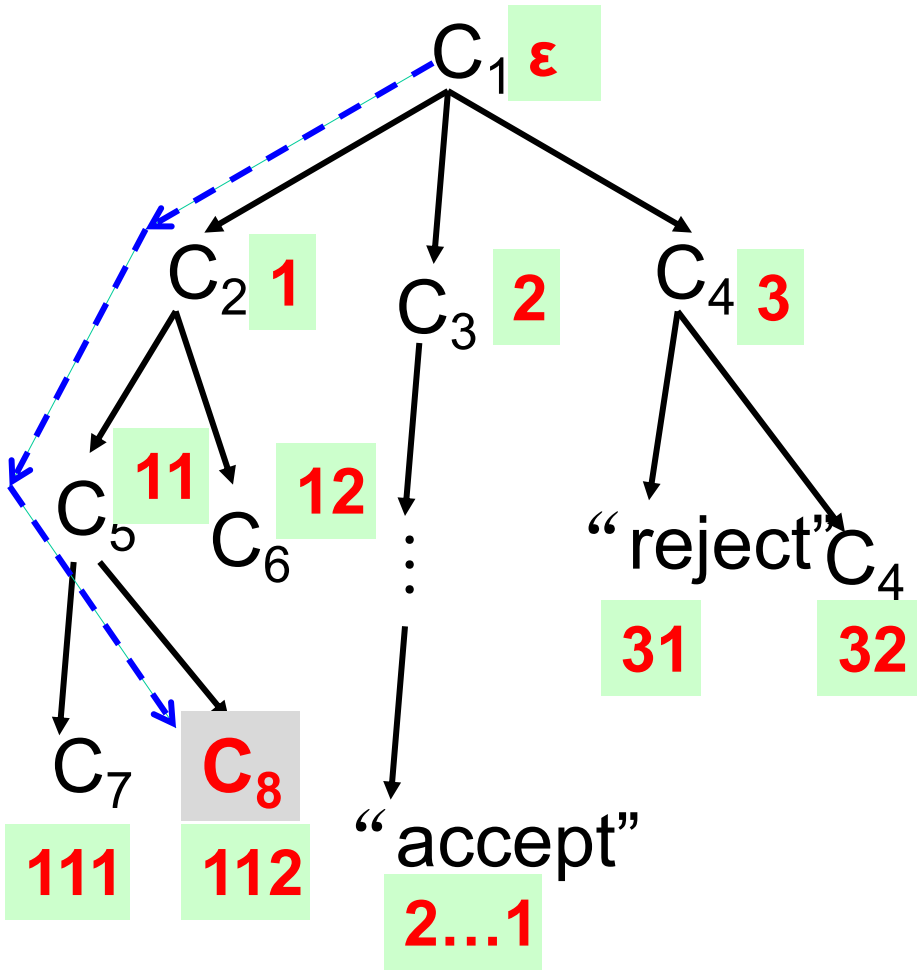


FIGURE 3.17
Deterministic TM D simulating nondeterministic TM N

第1带 保存输入 供多次扫描 (不确定，可后悔）
第2带 用于计算 （模拟一个CPU, 一条路线）
第3带 协调调度 不确定过程多个CPU的调度、树搜索与回溯的当前位置，辅助第2带工作。

同济大学 TONGJI UNIVERSITY

Variants of T（复习）



Schematic of the node's address

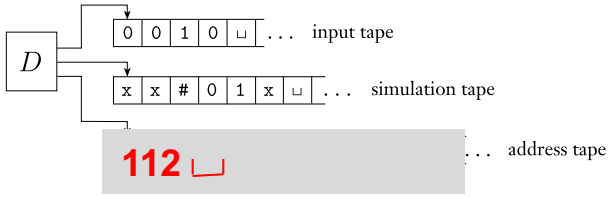


FIGURE 3.17
Deterministic TM D simulating nondeterministic TM N

Proof:

1.Initially, tape 1 contains the input ω,tape 2 and 3 are empty;

2.Copy tape 1 to tape 2;

3.Use tape 2 simulate N with input ω on one branch of its nondeterministic computation……

4. Replace the string on tape 3 with the next string in the string ordering. Simulate the next branch of N's computation by going to stage 2.

同济大学 TONGJI UNIVERSITY

7.3 The Class NP

DEFINITION 7.21

NTIME***(t(n))*** = {L| L is a language decided by an O(t(n)) time nondeterministic Turing machine}.

Corollary 7.22: 用NTIME(n^K)表示被NTM在O(n^K)时间内判定的语言族，则

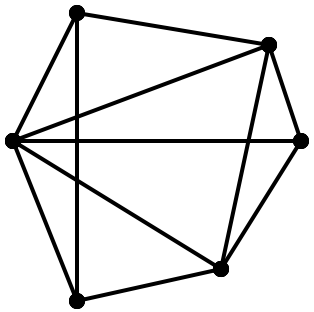
$$\mathrm{NP} = \bigcup_k \mathrm{NTIME}(n^k)$$

通常，证书比较直观（例如 兑奖，strcmp即可）
验证证书时只需正面验证 x∈A，反面不必验证 x∉A.

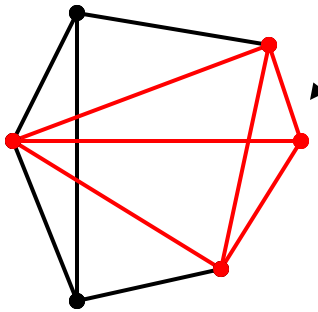
同济大学 TONGJI UNIVERSITY

7.3 The Class NP

- 团—图中的小集团， 两两 有勾结 的顶点集合
- Consider a graph G, is there a k-clique?



graph



4-clique

but no 5-clique

腐败分子常常拉帮结派，团团伙伙...

团问题
CLIQUE = {<G,k> | graph G has a k-clique}

同济大学 TONGJI UNIVERSITY

7.3 The Class NP

THEOREM 7.24 *CLIQUE* is in NP.

PROOF IDEA The clique is the certificate.

PROOF The following is a verifier V for *CLIQUE*.

V ="On input ⟨ ⟨ G , k ⟩ , c⟩:
1.Test whether c is a sub-graph with k nodes in G.
2.Test whether G contains all edges connecting nodes in c.
3.If both pass, accept; otherwise, reject."

ALTERNATIVE PROOF If you prefer to think of NP in terms of nondeterministic polynomial time Turing machines, you may prove this theorem by giving one that decides *CLIQUE*. Observe the similarity between the two proofs.

N = "On input ⟨G, k⟩, where G is a graph:
1.Nondeterministically select a subset c of k nodes of G.
2.Test whether G contains all edges connecting nodes in c.
3.If yes, accept ; otherwise, reject ."

同濟大學 TONGJI UNIVERSITY

7.3 The Class NP

THEOREM 7.25 SUBSET-SUM is in NP . 子集和 问题

问题的直观实例 (1)不找购物 (2) 装背包，又称背包问题

$\langle \{2,4,8\}, 10 \rangle \in \text{SUBSET-SUM}$ … because 2+8=10

$\langle \{2,4,8\}, 11 \rangle \notin \text{SUBSET-SUM}$

… because 11 cannot be made out of {2,4,8}

问题的形式化描述：

$$\text{SUBSET-SUM} = \{\langle S, t \rangle \mid S = \{x_1, \ldots, x_k\}$$

there is a subset $R \subseteq S$

such that $\sum_{y \in R} y = t\}$

商品价格
口袋中零钱

同济大学 TONGJI UNIVERSITY

7.3 The Class NP

THEOREM 7.25 SUBSEYT-SUM is in NP . 子集和问题

问题的直观实例 (1)不找零购物 (2) 装背包，又称背包问题

$\langle \{2,4,8\}, 10 \rangle \in \text{SUBSET-SUM}$ … because 2+8=10

$\langle \{2,4,8\}, 11 \rangle \notin \text{SUBSET-SUM}$

… because 11 cannot be made out of {2,4,8}

问题的形式化描述：

$$\text{SUBSET-SUM} = \{\langle S, t \rangle \mid S = \{x_1, \ldots, x_k\}$$

there is a subset $R \subseteq S$

such that $\sum_{y \in R} y = t\}$

100元 /整钱
小商品集合

凑足价值100元 的小商品

同济大学 TONGJI UNIVERSITY

7.3 The Class NP

PROOF IDEA The subset is the certificate.

PROOF The following is a verifier V for *SUBSET-SUM*.

V ="On input ⟨ ⟨S , t⟩, c⟩:

1.Test whether c is a collection of numbers that sum to t.

2.Test whether S contains all the numbers in c.

3.If both pass, accept; otherwise, reject."

ALTERNATIVE PROOF We can also prove this theorem by giving a nondeterministic polynomial time Turing machine for *SUBSET-SUM* as follows.

N = "On input ⟨S, t⟩:

1.Nondeterministically select a subset c of the numbers in S.

2.Test whether c is a collection of numbers that sum to t.

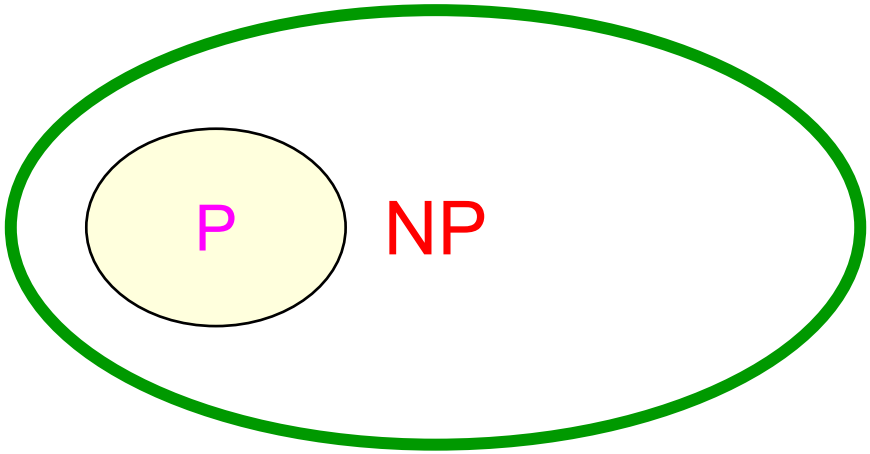
3.If the test passes, accept ; otherwise, reject ."

同濟大學 TONGJI UNIVERSITY

7.3 The Class NP

P=NP? 还不知道 （coNP = NP） ？

- 未解决问题
- 有一种夸张说法
- 如果解决了,全世界计算机科学家可放假三天



同济大学 TONGJI UNIVERSITY

7.4 NP-COMPLETENESS

关于NPC

- NPC 即，最难缠的一群NP，他们“拉邦结伙、互相规约、同衰共荣”, 有“繁衍能力”
- 多伦多大学，Cook教授,1972 年 证明 3SAT in NPC
- 3SAT作为种子，繁衍了上千个著名NPC
- NP完全性，具有重要的理论和实践意义
- 本节重点：
  - ① 证明SAT, 3SAT in NPC；
  - ② 通过规约，由3SAT繁衍其它NP问题；自学

同济大学 TONGJI UNIVERSITY

从 Boolean formula 到 SAT

Boolean variable x can be TRUE or FALSE, which is also denoted by "1" or "0". 真，假

Boolean operations are AND ($x \wedge y$), OR ($x \vee y$), and NOT ($\neg x$ or also $\bar{x}$). 与 或 非

Boolean formula: $\phi = (\neg x \vee y) \wedge (x \vee \neg y)$.

问题：Can we design a Boolean formula to *simulate* a Turing machine?

A Boolean formula may contain the Boolean operations AND, OR, NOT, and these operations form the basis for the circuitry used in electronic computer. So……

同济大学 TONGJI UNIVERSITY

SAT---Satisfiability Problem

A Boolean formula is **satisfiable** (可满足的), if some assignment （赋值） to the variables makes the formula evaluate to 1.

EXP: $\phi = (\neg x \vee y) \wedge (x \vee \neg y)$;

Assignment: x=y=TRUE , x=y=FALSE

The satisfiable problem is to test whether a Boolean formula is satisfiable.

SAT = {< ϕ > | ϕ is a satisfiable Boolean formula}

判定ϕ是否属于SAT，需要用多少时间呢？P，NP or NPC?

结论是 **SAT is NP-complete**。证明很复杂，Cook已经证明。

同济大学 TONGJI UNIVERSITY

SAT ⟶ 3SAT

Literal：如 (x)，(¬x) .文字（单项）
Clause = 析取的文字，如 (x ∨ ¬y ∨ z)， 子句
A formula ϕ is in conjunctive normal form (CNF),
if it is the AND of clauses. For example: 合取范式

$$\phi(x,y,z) = (x \vee \bar{y}) \wedge (z) \wedge (\bar{z} \vee y \vee \bar{y})$$

3CNF has only clauses with 3 literals: 3合取范式

3SAT = {<ϕ> | ϕ is a satisfiable 3CNF formula}

$$\phi(x,y,z) = (x \vee \bar{y} \vee \bar{z}) \wedge (z \vee z \vee z) \wedge (\bar{z} \vee y \vee \bar{y})$$

括号内：析取

括号间：合取

同济大学 TONGJI UNIVERSITY

Poly-Time Functions 多项式间函数

Definition 7.28
A function $f:\Sigma^* \rightarrow \Sigma^*$ is a poly-time-computable function if some polynomial time Turing machine exists that halts with f(w) on the tape, when started on any input w. （强调函数f的时间复杂度）

+,- * / are all poly-time. 算术运算

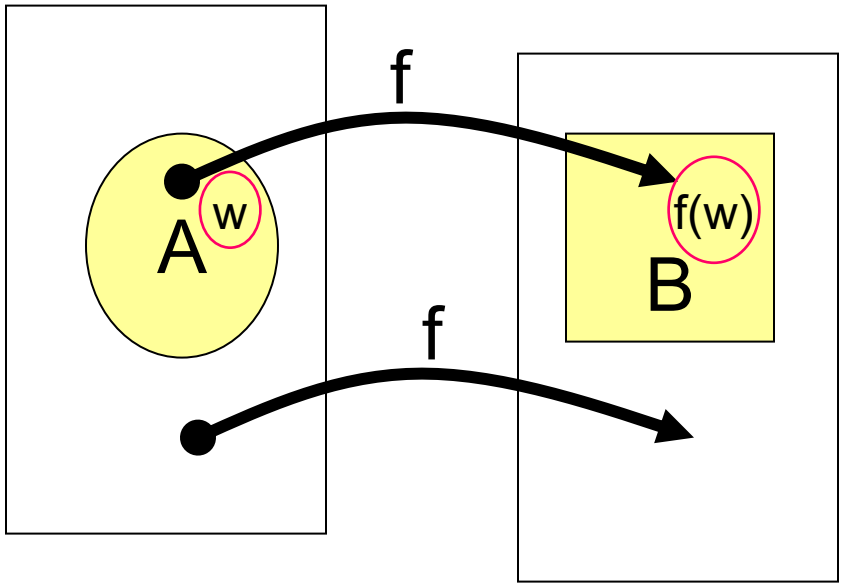
Important here is that object transformations,对象变换 like "Given a formula ϕ, make a graph G_ϕ" can almost always be done in **poly-time.**

同济大学 TONGJI UNIVERSITY

Polynomial Time Reducible

定义7.29 A language A is polynomial time reducible to another language B if there is a polynomial time computable function $f:\Sigma^*\to\Sigma^*$ such that: $w\in A \Longleftrightarrow f(w)\in B$ for every $w\in\Sigma^*$.

- Terminology/notation: $A \leq_P B$
- 类似映射规约$A \leq_M B$
- 应用：if $A \leq_P B$，and B is NPC，then A is NPC.（即成员检测）



同济大学 TONGJI UNIVERSITY

P obeys $\leq_P$ Ordering　P 遵守$\leq_P$归约次序

Theorem 7.31:
If $A \leq_P B$ and B is in P, then A is in P as well.

Proof: Let M be the poly-time TM for B and f is the reducing function from A to B. Consider the TM:

S= "On input w:

1. Compute f(w)
2. Run M on f(w) and give the same output."

分析：

1. 根据定义，计算f（w）需要多项式时间。
2. 第2步的前半部分和后半部分都是多项式时间完成。
3. 所以，S的时间复杂度也是多项式的。

同济大学 TONGJI UNIVERSITY

“CLIQUE = SAT = 3SAT” 复杂度等价

思路：团问题=满足问题=3满足问题

We will prove that the languages CLIQUE, SAT and 3SAT are all equally complex when measured with the polynomial time measure stick.

First we will prove that 3SAT is not more difficult than CLIQUE:

3SAT可转化为团问题

于是

“If you know how to solve CLIQUE in polytime, then you also know how to solve 3SAT in polytime”

同济大学 TONGJI UNIVERSITY

3SAT $\leq_P$ CLIQUE

THEOREM 7.32

3SAT is polynomial time reducible to CLIQUE.
(即 3SAT $\leq_P$ CLIQUE)

Idea for proof :

1. 3SAT is set of 3cn-formula ϕ , CLIQUE is a set of graph G with a k-CLIQUE.
2. The polynomial time reduction f converts the formula ϕ to graph G.
3. The formula ϕ is satisfialbe iff G has k-clique;

总之，问题的关键是，如何根据k个子句的公式，构造对应的k-团，且满足上述的第3点要求。 下页举例说明

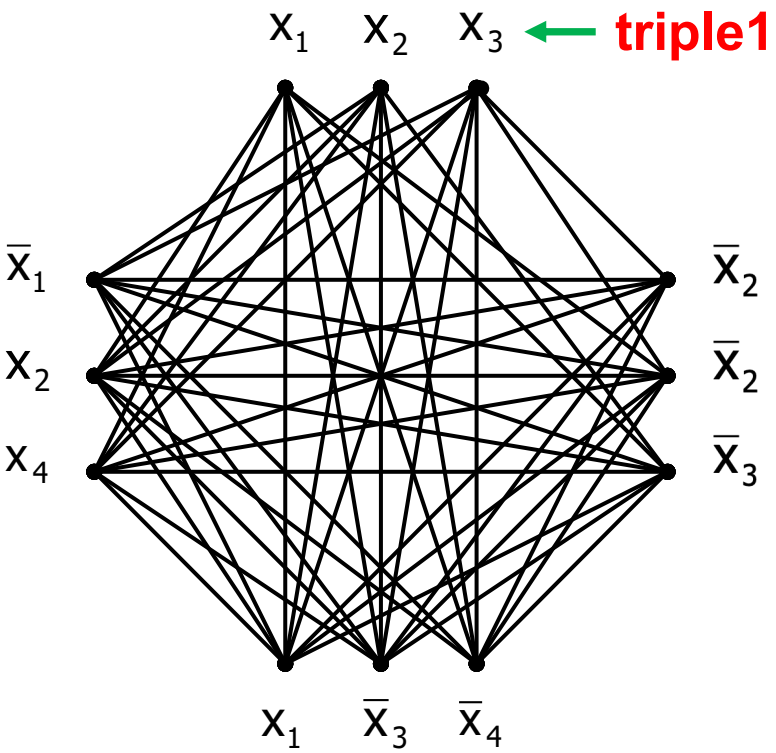
同济大学 TONGJI UNIVERSITY

Example 3SAT to CLIQUE

- **Formula (4 clauses, 4 variables): 4变量的3子句归约为团**

$$(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_3 \vee \bar{x}_4)$$

clause1



由3cnf公式构造对应图的方法：

① k个子句对应k个triple，每个triple包括3个顶点；

② 按子句中变量名称标记顶点；

③ 连接不在同一个triple中，且互不矛盾的任意两顶点。比如，$(x_1, \bar{x}_1)$是相互矛盾的。

上述3步工作可在多项式时间内完成，即上述算法是多项式时间规约。

同济大学 TONGJI UNIVERSITY

3SAT → CLIQUE. 已经完成P-时间映射归约

当公式为真时，每个3-合取范式中至少一个变量为真，设红箭头所指为真

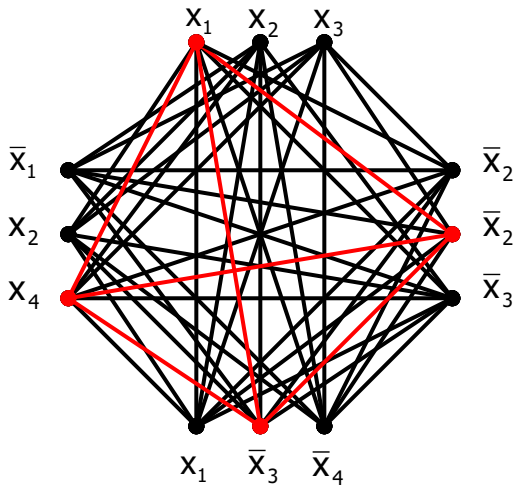
$$(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_3 \vee \bar{x}_4)$$

$x_1, \bar{x}_2, \bar{x}_3, x_4$

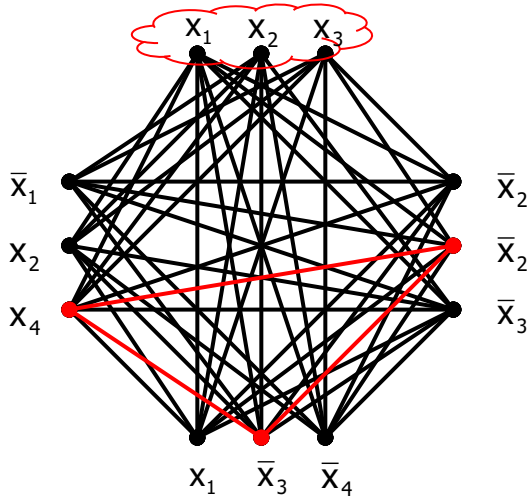
均True, 是一个满足的赋值，对应顶点成为团

$\bar{x}_1, \bar{x}_2, \bar{x}_3, x_4$

均True, 是个不满足的赋值



… 有 4-clique



… 非 4-clique

同济大学 TONGJI UNIVERSITY

3SAT $\leq_p$ CLIQUE

Now, we show that:
The formula ϕ is satisfialbe if and only if G has k-clique

证明过程略，参见教材**p303.**

结论:

$\phi \in$ 3SAT $\leftrightarrow$ G $\in$ k-CLIQUE (with k the number of clauses of ϕ).

注意:

映射 f: $\phi \rightarrow$ G 是 p-时间可计算的。

同济大学 TONGJI UNIVERSITY

7.4.2 NP-Completeness NP-完全

Definition 7.34: Language B is **NP-complete** if it satisfies two conditions:

1. B is in NP, and
2. every language A∈**NP** we have $A \leq_P B$.

只缘身在此山中

- B有繁衍力，
- 有吸引性

直观：**NP**完全 （ **NP**-彻底，**NP**-中坚）

NP-complete problems are the most difficult problems in **NP**…

同济大学 TONGJI UNIVERSITY

NP-Completeness

补充归纳： 关于NPC的4性质

- A,B in NPC, 则$A \leq_P B$ 可互相规约
- A in NPC 如果 A in P， 则NP=P
- A in NPC 如！(A in P） 则 NPC∩ P= Φ
- A in NPC， B in NP ， $A \leq_P B$ 则 B in NPC

直 观：NPC 即:最难缠的一群NP，他们 “拉邦结伙、互相规约、同衰共荣”, 有“繁衍能力”

多伦多大学，Cook教授,1972 年 证明 3SAT in NPC
3SAT作为种子，繁衍了上千个著名NPC

同济大学 TONGJI UNIVERSITY

NP-Completeness

NP-Hard问题

If we omit requirement 1, then we sometimes say that B is **NP**-hard.

只知道有**繁衍力**，
不一定身在此山中

NP-Hard问题要比 NPC问题的范围广，难找到多项式的算法，不一定是NP问题。放宽了条件，可能比NPC更难

有类问题是软NP(疑似NP)—普通人用NP时间解决。专家可设计好算法在 DP-时间解决，如2SAT问题。

56同济大学 TONGJI UNIVERSITY

NP-Completeness

THEOREM 7.35

If B is NP-complete and B ∈ P, then P = NP.

Proof: If A∈**NP**, then by the definition of **NP**-completeness: $A \leq_P B$.
From B∈**P**, it follows that also A∈**P**.

THEOREM 7.36

If B is NP-complete and $B \leq_P C$ for C in NP, then C is NP-complete.

同濟大學 TONGJI UNIVERSITY

The COOK-LEVIN THEOREM 库克-列文定理

Theorems 7.37

SAT is **NP**-complete problem

根据定义**7.34**：

1) SAT in NP，已经被证明。

只需证明:

2) every A∈**NP** we have A≤$_P$SAT.

证明 任何 **NP**问题→ 可满足性问题**SAT**

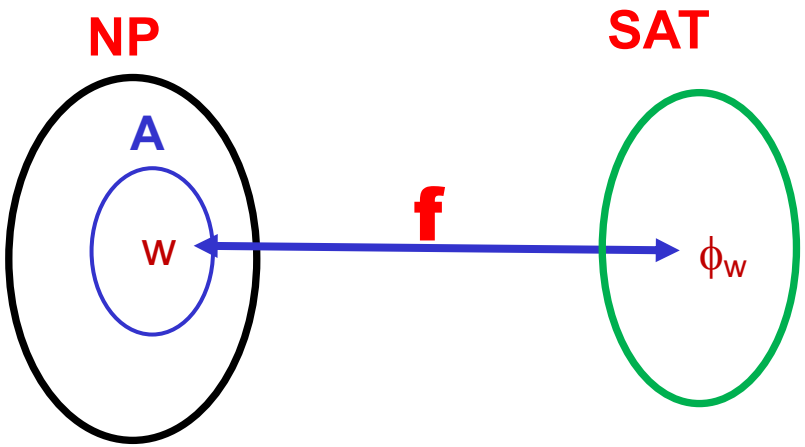
*任何*二字是本定理的威力和难点

在多项式时间内规约

同济大学 TONGJI UNIVERSITY

Proof Idea

目标：证明对于∀A∈NP，有A≤$_P$B。如下图所示：



1. **SAT**={<ϕ_w> | ϕ_w is a satisfiable Boolean formula}
2. **f**是多项式时间的规约函数，输入字符串，输出布尔公式。
3. ∵ A∈NP ∴∃ a Nondeterministic Polynomial time turing Machine(简称NP machine) **N** decides （判定）A, on input w.
4. **Key Idea**:模拟 NP **N**，构造规约函数**f**。

同济大学 TONGJI UNIVERSITY

Proof Idea

那么，如何构造规约函数呢？

把A 多项式规约到SAT

对每个 w, 构造合取范式(CNF) ϕ_w 使

$w \in A \Longleftrightarrow \phi_w \in SAT$,

并且从w计算ϕ_w 只需要 poly-time

换言之，

Let N be the nondeterministic **NTM** that accepts A.

our Key idea:

$w \in A \Longleftrightarrow \exists$ accepting path of N on w

$\Longleftrightarrow \exists$ 变量$x_1 \ldots x_m$， 使得$\phi_w(x_1 \ldots x_m)$ = TRUE

不确定机中有个派生路径接受w

iff

有满足合取式的幅值

同济大学 TONGJI UNIVERSITY

More Proof Outline

设N是 P-时间的接受A的 NTM，

Specifically we will establish the chain:

$w \in A \Longleftrightarrow \exists$ accepting path of N on w 派生路径

$\Longleftrightarrow$ 某个合取式被满足

There exists a sequence $C_1,\ldots,C_T$ of 格局序列 configurations with:

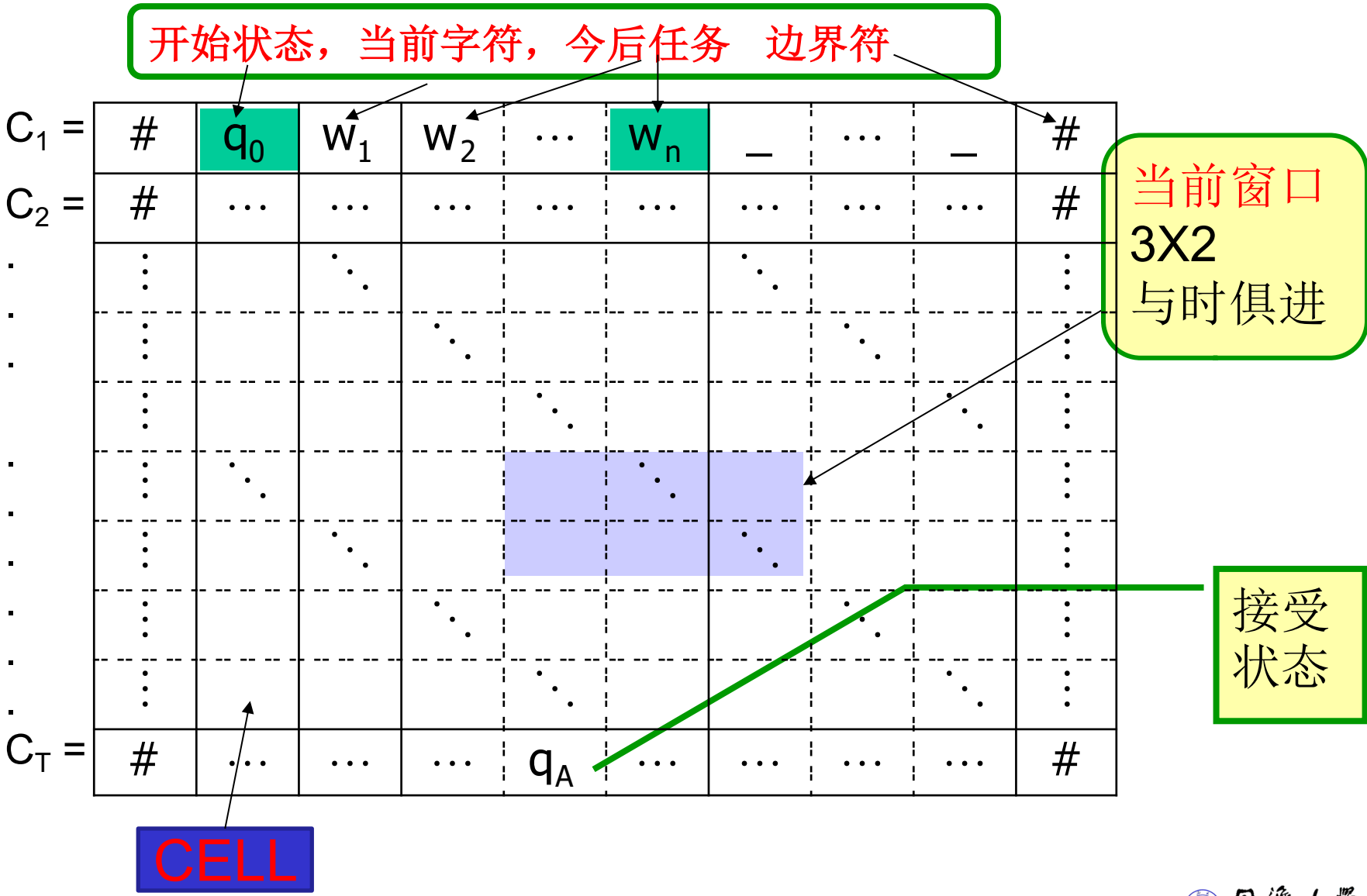
- C_1 开始格局 the start configuration of N on w
- (C_j,C_{j+1}) a proper N transition for every j 格局转换
- C_T an accepting configuration 接受格局

$\Longleftrightarrow \exists x_1 \ldots x_m$， 使得$\phi_w(x_1 \ldots x_m)$ = TRUE

（合取式被满足）

同济大学 TONGJI UNIVERSITY

A Tableau is an $n^k \times n^k$ table of configurations



同济大学 TONGJI UNIVERSITY

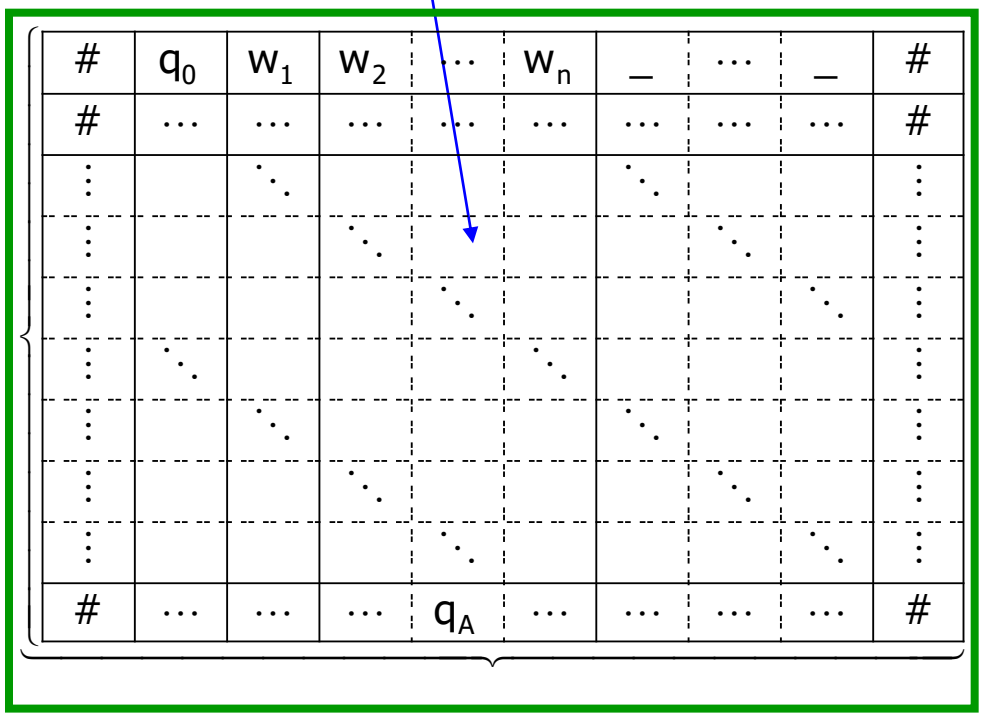
Size of Path of N on w

设N 是多项式时间 接受A的不确定图灵机，输入长为n时，接受路径有限长:设输入长度为n，演算步数不超过$O(n^k)$.

把格局序列填入 tableau （演算表格）of $n^k \times n^k$ 单元,

多项式时间 可完成填表工作 单元格cell(i,j)

下页用合取式 $\phi_w(x_1 \ldots x_m)$ 描述它 tableau…

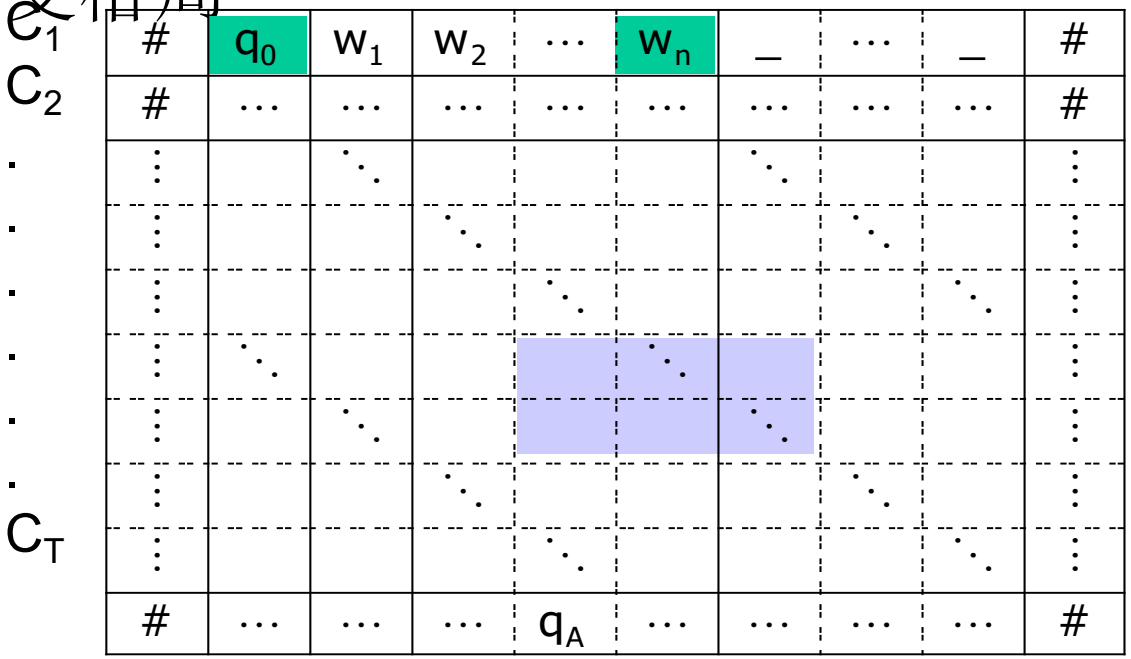


同济大学 TONGJI UNIVERSITY

How ϕ Describes the Tableau

用m个布尔变量作 合取式 $\phi_w(x_1 \ldots x_m)$， 设法

1. 合理描述单元，
2. C_1 是开始格局 （ N--NTM机， 输入w）
3. 格局转移 (C_j, C_{j+1})合法
4. C_T 是接受格局



同濟大學 TONGJI UNIVERSITY

How ϕ Describes the Cells

观察：填表符号 和 图灵机有关联。

为 C=Q∪Γ∪{#} ，其中Q状态集, Γ带符号集合, #为边界符

技巧 用单元内容定义合取式中变量 如下：

$$x_{(i,j,s)} = \begin{cases} \text{TRUE if cell}(i, j) = s \\ \text{FALSE otherwise} \end{cases}$$

$s \in Q \cup \Gamma \cup \{\#\}$

布尔变量总数： $n^k \times n^k \times |Q \cup \Gamma \cup \{\#\}|$

完成上述构造工作用时间 $O(n^{2k})$

polynomial in n. （k是常数, n是输入的长度）

同济大学 TONGJI UNIVERSITY

函数ONE(...)

Example: Describe in CNF the Boolean function

构造一个合取式函数，当恰有一个变量为真时，函数取真，其余为假（这个函数后面有用）
(单因子构件技术)

$$ONE(x_1,\ldots,x_n)=\begin{cases}1 \text{ if there is exactly one "true" } x_j\\ 0 \text{ otherwise}\end{cases}$$

需求定义

构造定义

至少一个为真

至多一个为真

Answer:

$$ONE(x_1,\ldots,x_n)=\left(\bigvee_{j=1}^{n} x_j\right)\wedge\left(\bigwedge_{j\neq k}^{n}(\bar{x}_j\vee\bar{x}_k)\right)$$

同济大学 TONGJI UNIVERSITY

ϕ_{cell} for Proper Cells

1. 并非所有赋值有意义.
2. 每单元格cell(i,j) 有一个描述.
   对 $1 \le i,j \le n^k$ 只有一个变量$x_{(i,j,s)}$ 设为 TRUE.
3. 符号 s 的个数的范围： 1 ～ $|Q \cup \Gamma \cup \{\#\}|$.
4. 对每单元造ONE函数，把n^k个单元的ONE函数全部作交运算

$$\phi_{cell} = \bigwedge_{i,j=1}^{n^k} ONE(x_{(i,j,1)}, \cdots, x_{(i,j,c)})$$

同济大学 TONGJI UNIVERSITY

ϕ_{start} for the Start Configuration

比较复杂，一个个看：

图灵机 N的开始状态是 q_0, 输入串 $w_1,\ldots,w_n$.

其它地方填入 _表示空白和左右边界#s

所以 开始状态对应的 公式为：

它为真 表示 开始格局正确 (下页有图）

1行，n+2列，输入字符W_n

$$\phi_{start} = X_{(1,1,\#)} \wedge X_{(1,2,q_0)} \wedge$$

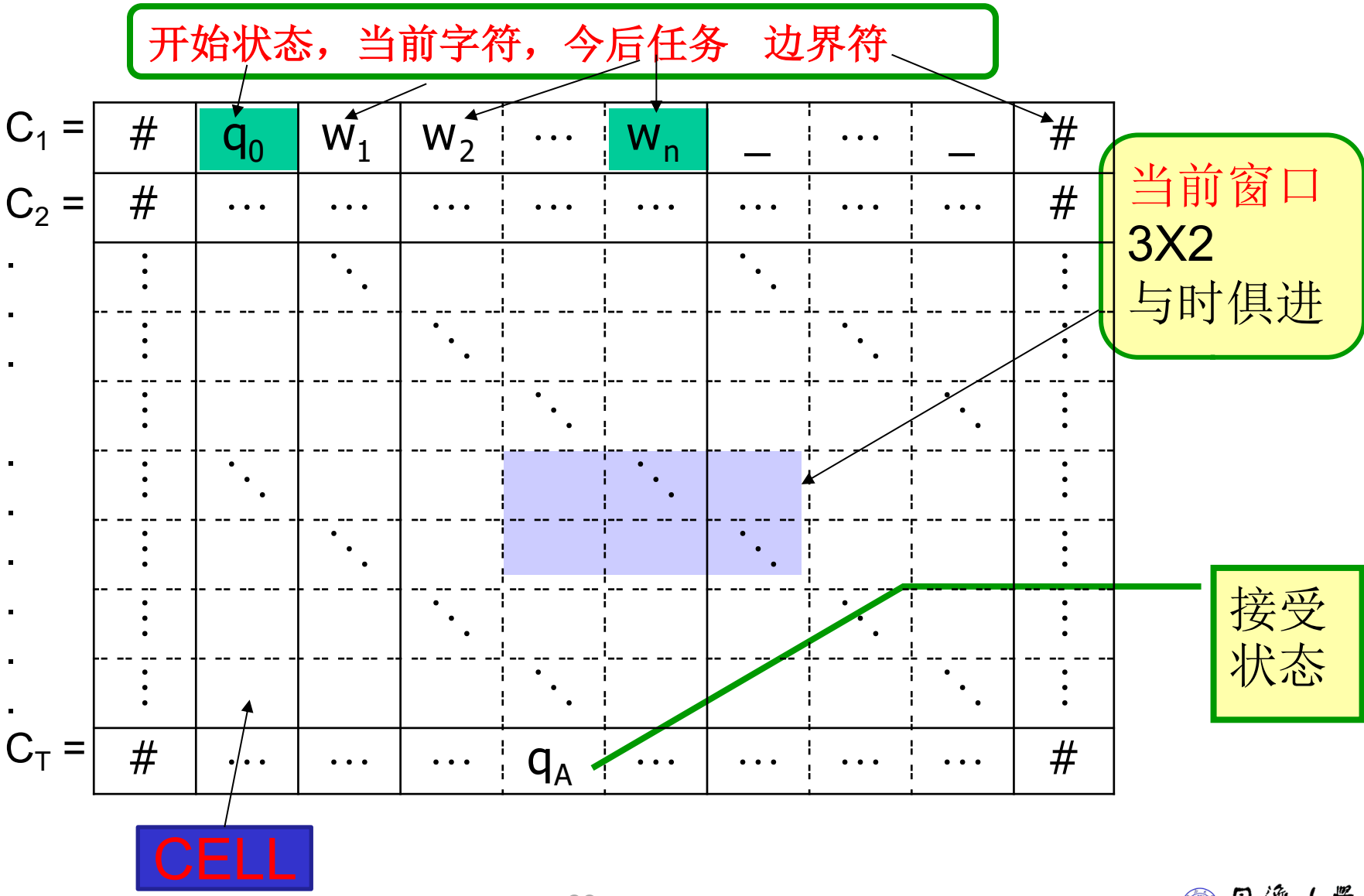
$$X_{(1,3,w_1)} \wedge \cdots \wedge X_{(1,n+2,w_n)} \wedge$$

$$X_{(1,n+3,\_)} \wedge \cdots \wedge X_{(1,n^k-1,\_)} \wedge X_{(1,n^k,\#)}$$

所有第3维下标连接起来为开始格局

同济大学 TONGJI UNIVERSITY

Accepting Path of N on w



同济大学 TONGJI UNIVERSITY

ϕ_{accept} for Accepting

当图灵机N 进入接受态q_{accept} 时，检查表中最后一行是否包含终止状态。
它对应的函数为真 表示 接受 (下页有图）

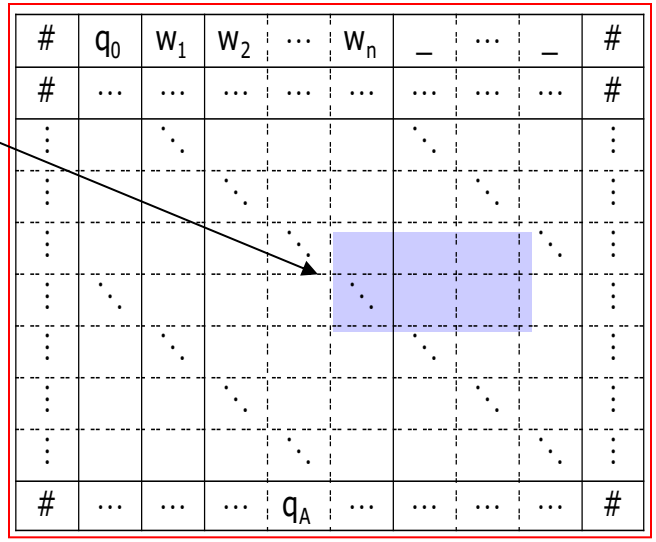
$$\phi_{accept} = \bigcup_{j=1}^{n^k} x_{(n^k, j, q_{accept})}$$

同济大学 TONGJI UNIVERSITY

ϕ_{move} for Proper Transitions

起止已经完成。中间格局序列 $C_1, C_2, \ldots, C_T$ 应符合图灵机 N的转移函数.

局部检查所有的 2×3 窗口.



这种窗口个数为
$(n^k-1)\times(n^k-2)$
下列函数取真 表示全部合格

$$\phi_{move} = \bigwedge_{i=1}^{n^k-1} \bigwedge_{j=1}^{n^k-2} (i,j) \text{ window legal}$$

同濟大學 TONGJI UNIVERSITY

Legal Windows

窗口中无状态符，表示读写头不在此窗口，带符号应该不变。所以 for all a,b,c∈Γ∪{#}.

a	b	c
a	b	c

不确定TM,可能多个下一状态
读写头右移
左动

对转移 $\delta(q_1,a) = \{(q_2,b,R), (q_3,c,L)\}$:

a	q_1	a
a	b	q_2

，

a	q_1	a
q_3	a	c

，以此类推

More Legal Windows

读写头在全局移动， 局部窗口可能看不到全部情况
这些是合法的窗口:

a	b	c
a	b	q_2

b	a	q_4
b	a	c

b	a	c
q_7	a	c

等等.

一旦进入接受状态后 图灵机 就不再变化:
此窗口合法:

a	q_{accept}	b
a	q_{accept}	b

同济大学 TONGJI UNIVERSITY

Some Illegal Windows

无状态

两状态

边界问题

下列窗口 表明格局序列有错误，这些窗口不合法

a	b	c
a	b	a

b	a	q_4
b	q_2	q_1

#	q_2	c
q_6	a	c

下一状态不正确

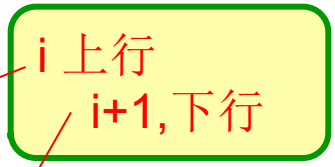
a	q_4	b
a	c	q_9

要点：如果 $(q_9,c,R) \notin \delta(q_4,b)$
即违反了图灵机 转换函数，
则不合法

同济大学 TONGJI UNIVERSITY

(i,j) Window Legal…

设 6格组 $L \subset (Q \cup \Gamma \cup \{\#\})^6$ 是合法窗口集合则 “以(i,j)为左上角的3X2窗口合法” 可以表为



$$\bigvee_{(s_1,\ldots,s_6)\in L} (x_{(i,j,s_1)} \wedge x_{(i,j+1,s_2)} \wedge \cdots \wedge x_{(i+1,j+2,s_6)})$$

下列公式取真，表示所有转移 正确 表达:

$$\phi_{move} = \bigwedge_{i=1}^{n^k-1} \bigwedge_{j=1}^{n^k-2} (i,j) \text{ window legal}$$

同濟大學 TONGJI UNIVERSITY

The Complete ϕ_w Formula

把上述4个条件合取:

$$\phi_w = \phi_{cell} \wedge \phi_{start} \wedge \phi_{move} \wedge \phi_{accept}$$

根据构造过程， ϕ_w 被满足←→图灵机M在输入w存在一个接受路径，即：

$$w \in A \Longleftrightarrow \phi_w \in SAT$$

归约已经完成，
还要证明，映射规约 $w \rightarrow \phi_w$ is poly-time…

同济大学 TONGJI UNIVERSITY

Polytime Reduction Check 1

设输入为 $w_1,\ldots,w_n$ 长度为 n，图灵机 N 的时间复杂度 $O(n^k)$

我们造的布尔表达式ϕ_w中变量$x_{(i,j,s)}$总数为 $O(n^{2k})$

问题: 能在多项式时间内描述 ϕ_w？
四部分逐一检查：

$$\phi_{cell} = \bigwedge_{i,j=1}^{n^k} ONE(x_{(i,j,1)},\ldots,x_{(i,j,c)})$$

$O(n^{2k})$ time.

同济大学 TONGJI UNIVERSITY

Polytime Reduction Check 2

Can we describe ϕ in poly(n) time?

$$\phi_{start} = x_{(1,1,\#)} \wedge x_{(1,2,q_0)} \wedge$$
$$x_{(1,3,w_1)} \wedge \cdots \wedge x_{(1,n+2,w_n)} \wedge$$
$$x_{(1,n+3,\_)} \wedge \cdots \wedge x_{(1,n^k-1,\_)} \wedge x_{(1,n^k,\#)}$$

… requires $O(n^k)$ time. 多项式时间可完成构造

同濟大學 TONGJI UNIVERSITY

Polytime Reduction Check 3

- Can we describe ϕ in poly(n) time?

$$\phi_{move} = \bigwedge_{i=1}^{n^k-1} \bigwedge_{j=1}^{n^k-2} \text{(i, j) window legal}$$

… requires $O(n^{2k})$ time.多项式时间可完成构造

$$\phi_{accept} = \bigvee_{j=1}^{n^k} x_{(n^k, j, q_{accept})}$$

… complexity $O(n^k)$.

同濟大學 TONGJI UNIVERSITY

Polytime Reduction Check 4

结论Given $w_1,\ldots,w_n$, the construction of

$$\phi_w = \phi_{cell} \wedge \phi_{start} \wedge \phi_{move} \wedge \phi_{accept}$$

requires $O(n^{2k})$ time and space: poly(n).

1 已知 SAT∈**NP**,
2 从 $w_1,\ldots,w_n$ 到 ϕ_w 的映射用 P-时间
把 一个*任意的* **NP** problem A 归约为 SAT:
即$A \leq_P$ SAT for all A∈**NP**.
由1,2 两条， SAT 是 **NP**-complete

同濟大學 TONGJI UNIVERSITY

3SAT is NP-Complete As Well

COROLLARY 7.42

***3SAT* is NP-complete.**

THEOREM 7.56

***SUBSET-SUM* is NP-complete.**

上述内容自学，结论可直接使用

同济大学 TONGJI UNIVERSITY

Proving NP-Completeness

小结　证明问题A是 NP-完全的技巧

1 . prove A is in **NP**.（不确定 猜—测（P-time））

2. 把一个NP-完全问题归约为A
(Give a reference for the **NP**-complete problem.)

现在为止，本书已经证明下列问题是NP-完全的
SAT, 3SAT ， CLIQUE，HAMPATH，SUBSET-SUM。

同济大学 TONGJI UNIVERSITY