计算复杂度讲座

晋宜蔡计算机科学系

University of Wisconsin
Madison, WI 53706
Email: jyc@cs.wisc.edu

目录

1 Genesis **7**

1.1 Structural versus Computational 7

1.2 Turing Machines and Undecidability 10

1.3 Time, Space and Non-determinism 13

1.4 Hierarchy Theorems 18

2 Basic Concepts **21**

2.1 P and NP 21

2.2 NP-Completeness 22

2.2.1 NP-completeness of VertexCover, IndependentSet, Clique . . 25

2.2.2 HamiltonianCircuit 27

2.3 Polynomial Hierarchy 29

2.4 Oracle Turing Machines 32

2.5 A Characterization of PH 33

2.6 Complete Problems for Σ_k^p 37

2.7 Alternating Turing Machines 37

3 Space Bounded Computation **39**

3.1 Configuration Graphs 39

3.2 Savitch's Theorem 40

3.3 Immerman-Szelepcsényi Theorem 42

3.4 Polynomial Space 45

3.4.1 QBF is PSPACE-Complete 46

3.4.2 APTIME = PSPACE 48

4 Non-uniform Complexity **51**

4.1 Polynomial circuits, P/poly, Sparse sets 51

4.2 Karp–Lipton Theorem 55

4.3 Mahaney's Theorem 57

5 Randomization **61**

5.1 Basic Probability 61

5.1.1 Markov's Inequality 63

5.1.2 Chebyshev Inequality 63

5.1.3 Chernoff Bound 64

5.1.4 Universal Hashing 67

5.2 Randomized Algorithms: MAXCUT 68

5.2.1 Deterministic MAXCUT Approximation Algorithm 69

5.2.2 Randomized MAXCUT Approximation Algorithm 69

5.2.3 Derandomizing MAXCUT Approximation Algorithm Using Universal Hash Functions 70

5.2.4 Goemans-Williamson Algorithm 70

5.3 Randomized Algorithm: Two Stage Hashing 73

5.4 Randomized Complexity Classes 75

5.4.1 Definitions 75

5.4.2 Amplification of BPP 76

5.5 Sipser-Lautemann Theorem: BPP $\subseteq$ PH 77

5.6 Isolation Lemma 81

5.7 BPP $\subseteq \Sigma_2^p$ - Another Proof Using the Isolation Lemma 82

5.8 Approximate Counting Using Isolation Lemma 84

5.9 Unique Satisfiability: Valiant–Vazirani Theorem 87

5.10 Efficient Amplification 89

5.10.1 Chor-Goldreich Generator 89

5.10.2 Hash Mixing Lemma and Nisan's Generator 90

5.10.3 Leftover Hash Lemma and Impagliazzo-Zuckerman Generator 93

5.10.4 Expander Mixing Lemma 97

5.10.5 Ajtai-Komlós-Szemerédi Generator 99

6 Hartmanis Conjectures **103**

6.1 Introduction 103

6.2 A Theorem of Cantor 104

6.3 Myhill's Theorem 106

6.4 The Berman-Hartmanis Conjecture 106

6.5 Joseph-Young Conjecture 108

7 Interactive Proof Systems **111**

7.1 Interactive Proofs – An Example 111

7.2 Arthur-Merlin Games 112

7.3 Merlin–Arthur Games 115

7.4 AM With Multiple Rounds 117

7.5 AM and Other Complexity Classes 118

7.6 LFKN Protocol for Permanent 121

8 IP=PSAPCE **125**

9 Derandomization **127**

9.1 Pseudorandom Generators 127

9.2 One-way Functions 130

9.3 Goldreich-Levin Hardcore Bit 133

9.4 Construction of Pseudorandom Generators 135

10 Computing With Circuits **137**

10.1 Binary Addition 137

10.2 NC and AC Classes 138

10.3 Multiplication 139

10.4 Inner Product, Matrix Powers and Triangular Linear Systems 140

10.5 Determinant and Linear Equations 141

10.5.1 Trace of a matrix 141

10.5.2 Symmetric polynomials 143

10.5.3 Csanky's Algorithm for Determinant 145

11 Circuit Lower Bounds **147**

11.1 Historical Notes 147

11.2 Razborov-Smolensky Theorem 147

11.2.1 Approximating Constant Depth Circuits by Low Degree Polynomials 147

11.2.2 Low Degree Polynomials Cannot Approximate Parity 150

11.3 Switching Lemma and Parity Lower Bounds 150

11.3.1 Decision Trees 150

11.3.2 Random Restrictions 152

11.3.3 Proving Circuit Lower Bounds for Parity: Overview 152

11.3.4 Switching Lemma: Proof 155

11.3.5 Switching Lemma: Improved Lower Bounds 161

11.3.6 Circuit Lower Bounds 162

11.3.7 Inapproximability Type Lower Bounds 164

12 Miscellaneous Results **167**

12.1 Relativized Separation of NP and P 167

第1章

起源

章节大纲:结构与计算数学. 历史视角. 复杂性理论简介. 希尔伯特的第十个问题. **Turing machine.** 不可解脱. 坎托尔的对角法. 停止问题不可断定。 时空界**Turing machine.** 等级定理. 复杂类**(L,NL,P,NP,PSPACE,E,EXP).**

1.1 结构与计算

自古以来,数学思想一直有两大分支:结构理论和计算方法。 例如,欧几里得的"元素"(Euclid's Elements)是几何学上直到那时为止已知的很多东西的合成,它也主要是结构性的,因为它强调(和确立)定理和推论证明;相比之下,狄奥芬图斯的著作,如在亚里士米提卡(Aristmetica)中,主要是算法性的,有些方法可能追溯到2000年前的巴比伦数学中. 当然,这些数学思想并非相互对立,而是相互补充。 即使在欧几里得的"元素"中,人们也会发现诸如"欧几里得算法"(The Euclidean Algorithm)这样的算法性宝石,它发现了两种正整数中最大的常见的二极体. 这是早期算法设计成功的一个光辉例子,其正确性和效率要求从纯粹的结构意义上证明. 在希腊传统之外,其他古代文明也以各种方式强调结构理论,即通过推论来奖励一般定理的设定和证明,或者通过计算来寻求解决问题的有效计算方法。 例如,中国古代数学家似乎主要关心计算.

向历史上绝大多数古典大师致敬,从阿基米德,牛顿,莱布尼兹,欧勒,拉格朗格,高斯,..... . . .. .,计算是整个数学的一个不可分割的方面. 计算被授予大数学大厦的次要地位,

比如波尔巴基 但实际上,大结构发现的许多动机最初是计算性的. 例如,计算法的发明是为了便于计算天体的轨道以及测量表面积和体积;加洛瓦理论和有限群论是用来调查基数方程的溶解性的发现;Prime Number定理在经过许多计算实验后首先被高斯猜想. 结构数学取得的许多进步也极大地影响了计算数学的进步,这也是事实。

虽然可以说算法的学科和数学本身一样古老,但认真的数学研究算法本身,而不是使用算法,是一个比较新的发展.

也许我们可以追溯到这个开始 设定理论, 最结构性的课题。 坎托尔在研究傅里叶系列(肯定是起源地最计算性的学科之一)时,产生了一套我们现在称之为(naive)定理的理念. 坎托尔的思想在许多方面是革命性的. 在其基本框架内,它高度非建设性。 例如,坎托尔给出了一个概念上简洁而简单的证明,证明存在超自然数,从而发明了著名的对角化方法. 在许多方面,这一证据是显著的:

首先,这比赫米特和林德曼分别就e和Q的超越性所取得的巨大成就简单得多。 或许人们仍然可以证明,“真正的”超数理论更符合赫米特、林德曼和利乌维尔的思路,而不是对角化的魔法所证明的存在。 但即使是今天最专注的“硬性分析”实践者也不会否定坎托尔方法的优雅和效率。 另一方面,今天许多有趣的计算问题,如纬度的基数减少,同时的狄奥菲恩近似,以及对对流体的体积估计,形成了非常活跃的研究领域,可以直接追溯到诸如狄里希莱特,柳维尔,赫米特和明科斯基等作品.

第二,正如Kronecker迅速指出的那样,Cantor的方法本质上是非建设性,在他看来,它与 " 哲学 " 相接。 特别是,它不符合Kronecker一直倡导的严格有限制性和建设性的做法。 最后,克罗内克从未接受坎托尔的想法. 保守主义者在哲学上不信任它,这具有讽刺意味,因为保守主义者特别关注数学基础的健全性,在未来几年中,这要证明与计算不可计量性密切相关,其中坎托尔的对角化方法是一种先导论。

第三,对角化方法是在图灵关于“停止问题”的不可裁定性证据中找到它的巨大应用。 它后来成为了递归理论中的基本数学工具之一,在以时间和空间等级定理的证明而建立的复杂性理论中.

由于其根本重要性,我们将审查坎托尔的对角化证明。

代数是多诺米子的根,具有整体系数. 非代数数称为超数. 如果一个集可以和整数一起放入一对一的对应中,则可以计算. 显然,所有代数数的集合都是可以数的,因为我们可以数出所有整体的多诺米数,而N级的每个多诺米数最多都有n个根.

Exercise: Show that the following is a *pairing function* which gives a one-to-one correspondence between non-negative integers $\mathbf{Z}_+ = \{0, 1, 2, \ldots\}$ with the cartesian product $\mathbf{Z}_+ \times \mathbf{Z}_+$.

$$\langle i, j \rangle = \binom{i + j - 1}{2} + j.$$

锻炼:显示理性数组,代数数组都是可计数的.

定理1.1 实数组不可计数;特别是有非数实数.

一个奇特的历史注释:为了不冒犯克罗内克,克罗内克同时有权势,有些轻薄,可能阻碍这部作品的出版,坎托尔不得不严格地用非数论数字的存在来表述他的主要结果,而丝毫不提他正在出现的无限论.

Proof. Consider all binary infinite sequences $B = \{\beta\}$, where

$$\beta = b_1 b_2 \ldots b_n \ldots,$$

and $b_i \in \{0, 1\}$. We know that the real numbers in $[0, 1]$ can be put in 1-1 correspondence with this set B, via binary expansion.[1]

索赔:B无法计算。

Suppose otherwise. Let $\beta_1, \beta_2, \ldots, \beta_n, \ldots$, be an enumeration of all B, where each $\beta_i = b_{i,1} b_{i,2} \ldots b_{i,n} \ldots$. Write these sequences by the rows, and we obtain an infinite table as follows.

1 在[0, 1]中存在一些实际数字的技术问题,有两个不同的无限二进制扩展. 但很容易看出,这些真实数字正是那些有限度终止二进制扩张的数字,因此是理性的数字,并且显然是可以计算的. 因此,与B的1-1份信函的索赔仍然有效。

	1	2	3	4	$\cdots$	i	$\cdots$
β_1	$\boxed{b_{1,1}}$	$b_{1,2}$	$b_{1,3}$	$b_{1,4}$	$\cdots$	$\cdots$	$\cdots$
β_2	$b_{2,1}$	$\boxed{b_{2,2}}$	$b_{2,3}$	$b_{2,4}$	$\cdots$	$\cdots$	$\cdots$
β_3	$b_{3,1}$	$b_{3,2}$	$\boxed{b_{3,3}}$	$b_{3,4}$	$\cdots$	$\cdots$	$\cdots$
β_4	$b_{4,1}$	$b_{4,2}$	$b_{4,3}$	$\boxed{b_{4,4}}$	$\cdots$	$\cdots$	$\ddots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$		
β_i	$b_{i,1}$	$b_{i,2}$	$b_{i,3}$	$b_{i,4}$	$\cdots$	$\boxed{b_{i,i}}$	$\cdots$
$\vdots$	$\vdots$						

Now we define a particular $\beta = b_1 b_2 \ldots b_n \ldots \in B$, which by its definition can not be on this list: We go down the diagonal of this infinite table; for the n th entry, if it is 0 we let our new $b_n = 1$, and if it is 1 we let $b_n = 0$. Formally, $\forall n \geq 1$,

$$b_n = \begin{cases} 0 & \text{if } b_{n,n} = 1 \\ 1 & \text{otherwise.} \end{cases}$$

Thus we "constructed" our $\beta \in B$ to be "disagreeable": It disagrees with the n th item on the list in the n th place. Hence this β cannot be among those listed. (If it were the n th sequence, what should its n th entry be?) ♣

It is self evident why Cantor's method is called the diagonalization method.

Kronecker's objections not withstanding, Cantor's set theory opened up a mathematical "paradise", from which, Hilbert was said to have remarked that mankind will never be driven out again. Nevertheless, it did bring troubles with it. In particular, at the turn of the 20th century, a number of set theoretic paradoxes were found that pertain to the foundation of mathematics.

坎托尔的方法为什么被称为对角法,这自明了.

Kronecker的反对是不容置疑的,坎托尔的既定理论打开了一个数学的"paradise"(paradise),Hilbert据说从中说,人类再也不会被赶出去。 然而,它确实带来了麻烦。 特别是在20世纪之交,发现了一些与数学基础有关的成套理论悖论.

However, if $A \notin A$, then again by definition, $\mathcal{A} \in \mathcal{A}$.

以下是著名的罗素悖论.

1.2 Turing Machines and Undecidability

At the turn of the last century, paradoxes such as this one stirred up a lot of uneasiness with the foundation of mathematics, which was to be formulated increasingly based on Cantor's

1.2 Turing machine和不可改变性

在上世纪之交,这样的悖论在数学基础上激起了许多不安,数学基础的形成越来越基于Cantor的

幼稚的理论 这产生了重新审视数学基础的强烈愿望,并试图在不矛盾的逻辑基础上将所有数学正式化. (在现代的定理理论中,这种悖论通过更加谨慎地将什么作为定理可以被接受来避免).

希尔伯特是形式主义学校的主要倡导者. 在1900年他向国际数学家大会发表的著名演讲中,他列举了23个他认为最有可能在新世纪激发全世界数学家想象力的问题. 希尔伯特的一些问题都涉及基础问题,如"连续假说". 列表中的第10个问题询问如下: 寻找一个系统的程序,以确定几个变量中包含整体系数的多元性是否有整体解决方案. 更广义地说,希尔伯特发起了"决定问题"的研究,目的是寻找一个算法来决定每个问题的例子. 接下来的40年的研究表明,可计算性的研究与数学的基础密切相关. 其他几个希尔伯特问题也对数学基础和计算理论的未来发展产生了深远影响,如康蒂努姆假说(#1)和一致性问题(#2),但都是决定问题,a.k.a. 这是图灵的开创性工作的主要动力,它奠定了计算理论的基础。

事实证明,系统程序计算的概念与逻辑系统中正式证据的概念密切相关。 图灵,G'odel, Church, Post等人的作品,确定希尔伯特设想的原始方案不可能完整. 虽然希尔伯特在第10个问题中只要求算法,但这种算法不存在的可能性可能从未被他想到. 最终情况就是如此,因为1970年,马蒂亚塞维奇终于证实了这一点。 但是,是希尔伯特提出这个问题的, 并且把注意力集中在什么构成算法,什么是计算的概念上。 在回答希尔伯特时,计算论诞生了.

为了解决Hilbert的Entscheidungs problem,图灵定义了他的通用计算机模型——Turing machine. Turing machine(TM)具有有限的状态控制以及无限的输入/输出带,并带有读写头. 一个决定性的Turing machine在每一步中都以它目前状态决定的独特方式移动,它现在正在扫描的符号. 动作包括可能改变状态和扫描符号,以及左右移动头部. 一个非决定性的Turing machine(NTM)可能有几个法律动作,这些动作仍然由其当前状态和扫描符号决定. (NTMS对计算理论不重要,但对后来的复杂理论很重要.)如果从初始配置开始的计算以接受状态结束,则输入x会被确定性的TM接受. 就NTM接受而言,其定义是存在一些以接受状态结束的法律动作序列。 同样为了复杂理论的目的,我们也定义了多段Turing machine. 对于有亚线性空间的空间捆绑的Turing machine,我们允许一个单独的只读输入磁带,以及一个工作磁带(或几个磁带2),空间捆绑只计算在工作磁带上. 我们假设

2 对于空间复杂性来说,一个工作磁带就足够了。

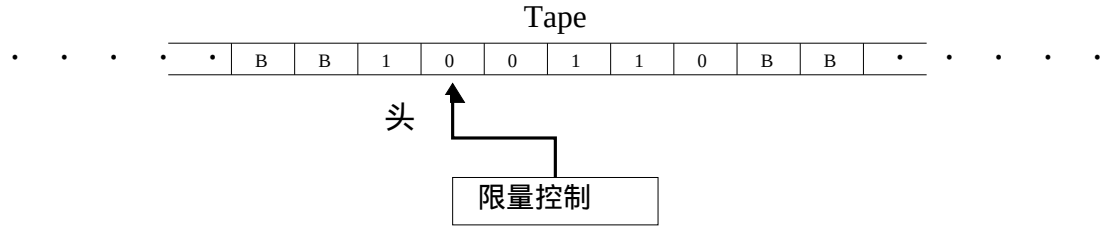


图1.1:Turing machine

读者已熟悉这些概念;详细定义见[3、4、2]。

计算模型并非唯一的计算模型。 在1930年代,制定了若干不同的模型;然而,在适当的时候,每个一般模型都显示相当于TMs。 " 教会-图灵论 " 抓住了TMs的普遍性,指出任何可以计算的东西都可以用Turing machine计算。

Cantor的对角化方法在技术上是艾伦·图灵的开口萨尔沃. 适应Cantor的方法,Turing 显示,有些问题无法用任何Turing机器来计算,因此,用Church-Turing论文来计算,无论用什么算法都无法计算。 此类问题被称为不可解答的问题. 特别是,所谓的Turing机器的停止问题就是这样一个问题。 此外,还发展了ducidon理论,可以表明一系列问题无法被断定。

停止问题无法断定的证据如下。

List all the Turing Machines $M_1, M_2, \ldots$ row by row, and index the columns by the inputs of all finite strings, which we will identify with integers $j = 1, 2, \ldots$. Mark the (i, j) entry of this table by $M_i(j)$, the outcome of machine M_i on input j. We will only consider the outcome of either "halt" or "not halt". We are not claiming this outcome can be computationally determined in general, and in fact the purpose of this proof is precisely to show that it is impossible to determine this outcome by TMs, and thus by the Church-Turing thesis, undecidable by any algorithm.

因此,我们得到了一个无限的桌子,这在坎托尔的证明中非常相似。

Now suppose there is a decision procedure in the form of a Turing Machine M that, for any $\langle i, j\rangle$, can decide whether $M_i(j)$ halts or not in a finite number of steps. Let's say $M(\langle i, j\rangle) = 1$ if $M_i(j)$ halts and $M(\langle i, j\rangle) = 0$ otherwise. Then one can design another Turing Machine M' as follows: On any i, $M'(i)$ simulates $M(\langle i, i\rangle)$. If $M(\langle i, i\rangle) = 1$, then $M'(i)$ enters a loop and never halts. If $M(\langle i, i\rangle) = 0$, then $M'(i)$ halts.

既然Turing machine器都列出来了 就有K,所以我们的M
是Mk语. 但Mk(k)会怎么样?

	1	2	3	$\cdots$	i	$\cdots$
M_1	$\boxed{M_1(1)}$	$M_1(2)$	$M_1(3)$	$\cdots$	$\cdots$	$\cdots$
M_2	$M_2(1)$	$\boxed{M_2(2)}$	$M_2(3)$	$\cdots$	$\cdots$	$\cdots$
M_3	$M_3(1)$	$M_3(2)$	$\boxed{M_3(3)}$	$\cdots$	$\cdots$	$\cdots$
$\vdots$	$\vdots$	$\vdots$		$\ddots$		
M_i	$M_i(1)$	$M_i(2)$	$M_i(3)$	$\cdots$	$\boxed{M_i(i)}$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\ddots$

If $M_k(k)$ eventually halts, then by the assumption of M, $M(\langle k,k\rangle) = 1$, and thus $M'(k)$, which is $M_k(k)$, never halts. If $M_k(k)$ does not halt, then again by assumption, $M(\langle k,k\rangle) = 0$, and $M'(k)$ halts. Either way we have a contradiction. ♣

接下来,我们将定义时间和空间的计算。 这是计算复杂性理论的领域. 我们将看到对角化方法再次出现,以确定更多的时间或更多的空间可以计算得更多。

1.3 时间、空间和非决定因素

以下概念是基本概念,详见任何参考书[3,4,2,1].

A DTM is formally $(Q, \Sigma, q_0, \delta, F)$, where Q is a finite set of states. Σ is a finite set of alphabet symbols, excluding a special symbol $\sqcup$. $q_0 \in Q$ is the starting state. $F \subseteq Q$ is a subset of accepting states. δ is a partial transition function that maps $(Q - F) \times (\Sigma \cup \{\sqcup\})$ to $Q \times \Sigma \times \{L, R\}$. The interpretation is as follows: when TM M is in state $q \in Q - F$, and reading symbol $A \in \Sigma \cup \{\sqcup\}$, if δ is defined, say $\delta(q, A) = (q', A', \Delta)$, where $\Delta = L$ or R, then in one step the partial transition function δ provides the next state $q' \in Q$, and overwrite the symbol by $A' \in \Sigma$, and moves left ($\Delta = L$) or right ($\Delta = R$). Thus, a TM is defined by a finite set of quintuples of the form (q, A, q', A', Δ), which can be written as an integer in a standard encoding scheme, e.g., using repeated applications of our pairing function $\langle \cdot, \cdot \rangle$. We can further assume every integer encodes a TM, with the simple convention that an "invalid" encloding corresponds to a TM that has no legal moves. Furthermore, we will almost exclusively use the binary alphabet set $\{0, 1\}$.

我们将使用类似定义的多磁带TM。 确定TM M的时间复杂性是

$$\text{time}_M(n) = \max\{\#\text{ of steps in } M(x), \text{ for } |x| = n\}.$$

对于任何函数f(n),

$$\text{DTIME}[f] = \{L \mid \text{ for some } M \text{ }, L(M) = L, \text{ and } \text{time}_M(n) \leq f(n), \text{ for all large } n.\}.$$

In other words, DTIME[f] consists of problems that are computable by some TM with running time at most $f(n)$ asymptotically. By some silly tricks such as enlarging the alphabet set, and the set of finite states, one can show that any constant factor does not matter. Thus DTIME[f] = DTIME[$O(f)$]. For technical reasons we will only consider "nice" functions $f(n)$, called fully time-constructible functions. This means that there is some TM M, which for every n, and any input x of size n, $M(x)$ runs in exactly $f(n)$ steps and then halts.[3] Almost any reasonable functions $\geq n$, such as n, n^k, $n^i(\log n)^j$, $2^{(\log)^k}$, 2^{n^k}, and $2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}$. They are also closed under addition, multiplication, exponentiation, etc. We will always assume $f(n) \geq n$, the time needed to read the input.

选择TM的型号是因为其相对坚固. 例如,人们可以显示,在时间f(n)中运行的任何k-tape TM都可以在时间O(f(n)log(f(n))中由2-tape TM模拟. 为了我们的目的,我们只需要时间O(f(n)2)中更微不足道的模拟,甚至需要1塔的TM。 这种模拟可以很容易地看出来如下: 将单磁带演化成2k轨,并使用一个超过22k符号的大字母集,例如. 将所有 k 磁带的内容保留在单盘上,并附上每个头部位置的标记. 然后,K-tape TM的计算步骤的一步被1-tape TM模拟为2扫. 注意,每次扫描已使用的磁带区域时,最多需要O(f(n)级。 对于时间的复杂性,默认模型是多磁带TMS,如以下定义.

Let poly denote the class of polynomials, or simply $n^i + i$, $i = 1, 2, \ldots$. Then the union

$$\text{P} = \bigcup_{f \in \text{poly}}$$

是决定性多语种时间的类别。 显然,这一定义在限制在1-TMs时是不一致的。 也可以同样定义指数时间类

$$\text{E} = \bigcup_{k>0}$$

$$\text{EXP} = \bigcup_{k>0}$$

人们可以同样地定义空间的复杂性。 在一个方面,它更简单,因为我们可以使用k轨道来模仿k磁带,而且模拟中没有额外的空间高空. 所以,我们将只有一个工作磁带。 然而,在另一方面,有一个轻微的Complicaiton。 我们希望研究包含重要问题的亚线性空间复杂性时,就会出现这种情况。 为了说明子线性空间,我们除了读写工作磁带外,还使用单独的只读输入磁带.

3 复杂性理论的一个事实是,存在的职能并非完全有时间结构,但我们不会对此感到关切。

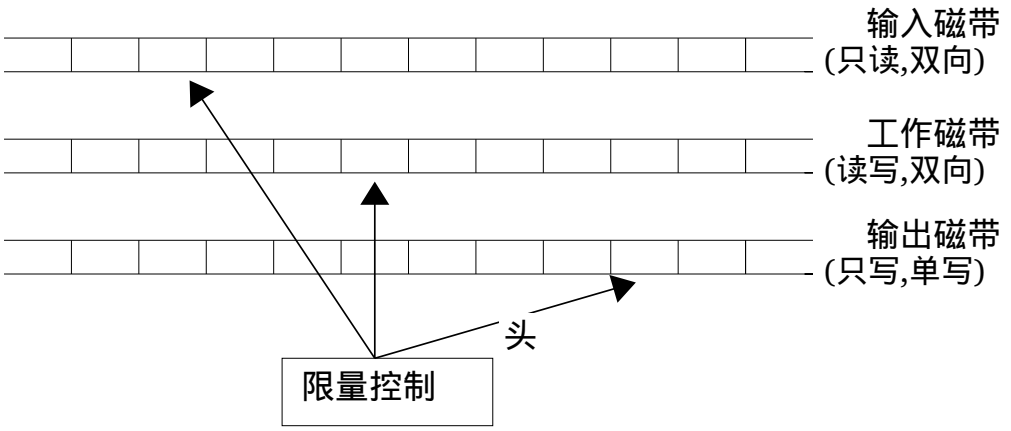


图1.2:多盘离线涡轮机

在只读的输入磁带上,长度n的输入被写入,但这些n的单元格并不算向空间复杂度. 我们只计算在读写工作磁带中使用的磁带单元格数量. 因此,对于空间复杂性来说,标准模型是被称为离线TM,它有一个只读的输入带,和一个读写工作带. 然后,人们可以明显地定义,

$$\text{space}_M(n) = \max\{\#\text{ of cells on work tape used in } M(x), \text{ for } |x| = n\}.$$

我们再次局限于“优秀”功能,即完全空间构建的功能。

演练:充分定义空间构造功能.

对于任何此类函数f(n),定义

$$\text{DSPACE}[f] = \{L \mid \text{ for some } M \text{ , } L(M) = L, \text{and space}_M(n) \leq f(n), \text{ for all large } n.\}.$$

我们定义

$$\text{PSPACE} = \bigcup_{f \in \text{poly}} \text{发证中心[f]},$$

the class of polynomial space. (There is a reason why we omit the word “deterministic”, as we shall explain later.) We also have deterministic logarithmic space L = DSPACE[log]. Note again that constant factors do not matter, thus DSPACE[log] = $\bigcup$c>0 DPACE[c日志].

所谓非定性的概念在复杂性理论中具有核心重要性. 这一概念与猜想和核查过程有关。 我们将考虑两个例子。 布尔满足性和图形无障碍。

假设我们在命题逻辑中被赋予布尔公式 {,即它是一个由逻辑和(∧),OR(∨)和NOT(¬)以及布尔变量x1,x2 组成的完善的公式. . . .. ,xn. (中文(简体)). 问题在于变量x1,x2, 是否有真实任务. . . .. , xn 这样, \\\\\\\\ \\\\\\\\

No polynomial time algorithm is know for SAT. But whenever a formula $\varphi(x_1, x_2, \ldots, x_n)$ is satisfiable, i.e., $\varphi \in$ SAT, there is an assignment, if one is given that assignment, one can easily verify that φ is satisfiable, in particular it evaluates to True under that assignment. We consider this process as consisting of two stages, first *guess* a witness, in this case a satisfying assignment, second *verification*, to check indeed it is a satisfying assignment. Note that φ is satisfiable if and only if such a guess exists which will pass the verification stage. Of course, one can consider this guess-and-verify to go hand in hand. In this Satisfiability Problem for example, we can imagine one guesses one bit at a time for each variable, and evaluates the formula as each variable is assigned. We will formally define non-deterministic Turing Machines (NTM) as TMs where for each state and tape symbol being read (q, A), the transition function δ provides a finite set of allowable next-step transitions consisting of a state q', a tape symbol A' and a left or a right movement, $\Delta = L$ or R, of the tape head. A NTM is defined by this set of quintuples of the form (q, A, q', A', Δ). For DTM, every (q, A) has at most one quintuple of the form (q, A, q', A', Δ) signifying at most one legal move; with NTM, each (q, A) can have a finite set of quintuples the first two entries being q and A (this finite set is part of the definition of δ).

这两种非决定性的计算观点,即每步2级猜测和验证或多个有效动作,对于SAT来说显然是等同的. 对于SAT等问题,或一般来说,对于非定时类(至少是线性cn),第一种方法,猜想和验证,是更直观的. 然而,对于非决定性计算的正式处理,采用第二种方法比较容易。 在处理可能无法终止的计算时尤其如此。 4 第二个方法的另一个更适合的地方是,我们考虑如下例子一样(在开始时没有足够空间来猜测和记录所有非决定性的动作),以次线性空间进行限定的计算。

考虑下图的无障碍问题:鉴于一个定向图G,以及两个顶点和顶点,我们询问是否有从s到t的定向路径。 我们通过GAP表示有G从s到t的定向路径的所有实例(G, s, t)的集合. (这个问题,在仅限于未定向图时,也是非常重要的. 通过将每个未定向的边缘替换成对向相反方向的一对定向的边缘,很明显,定向GAP比未定向的版本是一个更普遍的问题.)

For GAP, there is a simple polynomial time algorithm based on Depth-First-Search (DFS). However DFS uses at least linear space. By contrast, if the space bound is limited, we have the following non-deterministic algorithm: Start from $v_0 = s$, successively guess the next vertex v_i and verify that (v_{i-1}, v_i) is an edge. We can keep a counter to count up to n, and accept iff within $n - 1$ steps, some $v_k = t$. Note that we only need to keep at any time a pair of current vertices on the work tape; when we guess for v_{i+1} we need only to remember v_i (in order to verify that (v_i, v_{i+1}) is an edge), but we no longer need to keep

[4]We will not be concerned with non-terminating computations. For example, one can design a subroutine which acts as a time clock that runs exactly $n^i + i$ steps and shuts itself off, or a space bound which marks $f(n)$ (e.g. $f(n) = \log n$ or $n^i + i$) many tape squares.

vi − 1和所有之前猜测的顶点. 此外,我们只需要保留一个计数器就可以计数到n. 对于带有n顶点的图表,对顶点进行命名只需要对数n空间。 因此,GAP处于非决定性的对数空间.

我们只关心时间/空间的NTM,因此我们可以假设每个计算路径在指定的时间/空间约束内终止. 然后,对于NTM N和一个全时可构造的f,我们定义

$\text{time}_N(x) = \max\{n+1, \text{n p 沿 p 的步数,其中 N(x) 接受沿计算路径 p}\}$, 5

$$\text{time}_N(n) = \max\{\text{time}_N(x), |x| = n\},$$

$$\text{NTIME}[f] = \{L \mid \text{ for some } N \text{ }, L(N) = L, \text{ and } \text{time}_N(n) \leq f(n), \text{ for all large } n.\}.$$

Of course $L(N) = L$ means that $x \in L$ iff there is some computation path p, along which N accepts x. Similarly we can define non-deterministic space classes.

$\text{space}_N(x) = \max\{1, \text{n p 用于 N(x) 的工作磁带单元格, N(x) 接受 沿着 p*}, 6$

$$\text{space}_N(n) = \max\{\text{time}_N(x), |x| = n\},$$

$$\text{NSPACE}[f] = \{L \mid \text{ for some } N \text{ }, L(N) = L, \text{ and } \text{space}_N(n) \leq f(n), \text{ for all large } n.\}.$$

$$\text{NP} = \bigcup_{f \in \text{poly}} \text{NTIME[f],,,,,,,}$$

$$\text{NPSPACE} = \bigcup_{f \in \text{poly}}$$

和项目

$$\text{NL} = \bigcup_{c>0} \text{NSPACE[c日志].}$$

简而言之,正式来说,非决定性的TM(NTM)只是一种TM,在任何配置上都可能有不止一种法律动作. 这些下一步动作由当前状态及其正在读取的磁带符号指定. NTM接受一个输入xif存在一个以接受结束的法律计算路径.

It is obvious that L ⊆ NL ⊆ P ⊆ NP ⊆ PSPACE ⊆ EXP.

We will show later that NSPACE[f] ⊆ DSPACE[f^2], thus NPSPACE = PSPACE and NPSPACE will not be used.

P和NP的关系是计算机科学中最突出的开放问题.

5 当N不接受x时,时间复杂性为n + 1. 6 当N不接受x时,空间复杂性为1.

1.4 等级定理

复杂分类的定义试图根据每个问题的内在计算复杂性对可裁定的计算问题进行分类. 第一个问题是:这种分层是否确实存在,即这是否只是定义性的幻象?

为了真正确定计算复杂性理论存在的有效性,必须证明问题确实有不同的内在计算复杂性. 这不应仅仅取决于这样一个事实:虽然我们对一些问题有快速算法,但目前对另一些问题没有快速算法。 证据必须确定某一时间或空间受约束f(n),即存在某些可判定的计算问题,而这些问题在该受约束f(n)内没有任何算法。

结果确立了这些存在定理被称为等级定理. 人们可以显示,对于任何两次复杂函数T1(n)和T2(n),如果T2与T1相比增长足够快,那么确实存在问题,这些问题可以在时间T2中解决,但不能在时间T1中解决. (在这里,我们还需要技术条件,即T2是时间可构造的。)事实上,如果是limn →∞。

T1(n) 对数 T1(n)

We will prove a weaker theorem

1.2 如果 T2(n) 是时间可构造的和 limn 的,则考虑到任何 T1(n), T2(n)

$[T_1(n)].$

证据还特别规定:

1.3 鉴于任何完全递归时间约束T(n),DTIME[T(n)]中有一个不重复语言L。

也有非定时等级定理.

等级定理与存在不可解的问题的作用相同. 不仅如此,证明还适应了对角化方法. 在减数栏上,正如坎托尔的浮雕证明存在超常数字一样,这些等级定理并没有给我们带来某些众所周知的问题难以解决的具体细节。 当今复杂理论的研究更多地以寻求这种“真正的”问题为主。

我们只提供一份证明草图:

证得. 一般思想是对角化. 首先,我们注意到,所有TMS都可以被有效地列为M1、M2。 . . .. 对于一个可构造的T(n),人们也可以设计一个"钟",一个子例程,它与任何TM同时运行,并在T(n)步后关闭. 我们可以有效地列举所有“有时钟”的TM,所有TM在T(n)时间内运行,DTIME[T(n)]中的每一种语言都为本列表中的一台机器所接受.

We will enumerate all the Turing Machines $M_1, M_2, \ldots$, and consider each M_i on longer and longer inputs. We simulate each of them on these inputs, for up to $T_2(n)$ steps for inputs of length n. To be more precise, we will allocate a disjoint segment $S_i = \{\langle i, y\rangle \mid i \geq 1, y \in \Sigma^*\}$, and simulate M_i on every $x \in S_i$. If the simulation on $M_i(x)$ terminates in less than $T_2(|x|)$, then *we will do the opposite,* i.e., we accept x if $M_i(x)$ rejects, and we reject x if $M_i(x)$ accepts. If the simulation of $M_i(x)$ does not terminate within $T_2(|x|)$ steps, then we can decide arbitrarily, accept or reject.

To account for the possibility that the simulation of $M_i(x)$ does not terminate within $T_2(|x|)$ steps, we allocated an infinite subset of inputs for every M_i, Since one can simulate (easily) a multitape TM M_i with running time $T_1(n)$ by a one tape TM in time $T_1(n)^2$, for sufficiently large $x \in S_i$, the simulation of $M_i(x)$ will terminate within $T_2(|x|)$ steps. Now suppose $L \in \text{DTIME}[T_1(n)]$. Then for some TM M_i accepting L with runing time $T_1(n)$, the simulation will terminate with a different outcome, for sufficiently large n. Thus the language L' defined by this simulation does not belong to $\text{DTIME}[T_1(n)]$. However, by the construction it is in $\text{DTIME}[T_2(n)]$. ♣

请注意,模拟是由单TM进行,它必须有固定数量的磁带. 据悉,任何具有运行时间T(n)的k-tape TM,都可以在O(T(n)log(T(n)))时间中由2-tape TM模拟. 这是上述证据中唯一需要修改的,以便获得前面引用的更严格的等级定理.

这种证明的方法有时被称为延迟对角化. 这是因为,杀死TM Mi的"对角行动"可能在稍后阶段发生(就我们而言,对Si的投入来说,可能比i长得多)。 套集Si有无限的字符串,因此任意的长字符串. 因此,"对角行动"最终将会发生,每一次TM都有运行时间T1(n)。

There is also a similar version of the Hierarchy Theorem for space complexity. In fact since for space complexity we can simulate any TM in just one tape without any loss in asymptotic space complexity, the theorem reads even tighter: ($S_2(n) \geq \log n$ is needed to carry out some basic counting operations in simulations.)

THEOREM 1.4 *Given any $S_1(n)$ and $S_2(n) \geq \log n$, if $S_2(n)$ is fully space constructible and $\lim_{n\to\infty} \frac{S_1(n)}{S_2(n)} = 0$ (i.e, $S_1 = o(S_2)$), then there is a language $L \in \text{DSPACE}[S_2(n)]$ $-\text{DSPACE}[S_1(n)]$.*

方法不变。

锻炼:验证太空等级定理.

第2章

基本概念

章节大纲:在本章中,我们讨论一些基本概念. 我们定义了削减,NP的完成度,交替,相对化,导致波利诺米亚时间等级制.

2.1 P和NP

在第一章中,我们引入了决定性和非决定性的Turing machine及其复杂限定版本. 基于这些概念,我们界定了决定性和非决定性的时间复杂性类别。 我们首先回顾这些课程,继续讨论。

2.1 对于任何时间性函数f(n),

$$\begin{aligned}\mathrm{DTIME}[f] &= \{L|\ \textit{for some deterministic TM } M,\ L = L(M)\ \textit{and } M\ \textit{runs in time } f(n)\}\\ \mathrm{NTIME}[f] &= \{L|\ \textit{for some non-deterministic TM } M,\ L = L(M)\ \textit{and } M\ \textit{runs in time } f(n)\}\end{aligned}$$

由于每个决定性TM也可以被看作是一个非决定性TM,我们有以下提议.

PROPOSITION 2.2 *For any function $f(n)$, $\mathrm{DTIME}[f] \subseteq \mathrm{NTIME}[f]$.*

我们通过考虑在多边时间运行的决定性和非决定性的TMs来获得两个核心复杂性类别:P和NP:

定 义 2.3

$$\begin{aligned}\mathrm{P} &= \bigcup_{f\in\mathrm{Poly}}\\ \mathrm{NP} &= \bigcup_{f\in\mathrm{Poly}}\end{aligned}$$

Poly是所有多边名词的组合。

It follows from these definitions that P $\subseteq$ NP, and perhaps the greatest open problem in computer science asks whether this containment is in fact proper, namely NP $\neq$ P.

2.2 NP-完成性

复杂性理论中最有用的概念之一是完整性的概念. 一般来说,可以定义任何复杂类的完整语言. 本节专门讨论班级NP的完整语言. 我们首先确定另一个重要的概念,即减少。

Definition 2.4 (Karp reduction) *A language L_1 reduces to another language L_2 by a Karp reduction, denoted as $L_1 \leq$*m L2, 如果有函数 f : QQ, 这样

1. $x \in L_1 \iff f(x) \in L_2$, and

2. 国家 f 是可计算出的确定性多边计算时间。

卡普还原也被称为多诺米时间(polynomial time many-one reduction),这是多诺米时间版本的复发理论(英语:Recursion theoretic orretic)概念多1还原. 还有其他一些减少的概念,将在稍后讨论。 我们现在着手界定国家规划的完整概念。

Definition 2.5 *A language L is said to be* NP*-hard (under Karp reductions) if every language $L' \in$ NP Karp reduces to L. L is said to be* NP*-complete if $L \in$ NP and it is* NP*-hard.*

接下来我们展示的是 一种犬形NP语言 ANP。 定义如下。

$$A_{\text{NP}} = \{\langle M, w, 1^t\rangle |\ M \text{ is a non-deterministic TM that accepts } w \text{ within } t \text{ steps}\}$$

2.6 ANP提案为NP-完成.

证得. 很容易看出ANP在NP. 考虑到QQM,w,1t⟩as输入,我们模拟M在w(非决定性)上为t步骤,如果M接受w,则接受输入. 显然,这种算法在非决定性的多元时段运行.

We next show that A_{NP} is NP-hard. Let $L \in$ NP via a non-deterministic TM M that runs in time $O(n^c)$, where c is some constant. Our reduction algorithm, given input w, simply outputs $\langle M, w, 1^{n^c}\rangle$. It is clear that $w \in L$ iff $\langle M, w, 1^{n^c}\rangle \in A_{\text{NP}}$. Our reduction algorithm takes (deterministic) time $O(n^c)$, and hence runs in polynomial time. ♣

我们注意到,只要该类有“按时计时”的机器,这种证据就很容易适应许多其他复杂类。 例如,对于PSPACE类,我们可以定义一个限定空间的列表(由nj + j) TMs M⟨i,j⟩,并形成这样的“通用语言”,对于PSPACE来说,它将是完整的.

$$A_{\text{PSPACE}} = \{\langle M, w, 1^t\rangle |\ M = M_{\langle i,j\rangle} \text{ accepts } w \text{ within } t = |w|^j + j \text{ space}\}$$

因此,仅是标准复杂性类的完整语言的存在就不足为奇了。 然而,这些不是“自然”语言。 在表明存在一个NP完成问题之后,我们把注意力转向“自然”或“现实世界”完成NP的问题。 完成国民方案的重要性在于这些“自然”的国民方案完成问题。 如此著名的问题之一是SAT.

$$\text{SAT} = \{\varphi | \varphi \text{ is a satisfiable boolean formula }\}$$

著名的Cook–Levin定理称SAT是NP-完成的.

定理2.7(Cook–Levin)SAT为NP-完成.

证明的办法是,鉴于一个多诺时数NTM M和一个输入w,计算w上的M机可以编码成布尔公式,使M接受w,如果且只有在该公式是可调和的. 此外,如果机器在多诺时段运行,公式的长度也会在输入时长为多诺时段. 基本的想法是使用布尔变量和逻辑连接符 QQ,QQ,QQ组成一个命题逻辑公式,这个公式在CNF(一些词词的解开结合)中,它"说"计算M(x)接受一些非定型计算路径. 为了正式证明定理,我们请读者参考标准教科书[2,3,4]. 布尔逻辑可以表达这一点,这实在是不足为奇。 但是,实现SAT对于NP来说是一个如此普遍的问题,其影响是巨大的.

Following the work of Cook, Karp showed that many other “natural” problems are also NP-complete. It is with the realization of this abundance of “natural” NP-complete problems that the importance of NP-completeness is truly demonstrated. Apart from its theoretical importance, the notion of NP-completeness is also very useful from a practical perspective. If an NP-complete problem is shown to be solvable in polynomial time, then every problem in NP would also be solvable polynomial time, and hence, it would imply that NP = P. But, it is widely believed that NP $\neq$ P. One indication is perhaps that there are thousands of NP-complete problems and no poly-time algorithm has been found for any of them. (Although this is admittedly a weak argument in favor of the Conjecture.) So, showing a problem to be NP-complete is taken to be a strong indication that the problem is not solvable in polynomial time and hence computationally hard. In fact, following this scheme, over the past three decades, thousands of “practical” problems have been shown to be NP-complete. A book by Garey and Johnson [2], now a classic, includes a compendium of many of these NP-complete problems.

回到SAT,我们看到,由于布林公式, 很难决定它是否适合。 在这个公式中,我们允许输入是任意的布尔公式。 但事实上库克的证明表明它仍然是NP-硬的,即使我们限制输入公式是交联的正常形式(CNF)(即. 公式为 Ors 的 AND,例如 (x1 ×××2 ××3 ×××4 ××(x1 ×××6)). 通常将上述版本(CNF-SAT)简单地称为SAT.

一旦我们确定一个问题 QQ 完成 NP 完成, 以证明 NP 完成 另一个问题 QQ 在 NP 完成, 我们可以将已知的 NP 完成问题 QQ 减少到给定问题 QQ。 这仅仅是卡普削减的过渡性后果。 如果我们从一个已知为NP完成的问题开始,并且它具有尽可能严格的形式,这项任务就更容易了。 例如,比起一般的CNF版,减少SAT的量就更容易了。 在沿着这条线前进的过程中,我们接下来显示,一个更加限制性的SAT版本,即3-SAT,也是NP完成的.

3-SAT是表型中所有可调色布尔表达的一组

$$\varphi = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{3} \text{丽,j}$$

为了将SAT改为3-SAT,我们采用了一种称为“蝴蝶构造”的构造。 想法是,对于任何超过三个组成部分的条款,我们通过增加额外的变量,将条款分为3字节的条款. 考虑以下公式:

$$\varphi = (x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6)$$

我们只是将这个公式分为以下三个部分:

$$\begin{aligned} \varphi &\rightarrow (x_1 \vee x_2 \vee y_1) \wedge (\overline{y}_1 \vee x_3 \vee x_4 \vee x_5 \vee x_6) \\ &\rightarrow (x_1 \vee x_2 \vee y_1) \wedge (\overline{y}_1 \vee x_3 \vee y_2) \wedge (\overline{y}_2 \vee x_4 \vee x_5 \vee x_6) \\ &\rightarrow (x_1 \vee x_2 \vee y_1) \wedge (\overline{y}_1 \vee x_3 \vee y_2) \wedge (\overline{y}_2 \vee x_4 \vee y_3) \wedge (\overline{y}_3 \vee x_5 \vee x_6) \end{aligned}$$

拨打最后一个公式。 现在我们注意到以下几点: 如果我们在 QQ 中指定了所有xi 是虚假的, 那么为了满足 QQ , 我们可以在 QQ 中一个条款中从左到右进行解释, 我们可以看到所有 yj 必须是真实的 : x1, x2 是虚假的, 所以 y1 是真实的; 然后 y1, x3 是虚假的, 所以 y2 是真实的 ; 等. 但是,这仍然会使最后的条款不实。 (想象气泡从左向右推).

另一方面,如果我们指定了一些xi属实的话,那么我们可以通过适当指定每个yj来“支持”双方的xi,以便满足所有3-升条款的要求。 例如假设 x3 是真实的。 然后它看起来像是以下的,其中 QQ代表真实和 QQ 代表假的全字(包括非符号):

这就是称之为蝴蝶建造的动机.

y_1	$\overline{y}_1$	x_3	y_2	$\overline{y}_2$	y_3	$\overline{y}_3$
↑	↓	↑	↓	↑	↓	↑

考虑到在CNF中的任何SAT实例,我们对长度大于3的每个条款采用相同的转换,为每个条款增加新的变量。 如果一个条款的字数少于3,则有微小的转变,使3个字的合体相当于它. 那么很明显的是,原配方是可调和的,如果变换的配方是可调和的.

我们证明了以下的定理。

定理2.8 3-SAT为NP-完成.

One can also easily prove that the problem remains NP-complete if we restrict to SAT instances with exactly k literals per clause, for any $k \geq 3$. This is called k-SAT problem. It is interesting to note that 2-SAT is decidable in polynomial time as it can be formulated as a graph of logical implications and one can find if the implications are consistent or not in polynomial time.

锻炼:显示**2-SAT**在**P.**

随着3SAT的NP完成的手头,我们可以通过将3-SAT缩小为它们来表明其他许多问题是NP-完成.

2.2.1 克里克州**VertexCover**的**NP**完成情况

我们首先定义VertexCover, 或VC:

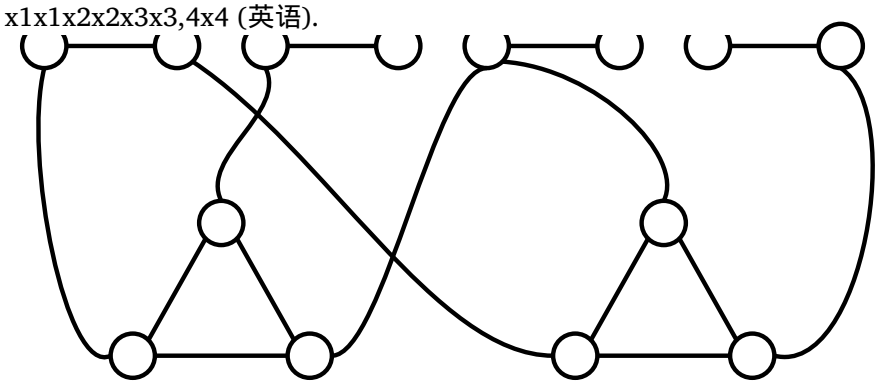
$$\mathrm{VC} = \{\langle G, k\rangle \mid \exists S \subseteq V(G), |S| \leq k, \text{ every edge of } G \text{ is incident to at least one vertex of } S\}$$

我们现在利用基于小装置的建造,将SAT减少到VertexCover。 这种构造的理念是在一个系统中创建对象(在这种情况下,我们将创建专门图表),这些对象对应另一个系统中的基本对象(clauses and variables),这样,第一个系统中的构造对象之间就存在某种关系,如果并且只有在第二个系统中所有对象之间存在另一种关系的话。 就《维也纳公约》的构建而言,我们希望能够涵盖某些顶点,如果它们所代表的变量的分派与满意的分派相符。

现在,我们给建造。 我们把3Sat减少到VC 我们将发现,对条款规模的限制是有利的。 假设有 n 变量 x1, x2, ... , xn 和 m 条款。 首先,我们构建顶点x1,x1,x2,x2. . . .. , xn, xn. (中文(简体)). 这些顶点将对应公式中存在的 n 变量可能存在的真理任务. 我们添加边缘(x1,x1),(x2,x2),(x2). . . .. , (xn, xn) ,它迫使我们必须为每个变量选择至少一个值的制约。 我们称这为内在的边缘

Then, we construct vertices $\hat{l}_{11}, \hat{l}_{12}, \hat{l}_{13}, \hat{l}_{21}, \hat{l}_{22}, \hat{l}_{23}, \ldots, \hat{l}_{m1}, \hat{l}_{m2}, \hat{l}_{m3}$, which correspond to the literals of each clause. These nodes are connected in a triangle for each clause: $(\forall i, 1 \leq i \leq m)\{(\hat{l}_{i1}, \hat{l}_{i2}), (\hat{l}_{i2}, \hat{l}_{i3}), (\hat{l}_{i3}, \hat{l}_{i1}))\}$. We call these the intra-clause edges.

最后,一个条款中的每个字节点QQlij与相应的顶点lij相连. 例如,如果lij是xk,我们有(^lij,xk);同样,如果lij是xk,它也会是(^lij,xk). 我们称它们为方阵间边缘.



Now we'll step back and reason about our construction. Suppose the formula represented is satisfiable. Then we can construct a vertex cover of size $n + 2m$. We pick the n vertices corresponding to a satisfying truth assignment σ. (If x_1 is set to "true" in σ, include the vertex labelled x_1 in the vertex cover, else include the one labelled $\overline{x_1}$.) Then, in each triangle corresponding to a clause, we cover two literals which include all literals which are set false. Note that if the formula is satisfied under σ, then any clause has at least one literal which is true, hence at most two literals which are false. With one vertex for each pair of x_i and $\overline{x_i}$, and 2 vertices in each triangle, we note that all of the intra-variable and intra-clause edges will be covered no matter what the assignment. We note also that all the inter-formula edges are covered as well, if σ is a satisfying assignment: An inter-formula edge connecting a variable node to a satisfied literal is covered by the variable node; an inter-formula edge connecting a variable node to an unsatisfied literal is covered by the literal node in the triangle for the clause. So we have a vertex cover of $n + 2m$ vertices.

反之,假设有一个封面带有n+2m的顶点. 我们显示,这种报道必须代表一个令人满意的任务。 为了覆盖所有可变和可变的内缘,我们需要覆盖至少n+2m的顶点,每个变量1个,每个条款2个。 因此,这些n + 2m 的顶点由每个对(xi,xi)的精确一个组成,每个三角形的精确两个. 但这意味着每个条款的3个方块间边缘中只有2个被三角形的字形节点覆盖,第三个方块间边缘必须被代表字形的节点覆盖. 因此,如果我们在顶点封面上精确地指定任何字面内容,则公式就满足了。

因此,如果存在大小为n+2m的封面,则该公式是可调和的。 我们证明了

2.9 VC定理为NP-完成.

我们还注意到,《独立报》是《国民党-完成报》。 首先,我们界定问题:

$$\text{IS} = \{\langle G, \ell\rangle \mid \exists S \in V(G), |S| \geq \ell \text{ such that } \forall u, v \in S : (u, v) \notin E(G)\}$$

Observe that, if C is a vertex cover of a graph G, then its complement $V - C$ is an independent set, and vice versa. Thus, a graph has a vertex cover of size $\leq k$ if and only if it has an independent set of size $\geq n - k$. We conclude that IndependentSet is NP-complete. They are essentially the same problem.

2.10 IS定理为NP-完成.

最后,我们注意到,Clique是NP-Complete。 Clique与IndependentSet本质上也是相同的问题:

$$\textsc{Clique} = \{\langle G, \ell\rangle \mid \exists S \in V(G), |S| \geq \ell \text{ such that } \forall u, v \in S : (u, v) \in E(G)\}$$

As such, by flipping the edges of G (construct a G' such that $V(G') = V(G)$ and $(u, v) \in E(G') \iff (u, v) \notin E(G)$), we see that by the VC reduction, Clique is also NP-Complete.

定理2.11 Clique为NP-完成.

2.2.2 汉密尔顿环形山

Another graph problem we can define is HamiltonianCircuit, or HC. We define a Hamiltonian circuit to be an ordering of the vertices of a graph G as $v_1, v_2, \ldots, v_n$, such that we can travel through all the vertices once and return to where we started: $(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n)$, $(v_n, v_1) \in E(G)$. Then,

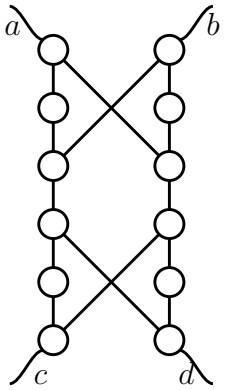
$$\text{HC} = \{\langle G\rangle \mid \exists \text{ a Hamiltonian circuit in } G\}$$

显而易见,HC是在NP,因为我们可以核实,对顶点的订购代表了多诺米时间的循环.

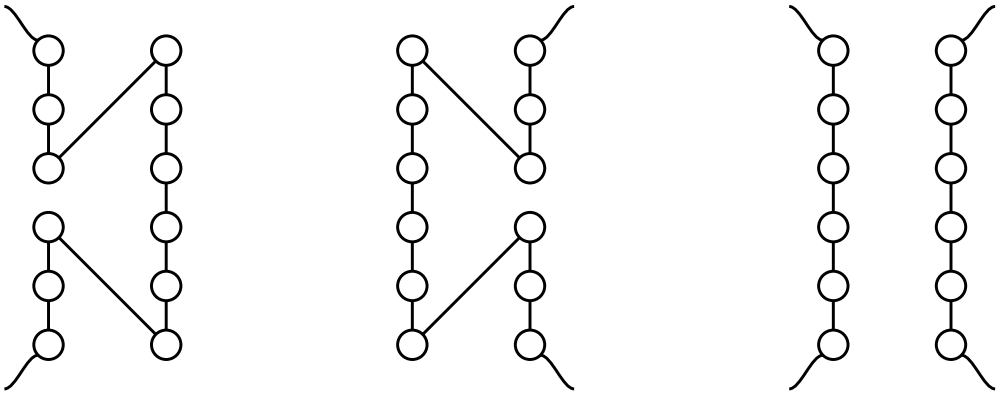
从《维也纳公约》中减少的数额

接下来是HC的NP-完成性的证明. 与《维也纳公约》一样,我们将利用基于装置的建筑。 然而,我们不会从3Sat中削减,而是会从VC中削减。

考虑下图,我们将称之为我们的工具:



只有三种方法可以绕过这个图(可以通过检查来验证): (1)a-b-d-c通过——通过a-b-d通过c,通过b-d;或者 (2)b-a-c-d通过——通过b-和通过d;或者 (3)通过两个通道:先通过a-c通过,然后通过b-d通过. 在所有情况下,都有可能重新排列出入境顺序。 以下是与这些路径相对应的三个图:



Given G, an instance for VC, we construct G', an instance for HC. For each edge $e = (u, v)$ in G, we use one such gadget in G' representing the edge e. The left half of this gadget represents u and the right half represents v. For each vertex u, we link together in some arbitrary order, top to bottom, all of the u halves of the gadgets that represent edges incident to v. (We'll refer to this as a *linked vertex chain*.) Finally, let k be the size of the VC in question, then we add vertices $a_1, \ldots, a_k$ and connect each of them to both the start and end of each linked vertex chain.

目的是要说明G中顶点是如何通过a1产生的每个相连顶点链的横贯来覆盖的. . . .. ,"Ak in G"("k"). 在上述图片中,如果u是VC中的K顶点之一,那么对于每个顶点在链接的顶点链上对每个顶点进行反转,我们将在(u,v)被u单独覆盖时,或者在(u,v)被u和v同时覆盖时,绕过所有顶点(如第一个选择的转盘),或者在(u,v)被覆盖时,绕过一半的顶点(如第三个选择的转盘). 如果不是在VC,而是v, 那么,这个装置的所有顶点 将像中间 选择转盘。

We now give a short argument for the correctness of this construction. First, we show how to traverse the graph if there is a vertex cover of size k. We note that if there is a cover of size less than k, it can be easily extended into a cover of size exactly k simply by adding extra vertices to the optimal cover, as they cannot hurt. So we assume $k \leq n$ and have a cover of size k. For the ith vertex (say u) in the cover, we start at a_i, trace out the linked vertex chain for u before returning to a_{i+1}, and if $i = k$ we return back to a_1, completing the tour. For each linked vertex chain, say of u, we traverse each gadget corresponding to an edge (u, v) as follows. If the edge is covered by u alone, then we traverse in an *a-b-d-c* pass (where the *a-c* side corresponds to u), and if the edge is covered by both u and v, then we traverse in an *a-c* pass. Clearly, from a VC of size k we get a Hamiltonian Circuit.

现在,我们声称,如果我们有一个周期图G', 在原始图G中有一个顶点覆盖大小k. 如果我们有一个循环,它必须包括所有AI的顶点和所有小装置的顶点。 但我们知道,每次访问AI顶点之间,我们只能访问一个与G中一个顶点u相对应的链接顶点链. 我们称这些k顶点构成了G的顶点覆盖. 对于这个相连的顶点链上的每一个设备,我们可以看到它是如何被访问的,它使顶点的盖子在G中相近。 由此可见,这些K顶点确实在G中构成了顶点覆盖。

定理2.12 HC为NP-完成.

2.3 多元等级制度

我们把NP定义为非定型多语制时间TM所接受的所有语言的类别. 我们可以给NP一个等效的特征.

定义2.13(确定)多诺时段存在布尔二进制上游 D( ) , L 语言在 Σp 1 中, 且多诺 q , 以便对所有输入 x,

$_1$ *if there exists a boolean binary predicate* $D(\cdot, \cdot)$

an input x, simply guesses a string y of length $q(|x|)$ and then accepts if $D(x, y) = 1$. On the other hand, suppose $L \in \Sigma_1^p$ via an NP machine M that runs in time $q(n)$. Then, for any input x,

$$x \in L \iff (\exists \text{ a computational path } p \text{ of length } q(|x|)\ )[\ M \text{ accepts } x \text{ along the path } p\ ]$$

Clearly, $L \in \Sigma_1^p$.

In the above formulation of NP, the predicate D is usually called a *verifier* and if $D(x, y) = 1$, then y is called a *certificate* or *proof* for the input x. For example, in the

case of SAT, we can construct a verifier as follows. Given a formula φ and a truth assignment σ, accept (φ, σ), if σ satisfies φ. Any satisfying truth assignment of φ is a certificate.

我们可以概括一下'p'的定义 1,定义一个较大的类如下.

putable in polynomial time, such that L is in Σ_2^p *if and only if* 2.如果在多诺时存在布尔的上游D,则L语在 Σp 2中,例如:

where $|y|$ and $|z|$ are polynomially bounded in $|x|$.

在那里,和\\和\\\\\\和\\\\\\和\\\\\和\\\\\\和\\\\\\和\\\\\和\\\\和\\和\\\\\和\\\\和\\\\和\\\\和\\\\和\\和\\\\和\\和\\和\\和\\\\和\\\\和\\\\和\\\\和\\\\和\\\\和\\和\\和

if there exists a $(k+1)$-ary boolean predicate D computable in polynomial time such that

定义2.16(p) k和Π p(k) L种语言在Σ p中

k 如果存在(k + 1)- 双布尔上游 D 在多诺时可计算,例如

where Q^p is either $\exists^p$ for k odd or $\forall^p$ for k even, and the superscript p denotes that all $|y_i|$ are polynomially bounded in $|x|$. In the above definition, $\exists$ and $\forall$ alternate. We say that a language L is in Π^p_k

It is clear that

$$\cdots \cap \Pi^{k+1}_p \cdot$$

k,如果其补充Lc为 Σp

DEMINATION 2.17 *For* HY (PH)) PH= $\bigcup_k \Sigma^p_k = \bigcup_k \Pi^p_k$

A picture of the polynomially time hierarchy is shown in Figure 2.1.

很明显,

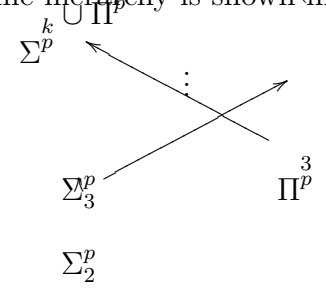


图2.1显示了多轨时间结构图。

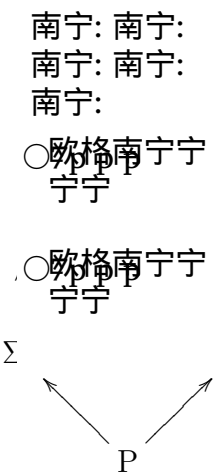


图2.1: 波利诺米时间等级

It is a big open problem whether the polynomially hierarchy is infinite, i.e., whether it has infinitely many distinct levels. It is quite conceivable that the hierarchy collapses to some k. But,

(单位:千美元) 但是,大多数人认为它是无限的。 人们可以证明一个简单的结果 与崩溃的等级。

THEOREM 2.19 *If* $\Sigma^p_k = \Pi^p_k$

定理2.19 如果

k(多诺时所分级崩溃到水平 k.)

For, say $\Sigma_p^k \subseteq \Pi_p^k$. $x \in L \iff \exists^p y \exists^p z E(\langle x, y\rangle, z)$ which can be rewritten as $\Pi_p^k \subseteq \Sigma_p^k$, as well as $x \in L$ implies $\exists^p y \forall^p z E(\langle x, y\rangle, z)$ $\Sigma_p^k \subseteq \Pi_p^k$

评论:我们注意到 (单位:种类)因为,说... ... k 表示,而且由 Σp 表示 K, 以及 Πp

pair x and z to form $\langle x, z\rangle$. Then calls E with $\langle x, z\rangle$ and z. This shows that L is a language in Σ_1, and hence, Σ_1^p $\Sigma_2^p = \Pi_2^p$ lexity theory. Next, we discuss oracles. Later we shall present an equivalent characterization of PH using oracles.

证得. 为了了解证据 我们假设

2. 国家 现在,一个字符串x,

2.4 Oracle Turing Machines

以另一个上游Q(⟨x,y⟩,z)重写,Q先是"不拉动"的⟨x,y⟩变为x和y,并称P. 因此,我们

query tape. The TM also has three special states called $q_?$ (the query state), q_{yes} and q_{no} (the answer states). Let A be a predetermined language. The computation of an oracle TM with oracle A is defined as follows. During its computation, the TM may write a string z on the query tape and enter the query state. In the next step, the machine will enter the state q_{yes} if $z \in A$ and will enter the state q_{no} if $z \notin A$. Informally, whether or not the query string z is in A is determined automatically. We say that the TM asks the query to an oracle for A and the oracle "magically" gives the correct answer in one step. Aside from this ability to ask the oracle about a query string, the TM is otherwise the same as before.

可以重写为

其中上游的 E ' 做如下: 它首先解析 ⟨ , z , z 和 z, 然后对 x 和 y 组成 ⟨ , y。 然后它叫E 与 Qx,y⟩and z。 这说明L是Σp中的一种语言

通常情况下,类似证据通过诱导可以发挥作用,即: k 意味着

oracle的概念在复杂性理论中发挥着不可或缺的作用. 接下来我们讨论oracle 稍后,我们将提出使用oracle的PH的同等特征。

2.4 oracleTuring machine

这个想法是把决定A成员部分的复杂性从机器的复杂性中抽象出来. 机器受自身复杂性的束缚,约束在计算中的其他所有步骤,包括决定要询问oracle的字符串和写下查询. 特别是,如果机器是多诺米时间限定的,它只能对多诺米边界长度进行查询.

解释oracle概念的另一种方式是思考算法. 带有oracleA的算法M表示M A,是一种通常的算法,只是它得到一个“魔盒”,能够回答M在单位时间内就A成员提出的任何问题。 我们把oracle视为一种程序,M可以给程序发出功能召唤. (虽然说来话长,oracle甚至不需要是一个可计算出来的集.)

我们把PA定义为所有语言的集合,这些语言可以在确定多语种时间中计算,并赋予甲种语言的使用权。 国家行动计划的定义类似。 我们可以将这个先知的概念定义为除P和NP之外的其他类别,我们以后将看到这一点。 我们可以进一步概括, 将复杂类作为先知。 例如,如果C是一个复杂类,NPC被定义为,

$$\mathrm{NP}^{\mathcal{C}} = \bigcup_{A \in \mathcal{C}}$$

We discussed Karp reduction (or p-time many-one reduction) before. A different reduction called Cook reduction (or p-time Turing reduction) can be defined via oracles. We say that a language B *Cook reduces* to a language A in polynomial time, if there is a deterministic polynomial time algorithm that computes B, given A as an oracle. We denote this by $B \leq_T^p A$. In other words, $B \leq^p$ $_T^A$ iff $B \in \mathrm{P}^A$.

2.5 A Characterization of PH

2.5 公共卫生特征

THEOREM 2.20 $\mathrm{NP}^{\mathrm{SAT}} = \Sigma_2^p$

Proof. We first show that $\Sigma_2^p \subseteq \mathrm{NP}^{\mathrm{SAT}}$. Let L be a language in Σ_2^p. So there is a polynomial time computable predicate $D(x, y, z)$ such that

证得. 我们首先显示... $x \in L \iff \exists^p y \forall^p z[D(x, y, z) = 1]$

where the lengths of y and z are bounded by $q(|x|)$ for some polynomial q. We can design a NP oracle machine M that can decide L when given an oracle for SAT. Given x, M first guesses a string y, then asks the question "Is it true that, $\forall^p z[D(x, y, z) = 1]$", where both y and z are polynomially bounded by $q(|x|)$. By Cook's Theorem of SAT being NP-complete, this can be transformed into a SAT query. (Strictly speaking an UNSAT query.) The SAT oracle can answer this question. If the answer from SAT is 'no', i.e., $\forall^p z[D(x, y, z) = 1]$", M accepts x. Such a y will exist iff $x \in L$. So M decides L correctly.

We next show that $\mathrm{NP}^{\mathrm{SAT}} \subseteq \Sigma_2^p$. Let $L \in \mathrm{NP}^{\mathrm{SAT}}$ and M be an NP Turing machine that decides L, given access to a SAT oracle. Each path in M may ask many queries to the oracle. We first construct an equivalent NP machine M' that asks at most one SAT question on each computational path, and it makes the query at the end of the path.

Consider a single path in M. This path may ask multiple questions to the oracle. Consider the first such question "$\varphi \in$ SAT?". At this point M' does not ask the oracle a question. Instead, it guesses either "yes" or "no" as a non-deterministic move. On the branch it guesses "yes", it proceeds to guess a satisfying assignment to the queried formula φ. If the guessed assignment satisfies φ, then M' proceeds with the computation of M with q_{yes}. If the guessed assignment does not satisfy φ, then this non-deterministic rejects. On the branch it guesses "no" it simply takes this "no" as the oracle answer and proceeds with q_{no} in M, while remembering that it guessed "$\varphi \notin$ SAT". Subsequent queries to SAT are handled similarly.

考虑M ' 中任何一条一直存活到终点并完成模拟M的路径. 这种路径与M中的路径相似,只是它已经假设某些公式是不满足的. 请注意,它幸存下来的事实意味着,它关于某些公式是符合的假设已经得到核实。 如果M被模拟的路径拒绝,那么M ' 也应该这样做。 现在假设M中的路径被模拟接受. M ' 中的路径必须决定是否接受。 这条路(因为它已经生存)本来可以证明,它所假定的可喜配方确实可喜。 它现在需要核实它所假定的不满意的公式是否确实不满意。 它通过一个单一的SAT查询来做到这一点. 假设公式是1,2... . . .. (原始内容存档于 2018-09-09)., φk. M

询问 \\ 1\ 2\。 . . .. ∨ φ. 注意, 如果上面的公式不能满足, 所有公式都不可满足。 如果oracle确认所有猜测为不能满足的公式确实不能满足,那么这条路径就接受.

现在,如果M有一条接受的道路,M就有一个接受的道路。 注意M ' 的接受路径与M 的一些有效的计算路径相对应,因为所有对询问的猜测答案在M ' 接受之前的结尾都会被验证. 现在,接受M ' 可以表述为`p '

One can extend Theorem 2.20 to other levels of the hierarchy. The above proof gives the

一种可以将定理2.20延伸至等级的其他级别. 上述证据给出了这一想法的要点,即"怀疑、拖延和核实"询问。 然而,为了正式地为各级公共卫生部门开展这项工作,更宜采用某种结构形式主义。 这是对复杂理论的味道, 叫做结构复杂理论。 所给出的论据的笼统性有一定的优雅;从消极的一面看,人们可能会被形式主义所蒙蔽,而忽略了论据的"真正"观点。 在此,尽管存在各种形式主义,争论的"真正"点是"怀疑、拖延和核实"询问。

We define an operator $\exists$.

Definition 2.21 *Given any language L, and a polynomial p, define*

2.21 根据任何语言L和一个多名p的定义

$$\exists.L = \{x \mid (\exists^p y)\langle x, y\rangle \in L\},$$

对于任何C级,

$$\exists.\mathcal{C} = \{\exists.L \mid L \in \mathcal{C}\},$$

where as usual $\exists^p y$ *denotes* $\exists y \in \{0,1\}^{p(|x|)}$.

Similarly we can define the operator $\forall$.

2.22 根据任何语言L和一个多名p的定义

$$\forall.L = \{x \mid (\forall^p y)\langle x, y\rangle \in L\},$$

对于任何C级,

$$\forall.\mathcal{C} = \{\forall.L \mid L \in \mathcal{C}\},$$

where as usual $\forall^p y$ *denotes* $\forall y \in \{0,1\}^{p(|x|)}$.

Clearly co$\forall.\mathcal{C} = \exists.co\mathcal{C}$. Also $\exists.\forall.\mathrm{P} = \Sigma$

我们要证明以下的定理.

定理2.23 对于任何k,

国家进步党 $= \Sigma_k^p,$

LHS上的 " 塔 " 的高度为k(国家邮政局的数)。 这一点也与任何oracle相对。

这一点在一系列索赔中得到了确认。

第1项索赔:关于任何一套oracle,

$$\mathrm{NP}^A = \exists.\mathrm{P}^A$$

It is trivial that $\exists.\mathrm{P}^A \subseteq \mathrm{NP}^A$. The NP machine simply guesses the string in the definition of $\exists$. and does the P^A computation.

Given a language $L = L(N^A) \in \mathrm{NP}^A$ for some NP machine N, we want to put its acceptance criterion in the form of $\exists.\mathrm{P}^A$. It would be quite trivial if N only asks its queries at the end of a computational path. So, we modify N to N', where on each path N' 模拟 N ,并且,每当 N 给出一个查询 N ' 时,只是猜测一个答案. 在N的计算路径末端,N ' 接受iffN接受,所有查询猜测都正确. ♣

第2号索赔:关于任何oracle,

$$\mathrm{NP}^{\exists.\mathrm{P}^A} = \exists.\forall.\mathrm{P}^A$$

One direction is easy, $\exists.\forall.\mathrm{P}^A \subseteq \mathrm{NP}^{\exists.\mathrm{P}^A}$. A simulating oracle NP machine can guess the string in $\exists$., and use its oracle in $\exists.\mathrm{P}^A$ to answer queries in $\forall.\mathrm{P}^A$. Note that oracles in complementary classes have the same power.

Now consider any $L = L(N^B)$, where $B \in \exists.\mathrm{P}^A$. Let $B = \{w \mid (\exists^p y)D(w,y)\}$, where D is in P^A. We wish to express the acceptance of N^B on x in $\exists.\forall.\mathrm{P}^A$. So we will guess ($\exists$) a polynomially long non-deterministic path p, as well as polynomially many answer bits $b_1, \ldots, b_m$, and some y_i (one for each $b_i = 1$ for the i th query "$w_i \in B$?"), such that, ($\forall$) for any z_j (one for each $b_j = 0$ for the j th query "$w_j \in B$?"), the following predicate holds which is computable in P^A— Machine N on x along path p asks some m queries $w_1, \ldots, w_m$ such that for the i th query, if $b_i = 1$ then $D(w_i, y_i)$, if $b_i = 0$ then $\neg D(w_i, y_i)$, and, given the answers being $b_1, \ldots, b_m$, N accepts x. (Note this last use of "some m queries ..." is not a use of quantifiers; the P^A computation simply follows the path p to arrive at its first query w_1, and depending on b_1 and its verification by D, proceed to find its second query w_2, etc.) ♣

当然,自从索赔1和2保留了所有oracle之后,它也保留在我们用任何丙类取代甲时。

第3号索赔:任何甲项索赔,

$$\text{国家民进党} \subseteq \Sigma_k^{p,A},$$

LHS上塔内NP的编号为k.

If $k = 1$ or 2, Claim 3 is direct from Claim 1 and 2. Assume $k > 2$. Claim 1 gives

$$\text{国家民进党} \subseteq \mathrm{NP}^{\exists.\mathrm{P}^{\mathrm{NP}^{\cdot^{\cdot^{\cdot^{\mathrm{NP}^A}}}}}},$$

其中在LHS上NP的"托盘"高度为k,在RHS上P以上的NP数为k − 2. 将索赔 2 应用到 RHS 中

$$\mathrm{NP}^{\exists.\mathrm{P}^{\mathrm{NP}^{\cdot^{\cdot^{\cdot^{\mathrm{NP}^A}}}}}} \subseteq \exists.\forall.[\mathrm{P}^{\mathrm{NP}^{\cdot^{\cdot^{\cdot^{\mathrm{NP}^A}}}}}] \tag{2.1}$$

其中RHS上P上NP的号码为k − 2. 这座塔显然被控制住了
全国民进党
高高K − 1的NP塔,通过诱导载于 Σp,A
然而,新进步党
-1 -1 - 时间 现在"插进去"在(2.1)我们得到

Now the reverse direction 国家民进党

Claim 4: for any oracle A,

$$\mathrm{NP}^{\mathrm{NP}^{\cdot^{\cdot^{\cdot^{\mathrm{NP}^A}}}}} \supseteq \Sigma_{p,A}^{k},$$

现在反方向。

第4项索赔:任何甲状腺

国家民进党

在LHS上NP的“塔”的高度是k。

$\Sigma_{k-1}^{\mathrm{NP}^{\mathrm{NP}^{\cdot^{\cdot^{\mathrm{NP}^A}}}}} \subseteq \mathrm{coNP}^{\cdot^{\cdot^{\mathrm{NP}^A}}}$ where the tower has

塔身高度为k − 1. 但作为oracle,互补的所级拥有同样的力量,这样国家计委才能

Claim 3 and 4 proves Theorem 2.23.

C的塔身高度为K − 1. ♣

第3和第4项索赔证明定理2.23。

2.6 Complete Problems for Σ_k^p

2.6 完整的问题

classes NP and PSPACE (see A_{NP} and A_{PSPACE} defined in Section 2.2). Similarly, one can construct canonical complete languages

我们看到NP和PSPACE两个班级的教条完整语言(见ANP和APSPACE第2.2节定义)。 类似地,人们可以为多诺米亚等级的每个级别构建完整的方言

(单位:千美元) 在本节中,我们为每种语文展示“自然”的完整语言。

和往常一样,我们说,对于“p”来说,一种语言L是完整的(在多语种减时的情况下)。

k 卡普减为L. 然悉的库克定理已经为我们提供了完整的语言,

where f is a (quantifier free) boolean formula over the n variables and $X_1, X_2, \ldots X_k \subseteq X$ is a partition of X. Here, Q is the existential quantifier if k is odd, and it is the universal quantifier if k is even.

We say that F is *valid* if the above expression is logically true. Meaning, there exists some choice of values for the variables in X_1 such that, for all choices of values for X_2, there exists some choice of values for X_3 such that, so on ..., the formula f evaluates to true.

Let QBF_k be the set of all valid quantified boolean formulas with k alternations. It is quite easy to adapt the proof of Cook's theorem to show that for any $k > 1$, QBF_k is

我们说,如果上述表达逻辑上是真实的,F是有效的. 意思是,对于X1中的变量存在一些数值选择,这样对于X2的所有数值选择,对于X3存在一些数值选择,比如. . . .. ,公式f评价为真实。

2.7 Alternating Turing Machines

K -完成。 (英语).

Another way of looking at the Polynomial Time Hierarchy is to use Alternating Turing Machines. An alternating TM is a nondeterministic Turing machine where each state is

k - 完成语。

2.7 替代Turing machine

另一种看待波利诺米亚时间等级的方法是使用AlternatingTuring machine. 交替的TM是一种非决定性的Turing machine,每个状态都在其中

标注为存在或通用。 它的功能与非定型Turing machine的功能相同,但接受的概念定义不同.

回顾普通Turing machine的配置是由Turing machine的当前状态,磁带上的符号,以及头部位置组成的扭矩. 随着Turing machine处理输入,它从一个配置移动到另一个配置. 在一种代用Turing machine中,我们定义存在配置是Turing machine当前状态为存在状态的一种,而通用配置是状态为普遍状态的一种.

接受被递归定义: 如果状态接受或拒绝, 最终配置是接受或拒绝配置。 对于非最终配置,接受的定义是: Turing machine的存在配置,如果在一步中其下一个配置中存在接受配置,则是一种接受配置. 一个通用配置,如果它的下个配置都接受配置,则是一种接受配置. 最后x被接受,如果初始配置是接受配置.

交替Turing machine和限定式交替的关系如下. 证如是等功德.

2.26 定理
k.恰当是多诺时调换Turing machine所接受的一组语言 L , 以 k 的交替配置与初始配置作为存在性配置。 同样,
k是多诺时调换Turing machine接受的一组语言 L , K 交替使用通用的启动配置.

Exercise: Prove Theorem 2.26
锻炼:证明定理2.26

第3章

空间受限计算

章节大纲:配置图和图的可访问性问题. 萨维奇的定理. 伊默曼-舍列普切尼定理.

3.1 配置图

我们首先讨论配置图。 它们对于构思正式的计算过程非常有用,并且经常从具体的图表理论问题的角度实现这个正式的过程. 它们对于研究空间界限计算特别有用。

Let M be a non-deterministic TM and x be an input of length n. At any point during the computation, the *configuration* of M, which determines all the relevant future steps of the computation, can be fully described by specifying the current state, the contents of the tape, and the current position of the tape-head. If the machine M is time $T(n)$ bounded, then this information takes no more than $O(T(n))$ bits. For space bounded computation we use off-line TMs. Let M be a $S(n)$ space bounded machine. Note that for space bounded machine the input tape is read only and its content does not change during the computation. Then we can specify the *configuration* of M at any step by the current state ($O(1)$ bits), the position of the read-only head ($\log n$ bits), and the contents of the work tape together with its tape-head ($O(S(n))$ bits). If $S(n) = \Omega(\log n)$, as we will always assume in space bounded computations in these Lectures, we can describe a configuration using $O(S(n))$ bits.

考虑磁带内容是输入x的配置(如果M不是离线TM,那么这个x后面是与时间约束等长的组合的空白符号),磁带头的位置是0,当前状态是M的起始状态. 这是(独特的)起始配置。 同样自然,任何将当前状态作为M最终状态的配置都称为最终配置. 技术上的TM计算是根据这些配置来定义的. 一个定义配置对上的二进制关系 :

c c′,如果fin 一步c be,可以跟随c′,即从c step到c′的一步有合法举动. 对于定型机器,最多可以在任何c之后有一个下一个配置;对于非定型机器,可以有一个以上. 这样的下一个有效配置的数量完全由过渡函数 \\\\

证得. 与外勤部不同,我们提出以下小意见:从u到v的路径最多,如果有的话;从u到w和w的路径最多,最多,分别是XT/2＋和XT/2＋。 因此,我们将采用递归式的“中第一”搜索。

首先,如果从s可以到达,那么它可以使用长度最多n的路径(实际上n－1就足够了;对于任何路径,我们可以去除任何循环,从而去除任何顶点沿路径重复出现的所有现象)来实现. 然后我们应用上述观察. 我们通过所有可能的选择 一个潜在的中间w。 我们按回转方式检查所需长度n/2的两条路径是否存在. 考虑到两个节点x和y,以及整数t(二进制),如果最多从u到 v.的长度有路径,则以下算法检查.

接触(u, v, t)

If $t = 1$ then
    If $[(u = v)$ or $(\langle u, v\rangle \in E)]$ then *accept*
    else *reject*
If $t > 1$ then
    G的每个节点x,
    如果[Reach(u, x, t/2) QQ(Reach(x, v, t/2)] 接受,如果尚未接受,则拒绝。

We can check if t can be reached from s by calling REACH(s, t, n). One subtle issue is that n has to be a power of 2 for our recursion to work well. If n is not a power of 2, we call REACH(s, t, n'), where $n' = 2^{\lceil \log n \rceil}$.

该算法的复发深度为QQlog n⅂. 因此,递归堆栈必须存储关于logn递归调用的信息. 每一次通话,我们需要存储 u, v, t 和x 在通话中使用。 这需要 O(log n) 空间。 因此,所使用的总空间为O(log n)2。 ♣

我们用上面的定理来证明

THEOREM 3.4 (SAVITCH) *For any space constructible function* $S(n) = \Omega(\log n)$, NSPACE$[S(n)] \subseteq$ DSPACE$[(S(n))^2]$.

证得. 让M成为使用O(S(n)空间的非定型机器. 我们建造一台确定性机器M ' ,利用O(S(n)2)空间模拟M。 输入 x, M `

considers the configuration graph G of $M(x)$. It uses the above algorithm for GAP to test whether any one of the final configurations of G is reachable from the starting configuration. As there are $2^{O(S(n))}$ nodes in G, the space used by M' is $O((S(n))^2)$. One subtle issue is that M' cannot construct the whole G, as that would require exponential space. So, it runs the algorithm for GAP without explicitly constructing the graph. Whenever a query "Is $(x, y) \in E$?" is made, it answers it by checking if configuration y can be reached from x in one step. Space constructibility of $S(n)$ is required as we need to compute the number of nodes in G (for which we should know the value of $S(n)$). ♣

我们接下来显示 GAP 是 NL - 完成(在对数空间减少下)。

DEFINITION 3.5 *A language B is* NL*-complete if $B \in$ NL, and for any language $A \in$ NL, there is a deterministic Turing machine M that uses $O(\log n)$ space such that, for any x, $x \in A \iff M(x) \in B$.*

在这里,我们默认地扩展了空间限制TM的模型,允许它只有一个单向写入输出磁带,这就像它的单向只读输入磁带一样,并不算向空间限制。 (当然很容易显示,如果它有空间约束S(n),输出长度以2O(S(n))为界限,否则计算会循环而不是终止.)

演练:显示对数空间的减少量是在组成下关闭的。

定理3.6 GAP为NL - 完成.

由于GAP在P,显然P在对数空间削减下关闭,因此一个必然结果是

COROLLARY 3.7 NL $\subseteq$ P.

Proof. It is easy to see that GAP $\in$ NL, as we simply guess a path (edge by edge and verify every step as we go) of length up to n.

Let $A \in$ NL via a machine M. We give a machine M' that reduces A to GAP . Given an input x, M' simply outputs the configuration graph of M on x. Clearly M' can be implemented in deterministic logspace. ♣

这个定理的精神如下: 讨论非定型空间的界限计算,而不是抽象的形式主义的自动马塔理论,不失去通俗性,人们可以讨论具体的问题GAP. NL只是GAP,

3.3 Immerman-Szelepcs'enyi定理

我们在此证明 Immerman-Szelepcs'enyi定理: NL在补充下关闭. GAP是NL-Complete, 如果我们证明GAP在NL,

鉴于图 G 和 verties s 和 t ,我们希望在 Logspace 中设计一个非决定性的算法,这样,如果图中没有从 s 到 t 的路径(我们称这样的路径为 s - t 路径),那么在 " 是" 中就有一些计算路径结束。 以"非自然"方式思考GAP。 通常情况下,我们短语算法这样,非决定性分支寻求肯定的答案. 然而,在这种情况下,如果某个分支说是,那么它就有无可争辩的证据,证明不存在S-t路径;此外,如果没有S-t路径,那么一些计算路径应该找到这种证据。

虽然这种"扭"令人惊讶,但从逻辑上讲,这并非自相矛盾。 例如,人们可能会试图在将s和t分开的图中找到一个"切"。 注意如果存在这种切除,则f

从S到T没有路 但我们不能简单地寻找一个切口, 因为(大概)代表切口会消耗太多空间。

让G成为带有n个顶点的图,其中有两个显著的顶点s和t. 算法分两个阶段运行. 在第一阶段,我们计算从起始顶点s到的顶点数。 定义 ci 是可到达的顶点数, 在 i 步内。 我们逐渐计算c0,c1。 . . .. , cn − 1 (中文(简体)). 它们的计算是非决定性的。 从ci到ci + 1,我们假设正确的ci已经计算过了. 尽管如此,在它的磁带上写着ci之后,只有一些计算路径会发现正确的ci + 1并进行. 因此,只有一些计算路径才能发现正确的cn-1并进入下一阶段。 不成功的道路会知道他们不成功而死亡。 在第二阶段,使用cn − 1的值来检查t是否无法从s中达到. 在下面的算法中,把任何幸存的路径想成是“我还没有死的事实意味着我有证据”。 假码以算法1给出. 接下来将描述算法中使用的理念.

We'll incrementally build up counts for $c_0, c_1, \ldots, c_{n-1}$. Finding c_0 is easy: only s is reachable from s in zero steps, and so $c_0 = 1$. c_1 will be the number of vertices for which there is a directed edge from s, to s itself. Suppose we have computed c_i correctly and stored it (only $O(\log n)$ bits are needed to store c_i). We describe how to compute c_{i+1} from c_i.

We undertake the following process for each v, in succession. Our "goal" for each v is to see if we can reach it in at most $i + 1$ steps, and if so we should increment c_{i+1} for it. The way we want to find this out is to "re-experience" the entire set of vertices counted in c_i, and for each such vertex u (which by definition are reachable from s in at most i steps), we check if (u, v) is an edge in G, or $u = v$.

当然,任何计算路径 正确计算ci在过去 遇到过所有这些 ci vertices。 我们没有空间记录这些半圆顶的所有名字。 因此,他们的身份必然被遗忘。 因此,我们通过所有顶点,从1到n, 尝试“再体验”这个顶点列表, 可以在i 步内到达。

因此,对于任何固定v,我们通过如下:

- For each u, we non-deterministically guess whether it was one of the c_i vertices reachable from s in $\leq i$ steps. For any u, if we guess "yes", then we try to validate our guess. We do this by non-deterministically guess a path of length at most i from s to u. If the validation succeeds, u is reachable from s in $\leq i$ steps. Here's the moment of truth from this u: if there is an edge from u to v in G, or $u = v$, we increment c_{i+1}. If the validation fails, i.e., this particular guessed path did not reach u, then our guess was wrong *somewhere*. It could be wrong either because u is not one of the c_i vertices reachable from s in i steps, or our guessed path is not a valid path (even though some such path might exist). In any case some guess along this path is wrong, so halt and die! If we guess that u is not reachable from s in i steps, do nothing and move on to the next u. In this whole process, we keep a count d of the number of vertices u for which we guessed "yes".

- Finally, after processing all u, check if $d = c_i$. (This is the "quality check": after all one can guess a lot of "no" and avoid all the validation and survive.) If $d = c_i$, we have accounted for all the c_i number of vertices that are reachable from s in i steps, since we have validated all the "yes" guesses. We conclude that all the "no" guesses were correct as well. Therefore, if a computational path lived thus far, and got $d = c_i$, then indeed we have "re-experienced" all the vertices reachable from s in i steps. It follows that we have incremented c_{i+1} for v along this computational path iff v is among the c_{i+1} vertices from s in $\leq i + 1$ steps. On the other hand, any path with $d \neq c_i$ simply halts and dies!

Algorithm 1: Shows $\overline{\text{GAP}} \in \text{NL}$
Input: $\langle G, s, t\rangle$: where, $G = (V, E)$ is a directed graph over n vertices, $s, t \in V$
Output: *accept* $\Longleftrightarrow$ there is NO path from s to t in G
IMMERMAN-SZELEPCSÉNYI(G, s, t)

```
(1)   c_0 = 1
(2)   for i = 0 to n − 2
(3)      c_{i+1} = 0
(4)      foreach node v ∈ V
(5)         d = 0
(6)         foreach node u ∈ V
(7)            Guess yes or no: if no, skip the next steps (8) to (12):
(8)               Guess a path of length i from s and if u is not among the
                  vertices on this path, then reject
(9)               d = d + 1
(10)              if [(u, v) ∈ E] ∨ [u = v]
(11)                 increment c_{i+1}
(12)                 goto 6 with next v
(13)        if d ≠ c_i then reject
(14)  d = 0
(15)  foreach node u ∈ V
(16)     Guess yes or no: if no, skip the next steps (17) to (19):
(17)        Guess a path of length n − 1 from s , and if u is not among the
            vertices on this path, then reject
(18)        if u = t then reject
(19)        d = d + 1
(20)  if d ≠ c_{n−1} then reject
(21)                  else accept
```

注意: 许多非决定性的路径会在计算中"死"！猜一个你不当的候选资格,任何v, 将会导致死亡,

然后走错误的道路。 然而,一些猜想的顺序会正确运行这个格鲁莱特一路,我们将会有一个正确的计数cn-1.

我们用它来回答GAP问题。 现在的想法是一样的: " 重新体验 " 所有cn-1的顶点,看看是否在顶点中。 因此,我们通过所有的顶点和(非决定性的)努力计算从s再次到达的cn-1顶点。

For each node $v \in V$, we do the following: We guess whether there's a path of length $\leq n-1$ from s to v. If the guess is "no", move on to the next vertex. If the guess is "yes", we verify it by guessing a path of length $\leq n-1$ and check if v is encountered along the path. If the verification fails, die! If it succeeds, increment a counter d'. At the "moment of truth" when a vertex v was verified to be among the c_{n-1} vertices reachable from s, we check if it is t.

Finally, we accept $\langle G, s, t\rangle$ iff in the end $d' = c_{n-1}$ and t was never marked as one among the c_{n-1} vertices.

请注意,在上面的伪码中,我们做了一些小的修改. 在第12行中,在我们确认v应算在ci + 1之后,我们立即着手进行下一个v'。 同样在第(18)行中,我们一旦确认基础设施服务可以达到,就拒绝。 一个微妙的点:在假码中,一个有某些错误猜测的路径必须死,这已经不再是这种情况了. 从概念上来说,按照上面的描述来思考是比较容易的: 所有的猜测最终都会通过某种方式被验证,任何做出错误猜测的计算路径最终都会死. 整个过程似乎只是一次又一次地重新体验一些步骤内可以到达的顶点的一组,我们做了计算各种计数ci的重要任务,以及t是否在cn − 1之间几乎是事后思考. ♣

3.4 波利诺米空间

我们用这一节来讨论多边计算空间。 让我们首先比较确定性和非确定性多边计算空间。

回顾PSPACE由所有语言组成,由在多诺米空间运行的定心Turing machine计算. 人们可以考虑类似的非定型类,即NPPACE,它由在多诺米空间运行的非定型Turing machine可以计算的所有语言组成。

不同于时间范围计算,其中的P vs. 在空间界限计算领域,NP是开放的,类似PSPACE vs. 核动力源PACE很容易解决。 以下定理是萨维奇定理的必然结果.

THEOREM 3.8 *PSPACE = NPSPACE.*

由于上述定理,我们几乎不谈非定型多诺米空间定界机的概念。

我们的目标是本节其余部分的两面。 我们首先为PSPACE展示完整的语言. 然后,我们比较交替的Turing machine和PSPACETuring machine.

在第2.2节中,我们为PSPACE提供了一种犬类完整的语言(称为APSPACE)。 在这里,我们展示了一种"自然"的PSPACE语的完整语言。

3.9 定义 让 f 成为 n 布尔变量 x1, x2 之上的布尔公式。 . . .. ,xn. (中文(简体)). 由 f 产生的量化布尔公式F由 :

$$F = \exists x_1 \forall x_2, \ldots, Q x_n [f(x_1, x_2, \ldots, x_n)].$$

Here, $Q = \exists$ *if* n *is odd and* $Q = \forall$*, if* n *is even. We say that* F *is* valid*, if the above statement is logically true. Meaning, "there exists some choice of* x_1 *(namely,* $x_1 = 1$ *or* $x_1 = 0$*) such that, for all choices* x_2*, there exists some choice of* x_3 *such that, so on . . ., such that, the formula* f *evaluates to true under the truth assignment* $(x_1, x_2, \ldots, x_n)$*.*

定义3.10(量化框架)

$$\text{QBF} = \{F | F \textit{ is a valid quantified boolean formula}\}$$

实例。 考虑量化布尔公式

$$F = \exists x \forall y \exists z [(x \vee \neg y) \wedge (\neg y \vee z) \wedge (\neg x \vee y \vee \neg z)]$$

F is a valid formula. To see that, let $x = 1$, and for $y = 0$ we can set $z = 0$ and for for $y = 1$ we can set $z = 1$. Hence, $F \in$ QBF.

请注意,尽管我们把量化框架定义为严格交替的限定词,从存在限定词开始,但这些要求并不重要。 我们总是可以附加假变量来适应这一点,而不增加表达式的大小太多(最多是大小的两倍).

我们的目标是显示QBF是PSPACE-Complete. 作为第一步,我们证明QBF是在PSPACE. 我们提出了一个解决QBF的递归程序。

SolveQBF(F) 执行下列案件之一。

1. 国家 例1(F是一个公式大于零变量):

- 如果F是“1”的公式,则返回(F是有效的)。
- 如果F是“0”的公式,则返回(F无效)。

2. Case 2 (F starts with an existential quantifier): Let F start with "$\exists x....$". Define two quantified boolean formulas F_1 and F_2 by substituting $x = 0$ and $x = 1$ in F, respectively. Declare F to be valid if at least one of F_1 or F_2 is valid. We can check this by making two recursive calls SolveQBF(F_1) and SolveQBF(F_2).

3. Case 3 (F starts with an universal quantifier): Let F start with "$\forall x....$". Define two quantified boolean formulas F_1 and F_2 by substituting $x = 0$ and $x = 1$ in F, respectively. Declare F to be valid if both F_1 and F_2 are valid. We can check this by making two recursive calls SolveQBF(F_1) and SolveQBF(F_2) (after SolveQBF(F_1) use one bit the remember the result, and then reuse the space in the second call of SolveQBF).

注意公式F1和F2的变数比F少一个. 因此,如果我们从一个公式开始计算n变量,递归树的深度将是n. 所以空间的总使用量是多诺米的. 简言之,我们可以简单地在多诺米空间(实际上为线性空间)中,对F定义的深度n的二进制树进行树-绕行. 我们证明了以下的定理。

定理3.11 SPACE中的QBFis.

我们的QBF程序是递归式的. 我们经常在写空间高效算法时使用递归法. 我们又看到另一例递归程序, 一般来说,递归程序使用的空间量为O(d) s,其中d是递归水平的深度,s是每递归水平的当地空间使用量。 应该清楚的是,人们可以通过使用堆栈很容易地破解复发,并使用同样量的空格O(d) s获得非复发程序。 但是,通常比较容易重复地写出这些程序。

接下来我们显示QBF是PSPACE硬的。 让 M 成为带有空间约束 p(n) 和 x 的机器成为输入。 X上的M的计算由大小2O(p(n))的配置图G捕获. 让c0和ccc成为M的初始和接受配置. (我们也许可以假设有一个独特的 M. 的接受配置。) 然后, 我们想要检查从 c0 到 cacc 最多 2O(p(n)) 是否有长度路径。 我们要将上述问题表述为量化布尔公式。

让我们在两个任意的组合中普遍这样做。 让 c1 QQ2k c2 表示从配置 c1 到配置 c2 最多有 2k 长度路径。 我们可以用萨维奇定理中的想法来表达上述标准, 回顾,如果而且只有在有配置c ' 这样的条件下,有从c1到c ' 和c ' 到c2的长度最多2k-1的路径存在,那么这种路径就存在:

$$c_1 \vdash_{2^k} c_2 \iff \exists c'[(c_1 \vdash_{2^{k-1}} c') \wedge (c' \vdash_{2^{k-1}} c_2)]$$

我们可以在方括号中递增表达式。 但是,这种方法最终会成指数大小的公式(因为方括号中的表达涉及两个递归调用). 我们玩个小把戏 我们可以将标准表述如下。 标准

$$(c_1 \vdash_{2^{k-1}} c') \wedge (c' \vdash_{2^{k-1}} c_2)$$

等同为

$$\forall(\text{ configurations } u \text{ and } v)[(\ (u = c_1 \text{ and } v = c') \vee (u = c' \text{ and } v = c_2)\ ) \Longrightarrow (u \vdash_{2^{k-1}} v)].$$

The advantage of this transformation is that the later expression used only one syntactic expression of $u \vdash_{2^{k-1}} v$. We can expand it recursively. At the base case $c_1 \vdash_1 c_2$ for $k = 0$, it can be expressed in a quantifier free boolean expression as in Cook's Theorem. Thus for $c_1 \vdash_{2^{O(p(n))}} c_2$ we obtain in polynomial time a quantified boolean formula F such that the machine accepts x if and only if F is valid.

由于配置图有2O(p(n))顶点,最终公式为长度O(p(n)). 它只使用多名数的修饰符. 从技术上讲,最后公式并不完全符合我们量化布尔公式的定义(定义2.24)所要求的形式。 但是,F可以通过使用标准的逻辑等价来方便地转换成所需的形式. 我们已经证明,这是按照定理。

3.12 QBF定理是PSPACE-Complete.

Theorem 3.11 gives us an easy mechanism to compare PH and PSPACE. Recall that,

k-完成(定理2.25)。 很容易看出QBF(无限制的交替数)是QBFk(只能有一些恒数k交替数)的统称. 因此,作为定理3.11的必然结果,我们有以下定理(无论怎样,当你从树上转动的角度考虑一下,在 p中通用语言时,这个定理是相当明显的).

k 由修饰词定义,或用 k- 转录TM.)

THEOREM 3.13 *For any* $k > 0$, $\Sigma_k^p \subseteq$ PSPACE. *Thus,* PH $\subseteq$ PSPACE.

3.13 任何k > 0, +p

We can now extend Figure 2.1 to Figure 3.1.

我们现在可将图2.1扩大到图3.1。

3.4.2 APTIME = PSPACE

Our next goal is to connect PSPACE machines and alternating Turing machines (see Sec-

我们的下一个目标是连接PSPACE机器和交替Turing machine器(见第2.7节)。 我们已经看到,交替的Turing machine与常数交替的Turing machine在多诺分级的不同层次上都接受这些语言(定理2.26)。 在此,我们将考虑采用无限制的交替次数的多时自动取款机。

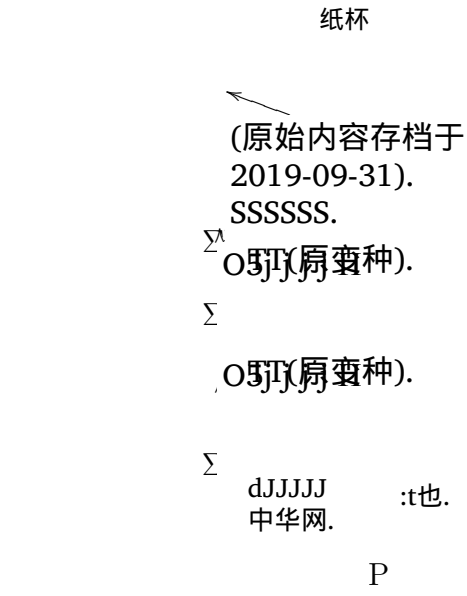


图3.1: 波利诺米时间等级和PSPACE

We say that a language is in the class APTIME, if it is accepted by an alternating Turing machine running in polynomial time. These machines are allowed to use arbitrary number of alternations. But, of course, as the machine runs in polynomial time, it can make at most a polynomial (in input length) number of alternations. We proceed to prove that alternating polynomial time and deterministic polynomial space are equal in power. In other words, we prove that APTIME=PSPACE.

First, let us show that PSPACE $\subseteq$ APTIME. One can give a proof using ideas from Savitch's theorem, similar to that of proving the PSPACE-hardness of QBF. But, as we have already shown that QBF is PSPACE-hard (Theorem 3.12), we will use it to prove the claim. Given a PSPACE machine M and an input x, our APTIME machine first deterministically computes a quantified boolean formula F using the reduction given by Theorem 3.12. So, the machine M accepts x if and only if F is valid. Validity of F can easily be tested in APTIME. Without loss of generality, assume that F is in the standard form given by Definition 3.9. Let F be over n boolean variables. The machine will start in existential mode and choose a value for x_1 (either 0 or 1). Then, it will enter universal mode and choose a value for x_2. In this way, we continue by alternating between existential and universal modes and choose values for $x_1, x_2, \ldots, x_n$. Finally, each computational path will have a full truth assignment in hand. The path will evaluate the underlying boolean formula on this truth assignment. If the evaluation returns "true" then this path will *accept*, else *reject*. From the definition of alternating Turing machines, it is clear that the machine accepts F if and only if F is valid. (One can think of a computation graph of our APTIME machine as a complete binary tree of depth n, where the kth level will have 2^k nodes labeled x_k. Nodes in odd level will be existential and those in even levels will be universal. Each of the 2^n leaves will correspond to a full truth assignment and the leaf would evaluate the underlying boolean formula on

)我们证明了以下说法。

THEOREM 3.14 PSPACE $\subseteq$ APTIME.

We next show that APTIME $\subseteq$ PSPACE. Essentially, we consider the configuration graph of the APTIME machine and do depth first search to find out if its starting configuration is labeled "accept". Our PSPACE procedure is similar to the recursive procedure used to show that QBF is in PSPACE (Theorem 3.11).

我们现在再详细谈谈。 让M成为APTIME机器,X成为输入. 每个配置可以存储在多诺米空间中. 我们写一个程序,将配置c作为输入,输出它是否是接受配置. 我们使用复用 并且只是遵循旋转Turing machine的定义。 当c是一个叶子配置时,很容易。 我们只是检查c是否为接受状态(如M描述中所描述的). 如果是的话,c是接受配置,否则就是拒绝配置. 如果输入c是一个非叶配置. 考虑每个孩子 c ' ,一个一个。 递归,找出C ' 是接受或拒绝。 如果c是一个存在性配置,那么c正在接受,如果并且只有在它的孩子至少有一个接受. 同样,如果c是一个通用的配置,那么c会接受如果并且只有在它的所有子孙都接受配置时.

利用上述程序,可以发现M的起始配置是接受还是拒绝. 由于M在多名制时间运行,我们的程序只需要多名制的复发水平。 每个关卡只需要多名的空间。 因此,整个程序都是在多边空间进行的。 我们证明了以下说法。

THEOREM 3.15 PSPACE $\subseteq$ APTIME

结合定理3.14和定理3.15,我们得出以下结论.

THEOREM 3.16 APTIME = PSPACE

第4章

非一致复杂性

章节大纲:波利诺米环路,P/poly,Sparse套路. 三者等义. 卡普-利普顿定理. 马哈尼定理.

4.1 聚诺电路、P/poly、Sparse电路

如第1讲座中所讨论的,计算复杂论起源于计算论. 这个理论基于图灵的通用计算机概念,这个概念是由严格有限的规则所指定的。 这一理论的一项主要成就是表明存在无法断定的问题. 虽然这在数学和哲学上都很重要,但一旦我们吸收了严格限量装置的概念,试图计算Turing machine公司提出的无数个问题,但事实上,有些计算问题,没有一个TM能够正确计算出所有情况,就不那么令人惊讶了。

If we carry out a "delayed" diagonalization proof, one can easily produce undecidable problems much like the Halting Problem, which nevertheless have very simple combinatorial structures. For example, we can have undecidable problems for which there are at most one string per length n. We can even make it much much sparser; all we need to do is to reject all strings at length n except for the string 1^n, and only for a very sparsely distributed set of values of n's (say, $n = 2^{2^k}$), we carry out a "delayed" diagonalization (say, against the machine M_k). Combinatorially, this set is rather simple: we can, for example, represent this set at length n by $x_1 \wedge \overline{x_1}$ (when it is the empty set $\emptyset$) or $x_1 \wedge x_2 \wedge \ldots \wedge x_n$ (when it consists of just 1^n). Of course which case it is for each n is undecidable computationally.

然而,通过简单的计数参数,22n布尔函数大多来自{0, 1}n

至{0, 1}没有如此简单的表述。 因此,似乎存在着一种不同的复杂程度,而这种复杂程度并没有在TM模型中对可计算性的考虑中得到考虑,这个小盒子试图决定所有可能的长度输入。 此外,在

今天的复杂社区,大多数研究者认为,NP-完成问题似乎抵制任何多诺时间解决方案的问题的关键不是因为缺乏一个试图计算无数情况的严格限定性装置,而只是因为问题过于“复杂 ”。 “复杂性”这一概念是指任何固定大号的有限数目(2n)。 直觉的感觉是,对于NP完成问题等明显困难的问题,对于大n来说,解决所有长度n的例所需的逻辑操作数量只是比任何固定的多诺米在n中增长得更快,也许更可能是n中的指数函数. 当然,这些仍然是没有证据的推测。

为了表达这种直觉,需要界定一个“Combinatory”复杂性的不同概念,这一概念将注意力集中在有限长度n上,并忽略了一个人是否可以用一个“单一”的限定装置,如TM,将其全部“串在一起”的问题。 这导致了非统一性复杂性的概念。

我们考虑布尔电路 n 输入位上的布尔电路是一个定向环形图(DAG),每个节点的标签要么是输入变量xi,要么是逻辑门. 我们可以在不失去一般性的情况下,假设这些大门已经构成一个完全的逻辑基础。 (实际上, , , 或 , , , ,) 已经 , 但通常我们包括所有三个。) 默认的假设是, 由 , 或 , 或 , 或 , 标有 , 或 , 标有 , 或 标有 , 或 标有 , 标有 , 或 标有 , 标有 , 或 标有 , 标有 , 标有 , 或 标有 , 标有 , 或 标有。 有时我们允许 , , , , , , , ,。 一个布尔电路可能有一个或多个节点标记为输出门(s). 默认是一个单一输出节点. 然后一个布尔电路计算出一个布尔函数,从明显意义上讲。 布尔电路的大小是图中的边数(线). 如果在输入和输出节点以外的每个节点的外度是1(a.k.a. 内部节点),然后布尔电路称为公式. 在这种情况下,布尔电路本质上是树,除了输入级. 根据De Morgan定律,人们可以将 QQ 门推到公式的输入级别,而不增加其大小(或者最多增加 + n,如果不是所有否定变量都存在的话). 在电路或公式中,当所有否定符号都直接出现在变量上时,有时连接 XXX 及其相应的变量的边缘不按大小计算,即我们按字数x1,x2的输入值计算大小. . . .. , xnx1, x2, (英语). . . .. ,xn. (中文(简体)). 这种将 门推向输入级的过程,一般可以用于布尔电路,最多可以增加2倍. (复制每个节点 v 并附上副本用于 v.) 布尔电路的深度是从输入节点到输出的最长路径.

布尔函数 fn :{0, 1}n\\\0, 1} 总是用布尔电路和布尔公式计算; 分离正常形式 (DNF) 或连接正常形式 (CNF) 都会实现。 这些是深度2(在恒深电路中,我们不计算连接QQ门与输入级变量的线的深度,即我们按字数计算输入深度)。 然而,就大小而言,很容易显示许多布尔函数在n中具有DNF和CNF大小的exponetial.

练习: **n** 位上的对等函数

$$\oplus : \{0,1\}^n \to \{0,1\}$$

where $\oplus(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} x_i \bmod 2$ requires DNF and CNF size 2^{n-1}.

布尔函数 fn 的电路大小 C(fn) 是最小的布尔函数的大小,它计算 fn。 同样,人们也可以定义fn的公式大小(并且这些定义可以针对各种其他计算模型进行:例如,对于任何深度k,我们可以考虑fn的深度k大小,或者如果我们只考虑没有XQ门的单通电路,那么我们可以定义任何单通函数fn的单通电路大小.)

The class of problems we will focus on consists of Boolean function families $\{f_n\}_{n=1,2,\ldots}$ with $C(f_n) = n^{O(1)}$, i.e., function families which can be computed by circuit families with size bounded by a polynomial in n. A language L is said to have polynomial size circuits, if its family of characteristic functions at length n $\{L^{=n}\}$ does. A complexity class is said to have polynomial size circuits, if every language in the class does.

4.1 P定理具有多诺米大小的电路.

Proof. The proof is simple. Take any TM with time bound $T(n)$, and input x of length n. Then one can write down a square array of $T(n) \times T(n)$ cells, where each cell (i, j) refers to the computation of $M(x)$ at time step i and memory location j. It takes $O(1)$ gates to describe the following: the state M is in at time i, whether the tapehead is at location j at time i, and what content is in tape square j at time i. For $i = 1$ this is determined by the initial configuration. For $i > 1$, each cell (i, j) is determined by cells $(i-1, j-1)$, $(i-1, j)$ and $(i-1, j+1)$. For example if the tapehead is not at these three locations at time $i-1$, then the cell at (i, j) should remain the same as $(i-1, j)$; if the tapehead is among the three locations at time $i-1$, then the cell at (i, j) should be decided according to the finitary transition function of M. In any case locally it is computed by a Boolean function of size $O(1)$. Thus the circuit which simulates the whole computation of $M(x)$ has size at most $O(T^2(n))$.

请注意,这种电路可以通过对数空间计算来构建. ♣

这个定理的一个推论是,电路值问题(CVP)在对数空间减值下是P-完成的.

Circuit Value Problem: Given a Boolean circuit C_n on n inputs and an input of n bits $(b_1, b_2, \ldots, b_n)$, is $C_n(b_1, b_2, \ldots, b_n) = 1$?

缩写4.2 CVP是P-完成的.

一项复杂度量的有力性表现在小的变化中,其变化无常。 我们注意到,例如,无论我们使用多磁带还是单磁带TM,还是各种其他模型,P类都保持不变。 这种拥有多鼻尺寸电路的概念也相当有力。 它并不取决于我们选择的特定逻辑基础函数。 也不改变,如果我们要计算节点的大小, 或者允许无限制的扇形,或者推

所有否定 \\\\

Conversely, If $L \in \mathrm{P}/poly$ via advice function l, we can encode the advice string $l(n)$ in a sparse oracle. E.g.,

$$S = \{\langle 1^n, p_n\rangle | p_n \text{ is a prefix of } l(n), n \geq 0\}.$$

Given S, we can "suck" out the bits of $l(n)$ as follows: By querying $\langle 1^n, 0\rangle$ and $\langle 1^n, 1\rangle$ one can find the first bit of $l(n)$. Having found a prefix p of $l(n)$, we query $\langle 1^n, p0\rangle$ and $\langle 1^n, p1\rangle$, and we can extend the prefix p by one more bit (or we found that $p = l(n)$ is already fully extended.) ♣

The class P/poly is the non-uniform analog of P. It contains the class P, as well as non-recursive sets. Since it also includes non-recursive sets, it is not possible to include the class P/poly in any uniform complexity classes, such as EXP. However, the intuitive feeling is that sets in P/poly are "easy", in a combinatiorial sense. The belief is that NP $\neq$ P because NP $\not\subset$ P/poly. This remains the most concrete approach to a proof of NP $\neq$ P.

然而,在P/Poly类中,能否列入比P更统一的复杂类? 如果NP / Poly 呢? (NP / Poly / Poly) (NP / Poly / Poly / Poly / Poly / Poly). 一个不太可能的复杂性理论后果会提供证据表明NPQQQP/poly确实存在. 这是著名的卡普-利普顿定理的内容.

4.2 卡普-利普顿定理

In this section, we prove the classical Karp-Lipton theorem. It deals with the question of whether NP $\subset$ P/poly? It is believed that NP $\not\subset$ P/poly. Karp-Lipton theorem gives some evidence in this direction. It says that if NP $\subset$ P/poly, then PH $= \Sigma_2^P \cap \Pi_2^P$.

By NP-completeness, NP $\subset$ P/poly is equivalent to SAT $\in$ P/poly. Also by the equivalent formulations of P/poly, this is equivalent to the assertion that SAT(or any other NP-complete language, or every NP-complete language, or every language in NP) has polynomial size circuits. It is also equivalent to the assertion that SAT is polynomial time Turing reducible to a sparse set S: SAT $\in \mathrm{P}^S$.

Self Reducibility: The main idea is self-reducibility of SAT. Suppose we are given a circuit C that is suppopsed to compute SAT for all Boolean formulae of length at most n. It may or may not compute SAT correctly. It may make errors in two ways. It may reject a $\varphi \in$ SAT, and/or it may also accept a $\varphi \notin$ SAT. We can convert C into a circuit C', that does not make errors of the second type. C' works by using C as a black box. Upon input φ over the variables, $x_1, \ldots, x_m$, it feeds φ to C. If C rejects, C' rejects. If C accepts, C' tries to find a satisfying truth assignment σ to φ. To do that, C' converts φ to φ_1, by setting all the occurrences of x_1 in φ to be TRUE. And asks C if φ_1 is satisfiable. If so set $\sigma(x_1)$=TRUE, else set $\sigma(x_1)$=FALSE. Now repeat this process, with φ_1 in place of φ and x_2 in place of x_1, to get the value for $\sigma(x_2)$. Continue this process for the other variables we find a truth assignment σ. Then we check whether σ really satisfies φ, if so accept, else

拒绝。 如果C确实是SAT的正确线路,C ' 也是正确的线路。 但是,无论C是否是SAT的正确电路,C ' 只可能在一边出错. 它可能会拒绝一种令人愉快的公式。 但是,它永远不会接受一种令人不满意的公式,因为它只有在它核实了令人满意的真理任务时才会接受。 这种将问题实例(此处为SAT)简化为较小“大小”问题实例(此处为未分配的变量数目)的过程称为自我减少。

THEOREM 4.7 (KARP-LIPTON) *If* NP $\subset$ P/poly, *then* PH $= \Sigma_2^P \cap \Pi_2^P$.

Remark: Originally Karp and Lipton proved the collapse to the third level of the Polynomial
注释:最初是卡普和利普顿证明崩溃为波利诺米亚级的三级. 他们在论文中指出,**Sipser**将其改进为**PH**二级. 我们给出的证据 使用一个想法 首先被**Hopcroft**注意到。

Proof. The assumption implies that SAT has polynomial circuits. Then, we show that
证得. 该假设意味着SAT有多诺米电路. 然后,我们显示 QQP

where $|y|$ and 2 , (英语). 所以,对于一些P时间 上游D,我们有 convert the question of

$$\exists^p z[D(x, y, z) = 0]$$

在那里, 和 \\ 和 \ \ \ \ \ 和 \ \ \ \ \ 和 \ \ \ \ \ 和 \ \ \ \ \ \ 和 \ \ \ \ \ \ 和 \ \ \ \ \ 和 \ \ \ \ 和 \ \ \ 和 \ \ \ \ 和 \ \ \ \ \ 和 \ \ \ nputable \ 和 \ \ \ \ 和 \ \ \ \ 和 \ \ \ 和 \ \ 和 \ \ \ \ 和 \ \ \ 和 \ \ \ \ 和 \ \ 和 \ 属于 \ \ \ \ \ \ \ \ 的 \ 和 \ \ 的 \ 的。 我们可以把问题转换成

$$x \in L \Longleftrightarrow \forall^p y[\varphi(x, y) \in \text{SAT}]$$

, computation
通过库克定理进入SAT问题。 意思是,我们有一个多边计算时间的函数 (.,.)

work correctly for SAT for all formulas of length at most $p(n)$. Our Σ_2^P computation, in its "there-exists" phase, guesses a circuit C. Then uses self-reducibility to convert it into a one-sided error circuit C'. Now it enters the "for all" phase and checks whether

$$\forall^p y[C'(\varphi(x, y)) = 1]$$

2 计算,在其“存在”阶段,猜测是C线。 然后使用自减法将其转换为单侧误差电路C'. 现在它进入“为所有人”阶段,并检查是否

circuit for SAT. The corresponding C' will also be a correct circuit for SAT. As $x \in L$, $\forall^p y$, $\varphi(x, y)$ will be satisfiable and our C' would accept all of them for these polynomially bounded y's. So we see that,

$$(\exists^p C)(\forall^p y)[C'(\varphi(x, y)) = 1]$$

Suppose $x \notin L$. Then, there is some y_0, polynomially bounded in length, for which $\varphi(x, y_0)$ is not satisfiable. So, for all the guesses C, the corresponding C' would reject $\varphi(x, y_0)$, because C' never accepts any unsatisfiable formula. Thus, we see that,

$$(\forall^p C)(\exists^p y)[C'(\varphi(x, y)) = 0]$$

更正式地定义以下确定性多诺算法A(x,C,y):它首先使用自减性从C计算C'. 然后,它使用库克定理计算公式 \(x,y). 产出C'(X(x,y))。 这么说吧

$$x \in L \iff (\exists^p C)(\forall^p y)[A(x, C, y) = 1]$$

♣

4.3 马哈尼的定理

In the last section, we discussed Karp–Lipton theorem, which says that if SAT has polynomial size circuits, then the polynomial hierarchy collapses to the second level. In other words, if SAT Cook reduces to some sparse set S, then the hierarchy collapses to the second level. A natural question at this point is, what happens if SAT Karp reduces to a sparse set. Notice that this is a stronger assumption. Mahaney's theorem says that if SAT Karp reduces to a sparse set, then NP=P. The theorem also has implications in the context of Berman-Hartmanis conjecture, which we will discuss later. In this section, we prove Mahaney's theorem. This proof follows the idea of Ogihara and Watanabe.

DEFINITION 4.8 *Let $\sigma = \sigma_1\sigma_2 \ldots \sigma_m$ and $\tau = \tau_1\tau_2 \ldots \tau_k$ be binary strings. We say σ is to the left of τ (denoted $\sigma \preceq_l \tau$) if either σ is an extension of τ, or the first bit from the left where they differ σ has bit 0 and τ has bit 1. Formally, let $i = \max\{j \mid 0 \leq j \leq \min\{k, m\}, (\forall 1 \leq j' \leq j)[\sigma_{j'} = \tau_{j'}]\}$. Then $\sigma \preceq_l \tau$ if either $i = k$ or $i < k$ with $\sigma_{i+1} < \tau_{i+1}$.*

We can view the above definition pictorially as follows. Consider the binary tree of partial assignments to the n variables. Let σ and τ be two partial assignments, $\tau = \tau_1\tau_2 \ldots \tau_k$ and $\sigma = \sigma_1\sigma_2 \ldots \sigma_m$. σ is to the *left* of τ, if σ is below τ (i.e. τ is a prefix of σ) or σ is in a branch to the left of τ. The order $\preceq_l$ is the tree traversal order of Left-Right-Root.

We next define a language called *left cut* of SAT. Let φ be a formula and τ be a partial assignment. The pair $\langle \varphi, \tau \rangle$ is in L_{SAT}, if some assignment σ to the left of τ satisfies φ. Formally,

DEFINITION 4.9 *The leftcut set of* SAT *is the set* $\text{L}_{\text{SAT}} = \{\langle \varphi, \tau \rangle \mid \varphi$ *is a formula on* n *variables and* $\tau \in \{0,1\}^k, 0 \leq k \leq n$, *such that* $\exists\ \sigma \in \{0,1\}^n$ *with* $\sigma \preceq_l \tau$ *and* $\varphi|_\sigma =$ *True.*$\}$

Clearly $L_{\text{SAT}} \in \text{NP}$. The crucial property of L_{SAT} is the following. Let φ be a formula and σ and τ be two partial assignments, such that $\sigma \preceq_l \tau$. Then, if $\langle \varphi, \sigma \rangle$ is in L_{SAT}, then $\langle \varphi, \tau \rangle$ is also in L_{SAT}. In particular if $\varphi \in \text{SAT}$, then $\langle \varphi, \epsilon \rangle \in L_{\text{SAT}}$.

Suppose we are given $\langle \varphi, \sigma \rangle$ and $\langle \varphi, \tau \rangle$ and we have to bet on one of these to be in L_{SAT}. Clearly, we should place our bet on $\langle \varphi, \tau \rangle$, if $\sigma \preceq_l \tau$. In its contra-positive, if at least one of them is out of L_{SAT}, then we know $\langle \varphi, \sigma \rangle$ is out. The proof hinges on this property of L_{SAT}.

THEOREM 4.10 (MAHANEY) *For any sparse set $S \neq \emptyset$,* $\text{SAT} \leq_m^p S \iff \text{P} = \text{NP}$.

Proof. Let S be any non-empty sparse set. The implication $\text{P} = \text{NP} \Longrightarrow \text{SAT} \leq_m^p S$ is trivial, so we need only prove that $\text{SAT} \leq_m^p S \Longrightarrow \text{P} = \text{NP}$. We shall design a polynomial time algorithm to solve SAT. First of all, notice that L_{SAT} is in NP. (Given $\langle \varphi, \tau \rangle$, we guess a (full) truth assignment σ, then verify that $\sigma \preceq_l \tau$ and σ satisfies φ; if so accept, else reject.) From the assumption $\text{SAT} \leq_m^p S$, it follows that $\text{L}_{\text{SAT}} \leq_m^p S$, via some polynomial time computable function f.

We will design a polynomial time algorithm for SAT. Let φ be the input formula over n variables and we [illegible], the algorithm would, in fact, output the (lexicographically) left-most satisfying assignment.

m S,通过某种多诺时计算函数f.

我们将为SAT设计一个多名时间算法。 让 \\ 成为 n 变量的输入公式, 我们需要检查它是否匹配。 如果

is assigned a value). Nodes of the tree correspond to partial assignments. We will do a

让我们来考虑一下由... ...任务形成的二进制树。 此树的根对应空任务(由空字符串 QQ 表示, 其中没有变量被指定值)。 该树的节点对应部分任务. 我们先从根开始 搜索这棵树 这棵树有无数节点,我们不能指望进行全面搜索。 相反,在我们前进的同时,我们将把树推倒,只探索树的部分地区. 在任何时间,我们将只维持多个节点。 我们将确保,如果

in polynomial time, its output on these strings can be at most polynomial in n. Let this number be l. The sparse set can have at most N strings of length $\leq l$, where N is polynomial in l, and thus also a polynomial in n. As we explore the binary tree, level by level, we will maintain at most N nodes, at any point of time.

First, we explore the tree until we reach a level k, where the number of nodes $2^k > N$. At this point, we start pruning. We run the reduction f on all these 2^k partial assignments (with φ as the formula) and obtain 2^k output strings. We first look for duplicates. Let σ_1 and σ_2 be two partial assignments at level k such that $f(\langle \varphi, \sigma_1 \rangle) = f(\langle \varphi, \sigma_2 \rangle)$. If $\sigma_1 \preceq_l \sigma_2$, then we throw away σ_2, i.e. we will not explore that subtree any further. If $\sigma_2 \preceq_l \sigma_1$, then we throw away σ_1. Note that as $f(\langle \varphi, \sigma_1 \rangle) = f(\langle \varphi, \sigma_2 \rangle)$, either both (φ, σ_1) and (φ, σ_1) are in L_{SAT} or both are out. After removing all duplicates, if more than N nodes survive, we throw away the left-most partial assignment at this level. We keep removing the left-most

节点直到我们只有N节点为止. 显然,毕竟我们在这一级别上说并做了这些事,我们最多可以有能生存下去的N节点。

Then we continue at the next level, expanding only these subtrees. In general, suppose $r \leq N$ nodes survive at level m. In level $m+1$, we consider their $2r$ children. If $2r \leq N$, we can move on to level $m+2$. If not, we run the reduction f on these $2r$ nodes, and obtain $2r$ output strings. We first eliminate duplicates and then, if more than N nodes still remain, we eliminate the the left-most nodes, until we have at most N nodes. Continuing this way, finally we will reach the leaf level of the tree. Here, all the nodes correspond to full truth assignments of φ. We will have at most N surviving full assignments. For each of these assignments, we check if any of them satisfy φ. If so φ is satisfiable. Otherwise, we claim it is not satisfiable.

Clearly, we maintain at most N nodes at any level of the tree and N is polynomial in n. Depth of the tree is n. Thus the algorithm runs in polynomial time. Suppose φ is satisfiable. We argue that the left-most satisfying assignment will remain a descedant of some node kept at any level, and thus survive at the leaf-level. We do this by induction on the level number m. This is clearly so at the root level. Consider level m and let the number of nodes surviving at this level be r. By induction, the left-most satisfying assignment is a descendant of one of the r surviving nodes. We consider the $2r$ children at level $m+1$. If $2r > N$. Suppose $f(\langle\varphi, \sigma_1\rangle) = f(\langle\varphi, \sigma_2\rangle)$ and $\sigma_1 \preceq_l \sigma_2$. So we remove a duplicate. Let $\sigma_1 \preceq_l \sigma_2$. In this case, we throw away σ_2. Clearly, the left-most satisfying assignment is not a descendant of σ_2, for otherwise we would have the contradiction that $\langle\varphi, \sigma_2\rangle \in \mathrm{L_{SAT}}$ and $\langle\varphi, \sigma_1\rangle \notin \mathrm{L_{SAT}}$, as σ_1 is strictly to the left of σ_2. Thus, throwing away σ_2 did not violate the inductive hypothesis.

After removing duplicates, if still some $d > N$ nodes exist, let these nodes be $\sigma_1, \sigma_2, \ldots, \sigma_d$. Then we removed the left-most node σ_1. We argue that this is also a correct procedure. By contradiction, suppose the left-most satisfying assignment σ is a descendant of σ_1. Then, σ is to the left of all the d surviving nodes. Thus, all of $\langle\varphi, \sigma_1\rangle, \langle\varphi, \sigma_2\rangle, \ldots \langle\varphi, \sigma_d\rangle$ are in $\mathrm{L_{SAT}}$. So, $f(\langle\varphi, \sigma_1\rangle), f(\langle\varphi, \sigma_2\rangle), \ldots f(\langle\varphi, \sigma_d\rangle)$ are all in S. Note that, all these strings are distinct and there are $d > N$ of them. But, by our assumption of sparseness of S, there can be at most N strings in S (at this length). This contradiction shows that throwing away the left-most node also maintains the inductive hypothesis. This completes the induction proof. ♣

第5章

随机化

随机性,既作为一种证明技术,也是一种计算资源,在现代算法理论和复杂度理论中具有显著作用. 在本章中,我们开始研究计算中的随机化。

5.1 基础概率

It is perhaps one of the primal experiences, much like the primal experiences with arithmetic quantity or geometric shape, that people have come to "know" randomness. Any systematic exploration was started much later, starting with Pascal, Fermat and Laplace. But it is always a nettlesome quesiton as to what exactly is "randomness", and what exactly is "probability". I don't think this question has ever been truly satisfactorily answered, despite a lot of work on this topic. Perhaps there is no one single answer. However, brushing aside the "nature" of what is "probability", modern mathematics basically took the following perspective, after Kolmogorov. We start with an arbitrary measure space $(\Omega, \mathbf{E}, \mu)$, where Ω is some underlying set, equipped with a measure function $\mu : \mathbf{E} \to \mathbf{R}_+$, where $\mathbf{E}$ is a collection of subsets of Ω called a σ-algebra satisfying certain closure properties, and (being a probability measure) $\mu(\Omega) = 1$. (Being a measure, μ must satisfy σ-additivity, namely for disjoint countable family $\{S_i\} \subset \mathbf{E}$, $\mu(\bigcup_{i=1}^{\infty} S_i) = \sum_{i=1}^{\infty} \mu(S_i)$. $\mathbf{E}$ is a σ-algebra if $\Omega \in \mathbf{E}$, and $\mathbf{E}$ is closed under countable union, intersection, and complement. We will not discuss further the properties of a σ-algebra. For most of what we do, the set Ω will be finite, and $\mathbf{E}$ consists of all subsets of Ω.)

The Kolmogorov foundation of probability theory is a brilliant device to refocus the "theory of probability" as an internal mathematical subject, excluding all issues having to do with what is "randomness" as we experience it in the external world. It essentially gets rid of the issue of any reference to how this "probability theory" is to be related to the every day (or not so every day) notion of "chance" in the external world. A gambler (or any user of probability theory) has to decide what is an appropriate probability space to presume, as

a mathematical model for his particular situation at hand. Ths modeling is external to any mathematical theory of probability. The mathematics of "probability theory" in the sense of Kolmogorov only helps to "calculate" the outcomes once the basic probability of events have been assigned. Thus, in Kolmogorov's probability theory, there is no meaning of a "fair coin"; instead, one merely sets up a suitable probability space, such as $\{H, T\}$, and postulate that $\mu(H) = \mu(T) = 1/2$. It is murkier what a quantum physicist really means when he talks about the "probability" of observing this or that quantum state.

What if we want "two independent fair coin flips"? In Kolmogorov's probability theory, we can set up a product probability space, $\{H, T\}^2$, and assign $\mu(HH) = \mu(HT) = \mu(TH) = \mu(TT) = 1/4$. Note that in this 4-point space, there is, strictly speaking, no notion of time.

虽然科尔莫戈罗夫基金会是好的,但似乎还是有些没有结果,缺乏"概率思维"的某些直觉。 首先,我们确实喜欢有一个原始的概念 一个独立的硬币翻转。 此外,如果在几次独立的公平硬币翻转之后,我们决定再翻一次。 这不应扰乱先前的任何概率计算。 在科尔莫戈罗夫的架构中,我们必须重新构建一个新的概率空间。 从技术上讲,现在必须在这个新的概率空间里进行以前的所有概率计算。 当然,它们具有相同的价值,但从概念上讲,它们现在发生在不同的概率空间。 这,我发现不自然和不满意。 通常我们想在算法中 能够翻转更多的硬币 而这种时间步的概念是自然的,直观的帮助. 假设我们一定有一个超概率空间 预先设定,这是有点不正常的, 其定义本身就包含了一个内在的结构, 说明我们能有多少硬币翻转。 但似乎都是自作自受,

幸运的是,就我们在这里讨论的几乎所有问题而言,它将超越一些有限的(或最多无限和递归性的)空间。 大部分时间,它只是 过 {0, 1}n, 或类似的东西。 对概率的概念采取天真的方法不会产生哲学上的困难,但是这个概念确实承认了一个新的独立点的原始概念. 原则上,我们可以列举所有基本事件,并指定它们为"原子概率",从而增加一个因素。 因此,我们将采取以下观点。 我们只要需要,就会假设我们能够有一个独立的附加硬币翻转(这也许不是公平的硬币). 我们将以更直观的"概率"水平行事,仿佛我们具有一种明确的"任意性"的含义。 因此,我们可以说,例如,在一定的概率范围内,独立地进行以下随机试验。 我们采取这一办法时有一项理解,即每当出现任何潜在的困难时,我们立即退回到科尔莫戈罗夫的测量空间基础的安全茧,并有效地说:唯一有意义的事情就是在以下测量空间中发生的事情;对"外部世界"的任何影响(例如它所谓的随机步骤究竟意味着什么)并不是我们概率分析的责任和任何结论。

5.1.1 马可夫的不平等

第一次考虑的不平等是马尔科夫的不平等。 它处理任何只使用非负值的随机变量,而估计值是根据其期望值. 从技术上讲,我们需要假定它有有限的期望,但是,人们可以放弃它,因为当非负随机变量有无限期望时,估计是微不足道的。

THEOREM 5.1 *Let X be a random variable such that $X \geq 0$ and $0 \leq \mathrm{E}[X] < \infty$. For all $a > 0$,*

$$\Pr[X \geq a] \quad \leq \quad \frac{\mathrm{E}[X]}{a}$$

期望E[X]的定义是: $\int X d\mu$. This theorem gives us a bound on the probability that X takes on a value that is *greater* than its expected value by a given amount. We give a quick proof of the theorem:

Proof. Let I be an indicator variable for the event $X \geq a$. That is,

$$I = \begin{cases} 1 & \text{if } X \geq a \\ 0 & \text{if } X < a \end{cases}$$

Clearly, $\mathrm{E}[I] = \Pr[X \geq a]$. Then,

$$\Pr[X \geq a] = \int I d\mu = \int_{X \geq a} 1 d\mu \leq \int_{X \geq a} \frac{X}{a} d\mu \leq \int \frac{X}{a} d\mu = \frac{\mathrm{E}[X]}{a}.$$

♣

5.1.2 Chebyshev 不平等

定理 5.2 对于任何带有有限 E [X] 和 Var(X) 的随机变量 X, 并让 a > 0,

$$\Pr[|X - \mathrm{E}[X]| \geq a] \leq \frac{\mathrm{Var}(X)}{a^2}$$

Here the variance $\mathrm{Var}(X)$ is defined as $\mathrm{E}[(X - \mathrm{E}[X])^2] = \mathrm{E}[X^2] - \mathrm{E}[X]^2$, and technically we must assume it is finite. However, the above estimate is (trivially) true even if it is infinite.
Proof. Define a random variable $Y = (X - \mathrm{E}[X])^2$. Then,

$$\begin{aligned} \Pr[|X - \mathrm{E}[X]| \geq a] &= \Pr[Y \geq a^2] \\ &\leq \frac{\mathrm{E}[Y]}{a^2} \\ &= \frac{\mathrm{Var}(X)}{a^2} \end{aligned}$$

以上推论指出,这种不平等是利用马尔科夫的不平等获得的。 ♣

5.1.3 切尔诺夫边界

马可夫的不平等似乎相当脆弱。 然而,它非常适用于许多情况,主要是因为它只是假设所考虑的随机变量是非负性的,并且不假定其分布. 明智地应用它,可以产生显著的界限。 在这个节中,我们用它来证明Chernoffbound, 这是一个强大的不平等 处理大量独立随机变量。

在给出Chernoffbound的声明之前,我们介绍它将使用的变量. 让 X% 1, X2, ~ ~ , Xn} 成为一组独立, 分布相同的随机变量, 每个 Xi 都有。

$$\Pr[X_i = 1] = \Pr[X_i = -1] = \frac{1}{2}.$$

Let S_n be their sum: $S_n = \sum_{i=1}^{n} X_i$.

定理5.3 对所有n, 和所有 > 0,

$$\Pr[S_n \geq a] \quad \leq \quad e^{-a^2/2n}$$

在证明这一定理之前,我们注意到,可以将其重述为:

$$\Pr[S_n \geq \epsilon \cdot n] \quad \leq \quad e^{-\frac{1}{2}\epsilon^2 n} = (e^{-\frac{1}{2}\epsilon^2})^n$$

or

$$\Pr[S_n \geq \alpha \cdot \sqrt{n}] \quad \leq \quad e^{-\frac{1}{2}\alpha^2}$$

If we think of ϵ and α as positive constants, then the first inequality says the probability of having an $\Theta(n)$ deviation from expectation (0) is exponentially small, and the second inequality says that this "tail probability" balances out at $\Theta(\sqrt{n})$ away from expectation. This last statement is in accord with the central limit theorem, which states that

$$\lim_{n\to\infty}[S_n \geq \alpha \cdot \sqrt{n}] = \int_{\alpha}^{\infty} \frac{2\pi e - x2}{\sqrt{2}\ ddx.}$$

切诺夫号的优点是,它对所有n和a有效,而不只是一种限制声明。

现在,我们证明这个定理。

证得. 诀窍在于考虑Sn的推论,即随机变量eλSn. 这里的QQ > 0是稍后需要修补的某个数字. 我们计算eλSn的期望如下.

$$\mathrm{E}\left[^{e\lambda Sn\blacksquare} = \mathrm{E}\left[^{e\lambda pnn\lambda}{}_{=1}\, X_i\right] = \mathrm{E}\left[\prod_{i=1}^{n} \quad\right] = \prod_{i=1}^{n} \mathrm{E}\left[^{e\ e}\right.\right.$$

为了获得最后的平等,我们使用这样的假设,即QQi}是独立的随机变量,因此{eλXi}也是独立的. (回顾,对于独立随机

variables X and Y, the expectation is multiplicative $\mathrm{E}[XY] = \mathrm{E}[X]\,\mathrm{E}[Y]$.) For any i, the random variable $e^{\lambda X_i}$ takes values e^{λ} and $e^{-\lambda}$, each with probability $1/2$. Its expectation is

$$\mathrm{E}\left[\text{e e} \quad = \cosh(\lambda) = \frac{e^{\lambda} + e^{-\lambda}}{2}\right.$$

上述数量以下列条件为界:

$$\frac{e^{\lambda} + e^{-\lambda}}{2} \leq e^{\lambda^2/2}.$$

可以通过分析泰勒扩大LHS和RHS来证明上述不平等. 我们最后会做分析 使用这个束缚,我们

$$\mathrm{E}\left[\text{eλSn}\right. \leq \prod_{i=1}^{n} e^{\lambda^2/2} = e^{\lambda^2 n/2}$$

我们现在可以运用马可夫的不平等来获得:

$$\Pr[S_n \geq a] = \Pr[e^{\lambda S_n} \geq e^{\lambda a}] \leq \frac{\mathrm{E}\left[\text{eλSn}\right.}{(} \leq \frac{e^{\lambda^2 n/2}}{e^{\lambda a}}$$

The above inequality is true for any λ, so we are now free to choose λ to optimize it. To get the tightest upper bound, we minimize the the exponent in the above function. Standard calculus technique shows that a minimum occurs at $\lambda = a/n$. This gives us

$$\Pr[S_n \geq a] \quad \leq \quad e^{-a^2/2n}$$

最后,我们显示

$$\frac{e^{\lambda} + e^{-\lambda}}{2} \leq e^{\lambda^2/2}.$$

泰勒扩展LHS是

$$\cosh \lambda = 1 + \frac{\lambda^2}{2!} + \frac{\lambda^4}{4!} + \cdots$$

Taylor 扩展函数前显示

$$e^{\lambda^2/2} = 1 + \frac{\lambda^2}{2} + \frac{(\lambda^2/2)^2}{2!} + \frac{(\lambda^2/2)^3}{3!} + \cdots$$

按术语比较两个系列术语. LHS的kth任期为 2k/(2k)!,而eλ2/2的任期为 λ2k/(2k)!
is $(\lambda^2/2)^k/k!$. Focusing only on the even factors, we have $(2k!) \geq 2^k \cdot k!$. Thus,

$$\cosh \lambda \leq e^{\lambda^2/2}.$$

有几种不同形式的ChernoffBound 将会有用。 它们都涉及独立随机变量的尾端概率。 让 & x1, X2, ~ ~, Xn} 成为一组 n 独立的 0-1 随机变量, 带有

$$\Pr[X_i = 1] = p_i, \quad \Pr[X_i = 0] = 1 - p_i,$$

and let $p = \sum_{i=1}^{n} p_i/n$. We define the centralized random variables $\{Y_1, Y_2, \cdots, Y_n\}$ where $Y_i = X_i - p_i$, then

$$\Pr[Y_i = 1 - p_i] = p_i, \quad \Pr[Y_i = -p_i] = 1 - p_i,$$

and $\mathrm{E}[Y_i] = 0$. Let $S_n = \sum_{i=1}^{n} Y_i = \sum_{i=1}^{n} X_i - pn$.

THEOREM 5.4 *For any $\Delta > 0$,*

定理5.4 任何 Q 0.

$$\Pr[S_n \geq \Delta] \leq e^{-2\Delta^2/n}.$$

By symmetry, the bound applies to $-S_n$ as well, and so

通过对称,约束也适用于-Sn,因此

$$\Pr[|S_n| \geq \Delta] \leq 2e^{-2\Delta^2/n}.$$

Note that when we take $\Delta = \delta pn$, then

$$\Pr[|\sum_{i=1}^{n} X_i - pn| > \delta pn] < 2e^{-2\delta^2 p^2 n}.$$

习近平 - pn| > δpn] < 2e−2δp2n.

Here is another form. Let $\{X_1, X_2, \cdots, X_n\}$ be a set of n independent 0-1 random variables, with $\Pr[X_i = 1] = p$. Then

THEOREM 5.5 *For any $0 < \delta < 1/2$,*

定理5.5 任何 0 < < 1/2,

$$\Pr[\sum_{i=1}^{n} X_i > (1+\delta)pn] < e^{-\delta^2 pn/4}$$

and

$$\Pr[\sum_{i=1}^{n} X_i < (1-\delta)pn] < e^{-\delta^2 pn/2}.$$

Theorem 5.5 is better than Theorem 5.4 for small p.

定理5.5优于小p的定理5.4.

For not necessarily 0-1 random variables,

对于不一定0-1随机变量,

THEOREM 5.6 *Let X_i, $1 \leq i \leq n$, be mutually independent with all $\mathrm{E}[X_i] = 0$ and all $|X_i| \leq 1$. Let $S_n = \sum_{i=1}^{n} X_i$. Then for all $a > 0$,*

$$\Pr[S_n > a] < e^{-a^2/2n}.$$

pr[Sn > a] < e-a2/2.n.

The proofs of these versions of Chernoff Bound all follow similar lines.

这些版本的ChernoffBound的证明都遵循了类似的线条.

ution. Randomly

这种捆绑的版本也带有超几何分布. 随机摘n球而不更换,从N黑白球,用pN黑球. 让S成为被选中的n球中的黑球数.

磕□

THEOREM 5.7 *For any* $0 \leq t \leq 1$,

$$\Pr[|S/n - p| \geq t] \leq 2e^{-2nt^2}.$$

这被称为Hoeffding束缚。

5.1.4 普遍欢迎

DEFINITION 5.8 (UNIVERSAL FAMILY OF HASH FUNCTIONS) *Let U and T be finite sets. Let S be an index set for a family of functions $\{h_s : U \to T\}_{s \in S}$. $\{h_s\}_{s \in S}$ is called a* universal family of hash functions *if* $\forall \alpha, \beta \in T, \forall x, y \in U, x \neq y$,

$$\Pr_{s \in S}[h_s(x) = \alpha \wedge h_s(y) = \beta] = \frac{1}{|T|^2}$$

注意,1/QQTQQ2以上方程的RHS是当我们从T中独立和统一地随机选择两个元素时得到α和β的概率.

The proper treatment in the Kolmogorov framework will be to define a measure space with the underlying set S, equipped with the uniform distribution. Then for all fixed $x \in U$, the map $Z_x : s \mapsto h_s(x)$ is a random variable.

We have $\forall x \neq y \in U$, $\forall \alpha, \beta \in T$, $\Pr_{s \in S}[Z_x(s) = \alpha \wedge Z_y(s) = \beta] = \frac{1}{|T|^2}$. Hence, $\forall \alpha \in T, \forall x \in U$, pick $y \in U$, and $y \neq x$, (we assume $|U| \geq 2$),

$$\begin{aligned}
\Pr_{s \in S}[Z_x(s) = \alpha] &= \sum_{\beta \in T} \Pr_{s \in S}[Z_x(s) = \alpha \wedge Z_y(s) = \beta] \\
&= \sum_{\beta \in T} \frac{1}{|T|^2} \\
&= \frac{1}{|T|}
\end{aligned}$$

So, Z_x is a uniform random variable on T. It follows that for any $x \neq y \in U$ and $\alpha, \beta \in T$,

$$\Pr_{s \in S}[Z_x(s) = \alpha \ \wedge \ Z_y(s) = \beta] = \Pr_{s \in S}[Z_x(s) = \alpha] \cdot \Pr_{s \in S}[Z_y(s) = \beta].$$

So, for any $x \neq y \in U$, the random variables Z_x and Z_y are independent. The definition of universal hash function is equivalent to asking a set of *pairwise independent* and unifromly distributed random variables, $\{Z_x\}_{x \in U}$. The random variables in this set may be jointly dependent, but any two of them are independent.

However, instead of thinking in terms of pair-wise independent random variables $Z_x(s)$, we rather think of $h_s(x) = Z_x(s)$ as a random map from U to T, by randomly choosing an index $s \in S$. The two views are of course completely equivalent here.

An Example: Let p be a prime number. Then $\mathbb{Z}/p = \{0, 1, \ldots, p-1\}$ with the operations $+$ and $\cdot$ is a finite field. Consider the map $h_{s=(a,b)} : x \mapsto ax + b$ for $a, b \in \mathbb{Z}/p$. We will verify that $\{h_{(a,b)}\}_{a,b\in\mathbb{Z}/p}$ is a universal family of hash functions.

For all $x, y, \alpha, \beta \in \mathbb{Z}/p, x \neq y$, how many pairs $(a, b) \in \mathbb{Z}/p$ are there satisfying the following equations?

$$\begin{aligned} ax + b &= \alpha \\ ay + b &= \beta \end{aligned}$$

(在上述方程中,a和b是未知数.)这个2×2线性系统的决定因素是:

$$\begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix} = x - y \neq 0.$$

因此,存在着一种独特的解决办法,即这一公式。 因此,

$$\Pr_{s=(a,b)\in(\mathbb{Z}/p)^2}[h_s(x) = \alpha \wedge h_s(y) = \beta] = \frac{1}{p^2}.$$

So, $\{h_{(a,b)}\}_{a,b\in\mathbb{Z}/p}$ is a universal family of hash functions.

This can be generalized to any finite field $\mathrm{GF}(p^n)$. It is known that for any prime p and any $n \geq 1$, there is a finite field of p^n elements. Up to isomorphism such a field is unique. In particular, for any $k \geq 0$, the polynomial $X^{2\cdot 3^k} + X^{3^k} + 1$ is an irreducible polynomial in $\mathbb{Z}_2[X]$, and therefore we have an explicit finite field in the form of

$$\mathbb{Z}_2[X]/(X^{2\cdot 3^k} + X^{3^k} + 1).$$

In the definition of this particular family of universal hash functions via affine linear functions $ax + b$, if it is defined over a finite field $\mathrm{GF}(2^n)$, we can truncate any number of bits from n to make $|T| = 2^k$, for any $0 \leq k \leq n$. Whenever in the following we speak of a family of universal hash functions, unless otherwise stated, we always refer to this family of affine linear functions, and if necessary, over that particular family of finite fields $\mathrm{GF}(2^n)$, with $n = 2 \cdot 3^k$.

普世散列函数的功率来源于这样的事实:一方面,它们的行为或多或少像随机函数,另一方面,它们只能由2n比特(英语:2n bits)巧妙地指定.

5.2 随机算法:MAXCUT

As a taste for randomized algorithms, we discuss the graph problem MAXCUT. Let $G = (V, E)$ be an undirected graph over n vertices. A cut C of G is a partition of vertices V into disjoint union $V_1 \cup V_2$. We also identify a cut with the set of edges between V_1 and V_2, i.e., $e \in C$ consists of those edges with one of its incident vertices in V_1 and the other in V_2.

DEFINITION 5.9 (MAXCUT) *A* MAXCUT *of a graph* $G = (V, E)$ *is a cut* C *such that* $|C|$ *is maximized over all cuts of* G.

与MAXCUT类似,G的MINCUT被定义为所有G削减的最小QQCQ. 我们知道MINCUT在P。 它可以通过所有对顶点之间的最大网络流量来解决.

我们知道MAXCUT是NP完成的,因此我们并不期望以科学的方式解决它。 然而,我们可以寻找大致的解决办法。 为了量化我们的近似值的准确性,我们将引入一个新的术语。 我们要一个多时算法 实现切C这样

$$\frac{|C|}{|C^*|} \geq r$$

C 是一个最大切数。 这种算法称为r近似法.

5.2.1 确定性 MAXCUT 近似算法

We can define a greedy algorithm that achieves a 1/2-approximation:
For a graph $G = (V, E)$ with $V = \{1, \ldots, n\}$, define $E_i = \{(k, i) \in E : k < i\}$. Initially, let $V_1 = \{1\}$ and $V_2 = \emptyset$. Then, for each i from 2 to n, add i to either V_1 or V_2 so that the number of edges in E_i that are on the cut is maximized, i.e., put i in V_1 iff $|\{k \in V_2 : (k, i) \in E_i\}| \geq |\{k \in V_1 : (k, i) \in E_i\}|$. We claim that this heuristic achieves 1/2-approximation.

Let C be the cut obtained by the algorithm. The disjoint sets $E_1, E_2, \ldots, E_n$ partition

(一) 国家 Ei和E两套不相连

each $i \in V$, we have $E'_i \geq E_i/2$. We conclude that $|C| \geq |E|/2$. As the size of maximum cut $|C^*| \leq |E|$, $|C| \geq |C^*|/2$.

5.2.2 随机MAXCUT 近似算法

我们为MAXCUT提出了随机的1/2. 然后,我们显示,它可以在多诺米时间被去殖民化。 这个例子说明了在简单环境下随机化和去随机化的想法.

The randomized algorithm is very simple. Assign a monkey at each vertex and have each monkey to throw a dart. If it throws to the left, assign the vertex to V_1, and if it throws to the right, assign it to V_2. More formally, given a graph $G = (V, E)$, we assign each vertex independently with equal probability to either V_1 or V_2. This will give us a cut C of G, and we will show that the expected size of $C \geq |C^*|/2$.

Consider an edge $(i, j) \in E$. $\Pr[(i, j) \in C] = 1/2$. For $e \in E$, define χ_e to be a random

variable such that $\chi_e = 1$ if $e \in C$ and $\chi_e = 0$ if $e \notin C$. Then $|C| = \sum_{e \in E} \chi_e$. 因此,

$$\mathrm{E}[|C|] = \sum_{e \in E} \mathrm{E}[\chi_e] = \sum_{e \in E} \Pr[e \in C] = \frac{|E|}{2} \geq \frac{1}{2}|C^*|.$$

The first equality follows from linearity of expectation, $E[X + Y] = E[X] + E[Y]$, for any two random variables X and Y. This formula holds even if X and Y are not independent.

5.2.3 利用通用散列函数对 MAXCUT 进行定理

Let $G = (V, E)$ be a graph with $V = \{0, \ldots, n-1\}$. Set k so that $2^k \geq n > 2^{k-1}$. Choose a and b at random from $\mathrm{GF}(2^k)$. For each $i \in V$, treat i as a member of $\mathrm{GF}[2^k]$ compute $ai + b$. Assign i to either V_1 or V_2 according to the first bit of $ai + b$. We claim that the expected size of cut obtained is $|E|/2$.

Let $\chi_{(a,b)}(i) =$ the first bit of $ai + b$. We know that $\{ai + b\}_{a,b \in \mathrm{GF}(2^k)}$ is a universal family of hash functions. Thus, $\{\chi_{(a,b)}\}_{a,b \in \mathrm{GF}(2^k)}$ is a universal family of hash functions. Then, a cut C obtained by the above randomized algorithm is given by $C = \{(i,j) | \chi_{(a,b)}(i) \neq \chi_{(a,b)}(j)\}$.

Because $\{\chi_{(a,b)}\}_{a,b \in \mathrm{GF}(2^k)}$ is a universal family of hash functions, $\Pr[\chi_{(a,b)}(i) \neq \chi_{(a,b)}(j)] = 1/2$. Thus, using the analysis from Section 3, we have $\mathrm{E}[|C|] = |E|/2$.

我们可以在多诺米时间里去化上述算法. 不到4n2,还有4n2
different choices for (a, b). To derandomize, we can examine the cuts created by $\{\chi_{(a,b)}\}$ for all $a, b \in \mathrm{GF}(2^k)$ in polynomial time. One of these cuts is guaranteed to be at least $|C^*|/2$ because $\mathrm{E}[|C|] \geq |C^*|/2$. This gives us a deterministic r-approximation algorithm for MAXCUT.

虽然这种去随机算法并没有给出比贪婪算法更好的近似比,但它是一个平行的算法. 对于每一对(a,b),每个顶点的哪一边的断面的确定是独立于其他顶点的. 因此,可以同时执行。 此外,从每对(a、b)中产生的切除量独立于从其他对中产生的切除量。 因此,所有削减都可以同时确定。 在计算了这些削减之后,可通过国家信息通报计算得出最大值。

5.2.4 戈曼斯-威廉森算法

Goemans和Williamson给出了MAXCUT的近似算法,误差比率约为12%. 随机化与半定态优化相结合,在此发挥重要作用.

Typically a subset of $[n]$ can be described by a binary sequence in $\{0,1\}^n$. Although this is not essential, we will find it here more convenient to describe such a subset $V_1 \subseteq V$,

which corresponds to a cut $V_1 \cup (V \setminus V_1)$, by a vector $x \in \{-1, 1\}^n$, by letting $x_i = -1$ iff $i \in V_1$. Then the cut size is $\sum_{e=\{i,j\}} (x_i - x_j)^2/4$. Therefore the MAXCUT problem seeks to maximize

$$\frac{1}{4} \sum_{e=\{i,j\}} \text{(xxxj)2 (中文(简体)).}$$

subject to the constraint $x \in \{-1, 1\}^n$. This constraint can be expressed as

$$x_i^2 = 1, \forall i \in V.$$

这样的问题被称为四极编程问题,在一般意义上,它也是NP-硬的(这当然并不奇怪,因为我们刚刚将一个NP-硬的问题MAXCUT简化为它.

The next trick is to linearize the problem by introducing a set of new variables $y_{ij}, 1 \leq i, j \leq n$, with the intention that $y_{ij} = x_i x_j$. Under this new set of variables, the objective function becomes

$$\frac{1}{4} \sum_{e=\{i,j\}} \text{(yii + yjj − 2yij) (中文(简体)).}$$

将尽量扩大,但须遵守

$$y_{ii} = 1, \forall i \in V.$$

Observe that a "solution" y_{ij} need not correspond to any real solution x_i. For example, if $y_{ij} = x_i x_j$, then the matrix $Y = (y_{ij})$ is symmetric and positive semi-definite, being the product of X and its transpose X^{T},

$$Y = XX^{\mathrm{T}} = \begin{pmatrix} \\ \\ \\ \end{pmatrix} (x_1 x_2 \ldots x_n)$$

Not only that, since X is $n \times 1$, Y is of rank 1 (assuming $X \neq 0$, which is implied by $y_{ii} = 1$). However, we choose to ignore this rank condition, yet preserve the constraint that Y is symmetric and positive semi-definite. The reason for this is that there is a polynomial time algorithm, based on the ellipsoid method, which solves this "semi-definite" programming problem optimally.

我们所做的是"放松"最初的问题。 要求Y为正半定数,这似乎等于是无限的不平等

$$v^{\mathrm{T}} Y v \geq 0, \forall v \in \mathbf{R}^n.$$

这些是Yij的线性限制. 这是一种"半定数"编程算法的诡计,它可以暗中处理这些无限的不平等。 我们不再进一步对此作进一步阐述,而只是假设有这样一个多时算法。

回到MAXCUT, 在多名制时,我们发现一个对称和正半定值 Y , 与y*

these constraints. Let the maximum value be M^*. Since this is a relaxation of the original problem, M^* is an upper bound of the maximum cut size.

众所周知,对称和正半定式矩阵Y*可表示为:

a product of UU^{T}, where $U^{\mathrm{T}} = \begin{pmatrix} & \\ & \end{pmatrix}$, and where $u_i \in \mathbf{R}^n$. Moreover this decomposition

可在多诺米时间找到。

Thus, $y_{ii} = u_i^{\mathrm{T}} u_i = ||u_i||^2 = 1$ says that each vector u_i is a unit vector. Moreover, "formally" retracing the expansion of $\sum_{e=\{i,j\}}(x_i - x_j)^2$ into $\sum_{e=\{i,j\}}(y_{ii} + y_{jj} - 2y_{ij})$, we get

$$M^* = \frac{1}{4} \sum_{e=\{i,j\}} \text{- ui - u - 2}$$

The next idea of Goemans and Williamson is brilliant. *Randomly* choose a hyperplane Π in $\mathbf{R}^n$. This amounts to choosing a unit vector $v \in \mathbf{R}^n$, uniformly on the unit sphere, as the normal vector to Π. (This can be carried out approximately with exponentially small error; we are ignoring issues of discretizing the process here.) Now partition V according to which side of Π the vector u_i falls. More precisely, assign vertex i to V_1 iff the inner product $\langle u_i, v \rangle > 0$.

If we express the cut size thus formed as a sum of 0-1 random variables, which indicate whether edge $e = \{i, j\}$ belongs to the cut, then the expectation is

$$\sum_{e=\{i,j\}} \text{[Π离ui,uj].}$$

为了调查这种概率Pr[ΠΠΠ separate separate,uj],我们只需要从ui和uj横跨的二维空间来考虑. 显然这种概率是 ij/ ij , ij 是 ii 和 uj 之间的角度。 时间长度显然为两罪

简单的计算显示函数

4 sin2 θ 2 实现最小 69002507... 在2.331122370号... 因此,对削减的预期

$$\sum_{e=\{i,j\}} \frac{\theta_{ij}}{\pi} \geq \frac{.69002507}{\pi} \sum_{e=\{i,j\}} ||u_i - u_j||^2 \approx 0.878 \cdot M^*.$$

这是一个随机的MAXCUT算法,实现了87.8%的近似比. 有一些方法可以使这个算法去化, 但我们不会再讨论这个问题了.

5.3 随机算法:两个阶段

普世散列的一种简单应用是弗雷德曼,科姆洛斯,和施梅雷迪(英语:Szemeredi (FKS))所开发的两阶段散列式的词典数据结构. 我们将定义一个叫做碰撞号的数量, 不再使用一个复杂的工具作为马尔科夫不平等来证明它的属性。 碰撞次数在其他地方也普遍有用。

Given a finite set U and a subset $N \subseteq U$, we want to build an efficient dictionary. A dictionary is a data structure where we store information about the elements of N. Then, upon receiving a query $q \in U$, we want to look-up the dictionary and tell whether $q \in N$ or not efficiently. For example, let U be the set of all "words" of length five over the English alphabet and let N be the set of all proper English words of length five. This is where the name dictionary comes about. Various performance criteria of interest are time required to build the dictionary, space occupied by the dictionary and look-up time.

There are many deterministic schemes to implement a dictionary. These are usualy based on some balanced tree structure, and typically take $O(\log n)$ steps for each operation. Yao was the first to question this fundamental methodology ("Should tables be sorted") and proposed hashing as an efficient alternative. The FKS two-stage hashing scheme is extremely simple. It shows that given a universe U and a subset $N \subseteq U$, where $n = |N|$, one can build a dictionary for N that has the following properties:

- 字典可按预期时间O(n)编制。
- 字典占用O(n)空间。
- 词典检索需要O(1)时间。

Let us choose a set T to be our hash table. Size of the table $|T|$ will be fixed later. We choose a suitable hash function $h : U \mapsto T$ and for each element $u \in U$, hash it to the bucket numbered $h(u)$ in T. Then, given a query $q \in U$, we can check $q \in N$ or not by scanning the bucket given by $h(q)$. We say that elements $a, b \in N$ collide under h, if $h(a) = h(b)$. It is clear that more the number of collisions, it will take more time to look-up. Thus, the goal is to choose a good hash function h that minimizes the number of collisions, and yet does not use too much space.

Let H be a universal family of hash functions from U to T. Recall that such a family satisfies the following property. For any $u_1, u_2 \in U$ with $u_1 \neq u_2$,

$$\Pr_{h \in H}[h(u_1) = h(u_2)] = \frac{1}{|T|}.$$

Choose a hash function h at random from H. Let C be the random variable that gives the number of collisions under h. For $a, b \in N$, $a \neq b$, let $X_{a,b}$ be the indictor variable that denotes whether a and b collide under h (i.e. $X_{a,b} = 1$, if $h(a) = h(b)$ and 0, otherwise). Then,

$$C = \sum_{a \neq b} \quad \text{页:1}$$

Since the probability that a pair (a, b), where $a \neq b$, collides under a random hash function h is $\frac{1}{|T|}$, the expected number of collisions is

$$E(C) = \binom{n}{2} / |T|$$

Set $|T| = n^2$. Then, $E(C) < \frac{1}{2}$. By Markov's inequality, $\Pr[C \geq 1] \leq \frac{1}{2}$. So, $\Pr[C < 1] \geq \frac{1}{2}$. Since C is a integer variable, $C < 1$ implies that $C = 0$. Hence, if we choose h at random, with probability at least $1/2$, there will be no collisions under h. We can try various h at random until we find a h with $C = 0$. The expected number of trials needed is just 2. This scheme is wonderful, except that, we have to allocate $O(n^2)$ space to get this bound with a single random hash function. We can do better by using the following two-stage hashing scheme.

In the first stage of the scheme, we allocate only $|T| = n$. Then, for a random $h \in H$, $E(C) < \frac{n}{2}$. Using Markov's inequality again, $\Pr[C \geq n] \leq \frac{1}{2}$. By trying various hash functions at random, we can find a hash function with $C \leq n$. The expected number of trial will be only 2. Suppose we have found a hash function h_0 with the number of collisions $C_0 \leq n$. Next we proceed to the second stage of the scheme.

Let N_i be the set of elements of N that got mapped to i^{th} bucket of T under h_0. Let $n_i = |N_i|$. In the second stage, for each bucket $1 \leq i \leq |T|$, we build a hash table T_i for the
(原始内容存档于2018-09-31). I. 修补一个散列功能的通用家族 Hi:U 7–Ti. 然后,对于随机从Hi中选择的散列函数h,预期碰撞次数是:

$$E(C) = \binom{n_i}{2} \frac{1}{|T_i|} = \binom{n_i}{2} \frac{1}{n_i^2} < 1/2.$$

Thus, $\Pr[C > 1] \leq 1/2$, which implies $\Pr[C = 0] \geq 1/2$. For each i, by trying various h at random, with an expected number of two trials, we can find a h_i with no collisions. Thus in total we expect to try no more than $2n$ times.

散列表的总大小(两个阶段合并)是多少? 这是由

$$\text{Total size} = |T| + \sum_{i=1}^{|T|} |T_i| = n + \sum_{i=1}^{n} n_i^2.$$

We now need a bound for the summation. We obtain a bound by using the fact that the number of collisions in first stage $C_0 \leq n$. We express C_0 in a different way to get the bound.

$$
\begin{aligned}
C_0 &= \sum_{i=1}^{n} \binom{n_i}{2} \\
&= \frac{1}{2}\left(\sum_{i=1}^{n} n_i^2 - \sum_{i=1}^{n} n_i\right)
\end{aligned}
$$

请注意,Pn $_{=1}\, n_i = |N| = n$. Thus,

$$
\sum_{i=1}^{|T|} n_i^2 = 2C_0 + \sum_{i=1}^{n} n_i \leq 2n + n = O(n).
$$

So, the total space needed to store the main hash table T and the second level tables is only $O(n)$ (We will need space to store the hash functions h_0 and various chosen h_i. But this needs only $O(\log |U|)$ space, which we assume to be negligible). We next compute time required to build the tables. In the first stage, expected number of trials to find a "good" hash function h_0 is 2. Given a hash function h we can tell whether it is good or bad in time $O(n)$. Thus, expected time for first stage is $O(n)$. In the second stage, for each bucket i, expected number of trials to find a good h_i is 2 and the time needed to tell whether a h is good or bad is $O(n_i)$. Thus, total expected time in second stage is $O(n_1) + O(n_2) + \cdots + O(n_{|T|}) = O(n)$.

Now let us turn to look-up operation. Given a query q, apply h_0 to q to get the relevant bucket $i = h_0(q)$, then apply the second level hash function h_i to q and scan the bucket $h_i(q)$ in table T_i. As each h_i is chosen to be collision-free, there will only one entry in the bucket $h_i(q)$. If this entry is q, then report that $q \in N$. If it is not, report that $q \notin N$. This process takes $O(1)$ time since we simply apply hash functions (We can construct universal family of hash functions, where computing any $h(x)$ takes time $O(\log |U|)$, which is assumed to be negligible).

5.4 随机复杂分类

5.4.1 定义

定义5.10(BPP) BPP代表有界限的误差概率多元时段. 如果有一个布尔的上游D(...,),则在BPP中使用L语言,在确定多诺时可以计算,以便:

$$
\begin{aligned}
x \in L &\implies \Pr_y[D(x,y) = 1] \geq 3/4 \\
x \notin L &\implies \Pr_y[D(x,y) = 1] \leq 1/4,
\end{aligned}
$$

在任意位数使用的地方,QQYQQ在输入的QXXXXQ的长度上是多边的.

定义5.11(RP)RP是这类分类的最早版本,只是表示随机化的多名时间。 如果有一个布尔的上游D(...),则在RP中有一个L语言,在确定多诺时可以计算,以便:

$$\begin{aligned} x \in L &\implies \Pr_y[D(x,y)=1] \geq 1/2 \\ x \notin L &\implies \Pr_y[D(x,y)=1] = 0, \end{aligned}$$

在任意位数使用的地方,QQYQQ在输入的QXXXXQ的长度上是多边的.

DEFINITION 5.12 (ZPP) *A language L is said to be in* ZPP *if there is a polynomial time computable function $D : \{0,1\}^* \times \{0,1\}^* \mapsto \{0,1,?\}$ such that if $x \in L$, for any y, its output $D(x,y) \in \{1,?\}$, and if $x \notin L$, for any y, its output $D(x,y) \in \{0,?\}$. Moreover, D should have high success probability:*

$$\begin{aligned} x \in L &\implies \Pr_y[D(x,y)=1] \geq 1/2 \\ x \notin L &\implies \Pr_y[D(x,y)=0] \geq 1/2, \end{aligned}$$

在任意位数使用的地方,QQYQQ在输入的QXXXXQ的长度上是多边的.

5.4.2 扩大溴化阻燃剂

在本节中,我们表明,BPP算法成功的概率可以"放大"到指数接近1的高度,而额外工作只有多诺米量。 我们从5.4定理中的切尔诺夫号开始

让L成为概率多元时间TM M接受的语言,其含义如下:

$$\begin{aligned} x \in L &\implies \Pr[M(x;r)=1] \geq \frac{1}{2} + \epsilon \\ x \notin L &\implies \Pr[M(x;r)=1] \leq \frac{1}{2} - \epsilon \end{aligned}$$

for random strings r of an appropriate polynomial length in $n = |x|$. Here, ϵ can be as low as $1/p(n)$ for some fixed polynomial $p(\cdot)$. We wish to amplify the success probability of the algorithm. In particular, we want to get exponentially close to 1, meaning, for any fixed polynomial $q(\cdot)$, we want a machine M' with

$$\begin{aligned} x \in L &\implies \Pr[M'(x;r')=1] \geq 1 - e^{-q(n)} \\ x \notin L &\implies \Pr[M(x;r)=1] \leq e^{-q(n)} \end{aligned}$$

我们要求M ' 在多边时间运行,因此,M ' 使用的随机弦的长度也应当是多边的。

The idea is to run M a large polynomial number of times, and take the majority vote. Given an input x, M' will simply run the machine M on input x some $2m+1$ times, with $m \geq q(n)/(4\epsilon^2)$, say, and accept x iff at least $m+1$ runs of M on x accept. In this process we will need independent and uniformly chosen random $r_1, r_2, \cdots, r_{2m+1}$, with a total length of the random string $|r'| = O(|r|q(n)/\epsilon^2)$.

Let X_i be the 0-1 random variable indicating the i-th run $M(x; r_i)$, $p = \Pr[M(x; r) = 1]$, and $S = \sum_i (X_i - p)$. Suppose $x \notin L$, then $p \leq \frac{1}{2} - \epsilon$. Apply Theorem 5.4 with $\Delta = (2m+1)\epsilon$, we get

$$\Pr[M'(x) \text{ accepts }] = \Pr[\sum_i X_i \geq m+1] \leq \Pr[S \geq (2m+1)\epsilon] \leq e^{-q(n)}.$$

Similarly, if $x \in L$, then $p \geq$

$$\Pr[M'(x) \text{ rejects }] \leq e^{-q(n)}.$$

When ϵ^{-1} is polynomially bounded, so is $m = O(q(n)/\epsilon^2)$. This achieves exponentially small error probability.

5.5 Sipser-Lautemann Theorem: BPP $\subseteq$ PH

2. 国家 在不丧失通俗性的情况下,有一个可计算出的多元计算时间(.,.),以便:

$$\begin{aligned} x \in L &\implies \Pr_{y \in \{0,1\}^m}[D(x,y) = 1] \geq 1 - 1/m \\ x \notin L &\implies \Pr_{y \in \{0,1\}^m}[D(x,y) = 1] \leq 1/m \end{aligned}$$

where the number of random bits used m is polynomially bounded in input length $n = |x|$. For any input $x \in \{0,1\}^n$, define its witness set $W_x = \{y \in \{0,1\}^m | D(x,y) = 1\}$. Lo, if $x \in L$ then the witness set W_x is "fat", whereas if $x \notin L$ then W_x is "thin". Now, in order to realize two notions of fat and thin sets as a property of sets in Σ_2^p, we first formalize the notion of fat and thin sets, and then show that the property that a set is fat is in Σ_2^p. We will show how to test whether W_x is fat or thin in Σ_2^p.

We say that $S \subseteq \{0,1\}^m$ is *fat*, if $|S|/2^m \geq 1 - 1/m$. S is said to be thin, if $|S|/2^m \leq 1/m$. In general a subset S may be neither fat nor thin. But, the witness sets we are interested are always either fat or thin. For a string $u \in \{0,1\}^m$, define $S \oplus u = \{s \oplus u | s \in S\}$. Here, $s \oplus u$ denotes the m-bit string obtained by bit-wise XOR of s and u: if $s = s_1 s_2 \ldots s_m$ and $u = u_1 u_2 \ldots u_m$, then $s \oplus u = s_1 \oplus u_1 \cdots s_m \oplus u_m$. Think of $\{0,1\}^m$ as a vector space, S as a subset of this space and u to be a vector in it. Then, $S \oplus u$ is nothing but the subset obtained by "shifting" S by u.

我们首先非正式地讨论改变脂肪和薄片的效果。 假设S是脂肪。 然后,如果我们选择一个合适的数位的转动矢量u1,u2,. . . .. ,我们随机,极有可能,这些班次的结合将“覆盖”整个空间:

2: 条件“有一组矢量,使这些矢量所转移的Wx覆盖整个空间”可以表示为 Σp

2号!

LEMMA 5.13 *Let $S \subseteq \{0,1\}^m$.*

1. If S is thin, then for any $r < m$, for any set of r vectors, the shifts cannot cover the

1. 国家 如果S是薄的,那么对于任何r < m,对于任何一组r向量,转动不能覆盖整个空间:

$$\Pr_{u_1,u_2,\ldots,u_r \in \{0,1\}^m} \left[\bigcup_{i=1} (S \oplus u_i) = \{0,1\}^m \right] = 0.$$

2. If S is fat, then with high probability, randomly chosen shifts will cover the entire space:

2. 国家 如果S是脂肪,那么在概率很高的情况下,随机选择的班次将覆盖整个空间:

$$\Pr_{u_1,u_2,\ldots,u_r \in \{0,1\}^m} \left[\bigcup_{i=1}^{r} (S \oplus u_i) = \{0,1\}^m \right] \geq 1 - \frac{2^m}{m^r}$$

证得. 第一部分是显而易见的。 用于任意向量u1,u2,. . . .. 轮班的结合具有核心意义

既然我们假设r < m和S是薄的,\\\ n\\\\\\\\\\\\\\\\
\\

ıot cover the entire space means that some vector $y \in \{0,1\}^m$ is not covered by these shifts. So,

$$\Pr_{u_1,u_2,\ldots,u_r} \left[\bigcup_{i=1}^{r} (S \oplus u_i) \neq \{0,1\}^m \right] \leq \sum_{y \in \{0,1\}^m} \Pr_{u_1,u_2,\ldots,u_r} \left[y \notin \bigcup_{i=1}^{r} (S \oplus u_i) \right]$$

Fix any $y \in$ [illegible]. By the properties of $\oplus$ function[illegible] that $r \notin S \oplus u$ iff $r \oplus u \notin S$ (we use the fact that for any u, $u \oplus u = 0$). Thus, y will not be covered by the r shifts iff $\{y \oplus u_1, y \oplus u_2, \ldots, y \oplus u_r\} \cap S = \emptyset$. For any u chosen uniformly at random from $\{0,1\}^m$, $y \oplus u$ is distributed uniformly in $\{0,1\}^m$. (We use the fact that the function $f_y : u \mapsto y \oplus u$ is 1-1.) So, for a randomly chosen u, $\Pr[y \oplus u \notin S] \leq 1/m$, because S is fat. By independence, it follows that

$$\Pr_{u_1,u_2,\ldots,u_r} [\{y \oplus u_1, y \oplus u_2, \ldots, y \oplus u_r\} \cap S = \emptyset] = \prod_{i=1}^{r} \Pr_{u_i} [y \oplus u_i \notin S] \leq \left(\frac{1}{m} \right)^r.$$

我们的结论是,

$$\begin{aligned}\left[\bigcup_{i=1}^{r}(S\oplus u_i)\neq\{0,1\}^m\right] &\leq \sum_{y\in\{0,1\}^m}\left[y\notin\bigcup_{i=1}^{r}(S\oplus u_i)\right]\\ &= 2^m\cdot\left(\frac{1}{m}\right)^r\\ &= \frac{2^m}{m^r}.\end{aligned}$$

lemma的第二部分从上面的束缚。 ♣

2. 国家 对于一个合适的r值,lemma会显示,如果S胖,那么对于一些r向量,转动会覆盖整个空间,如果S瘦,那么对于任何一组r向量,转动不会覆盖整个空间. 上述财产可在 \\ p 中测试

2. 国家 正式证明如下。

THEOREM 5.14 (SIPSER–LAUTEMANN) $\text{BPP}\subseteq\Sigma_2^p$.

Proof. Let $L\in\text{BPP}$. Without loss of generality, there is a polynomial time computable predicate $D(\cdot,\cdot)$ such that,

$$\begin{aligned}x\in L &\implies \Pr_{y\in\{0,1\}^m}[D(x,y)=1]\geq 1-1/m\\ x\notin L &\implies \Pr_{y\in\{0,1\}^m}[D(x,y)=1]\leq 1/m\end{aligned}$$

where the number of random bits used m is polynomially bounded in input length $n=|x|$. For an input $x\in\{0,1\}^n$, define its witness set $W_x=\{y|D(x,y)=1\}$. If $x\in L$ then the witness set W_x is fat, whereas, if $x\notin L$, the witness set is thin. Choose $r=m/2$. Then, from Lemma 5.13, we have

$$\begin{aligned}x\in L &\implies \Pr_{u_1,u_2,\ldots,u_r}\left[\bigcup_{i=1}^{r}(W_x\oplus u_i)=\{0,1\}^m\right]\geq 1-\frac{2^m}{\overline{m^r}}\\ x\notin L &\implies \Pr_{u_1,u_2,\ldots,u_r}\left[\bigcup_{i=1}^{r}(W_x\oplus u_i)=\{0,1\}^m\right]=0.\end{aligned}$$

For sufficiently large m, the first probability >0. So, if $x\in L$, there exists r vectors such that the shifts cover the entire space, and if $x\notin L$, for any r vectors, the shifts do not cover the entire space. Observe that, for any $u_1,u_2,\ldots,u_r$,

$$\bigcup_{i=1}^{r}(S\oplus u_i)=\{0,1\}^m \iff \forall y\in\{0,1\}^m\left[\vee_{i=1}^{r}(y\oplus u_i\in W_x)\right]$$

由此可见,

$$x \in L \iff \exists u_1, u_2, \ldots, u_r \forall y \in \{0,1\}^m \left[\vee_{i=1}^{r} (y \oplus u_i \in W_x)\right].$$

$y \oplus u_i \in W_x$ simply means that $D(x, y \oplus u_i) = 1$. So we conclude that,

$$x \in L \iff \exists u_1, u_2, \ldots, u_r \forall y \in \{0,1\}^m \left[\vee_{i=1}^{r} (D(x, y \oplus u_i) = 1)\right].$$

上游点 Q r $_{=1}(D(x, y \oplus u_i) = 1)$ is testable in polynomial time, as r is polynomial in n and D is a polynomial time predicate. We conclude that $L \in \Sigma_2^p$. $_2$ predicate. To do that, we first rephrase $_2$. We can ♣

2. 国家 我们还可以直接证明这一说法,证明:

$$\left[\bigcup_{i=1}^{r} (S \oplus u_i) = \{0,1\}^m\right] \iff \left[\bigcap_{i=1}^{r} (S^c \oplus u_i) = \emptyset\right].$$

Moreover, if S is thin then S^c is fat. So, Lemma 5.13 can be rephrased as follows. Suppose S is fat. Then S^c is thin. Applying Part 1 of the lemma to S^c, we see that for any set of

此外,如果S瘦了,那么Sc胖了. 因此,莱玛5.13可以重新表述如下. 假设S是脂肪。 那Sc就瘦了点 将lemma的第一部分应用到 Sc 中,我们看到对于任意一组r < m 转向向量, u1, u2, ... 额 转弯的交汇处是非空的 现在假设S是瘦的。 那Sc是胖子了,你觉得呢? 将 lemma 的 Part 2 应用到 Sc 中,我们看到随机选择的 r 转向矢量 u1, u2。 . . .. 额,由于概率很高, 班次的交汇点是空的。 形式上

1. *If S is fat, then for any $r < m$, for any set of r vectors, the intersections of the shifts is non-empty:*

1. 国家 如果S是脂肪,那么对于任何r < m,对于任何一组r向量,转动的交叉点是非空的:

$$\left[\bigcap_{i=1}^{r} (S^c \oplus u_i) \neq \emptyset\right]$$

2. *If S is thin, then with hig bility, randomly chosen shifts will have an empty intersection:*

2. 国家 如果S瘦,那么在概率高的情况下,随机选择的班次会有一个空的交叉点:

$$\left[\bigcap_{i=1}^{r} (S \oplus u_i) = \emptyset\right]$$

$$\overline{m^r}$$

Applying Lemma 5.15 to witness sets, with $r = m/2$, we have

$$x \in L \implies \forall u_1, u_2, \ldots, u_m \left[\bigcap_{i=1}^{r} (W_x \oplus u_i) \neq \emptyset\right]$$

$$x \notin L \implies \exists u_1, u_2, \ldots, u_m \left[\bigcap_{i=1}^{r} (W_x \oplus u_i) = \emptyset\right]$$

以上财产可以表示为 QQp 2个上游:

$$x \in L \iff \forall u_1, u_2, \ldots, u_m \exists y \left[\wedge_{i=1}^{r} (y \oplus u_i) \in W_x)\right]$$

5.6 隔离Lemma

The isolation lemma provides a mechanism to approximately compute the size of some set $S \subseteq \{0,1\}^m$, such as a BPP witness set. In this section, we state and prove the lemma. In 2,并讨论大致计算.

Let S and T be finite sets and $\mathcal{H}$ be a universal family of hash functions from S to T. Two distinct elements $x, y \in S$ are said to *collide* under a hash function $h \in \mathcal{H}$, if $h(x) = h(y)$. We say that a hash function h *isolates* an element $x \in S$, if no element in S collides with x. One can imagine that h "likes" x and gives it a separate seat in T to sit alone comfortably! A set of hash functions $\{h_1, h_2, \ldots, h_r\}$ is said to isolate an element $x \in S$, if one of these function h_i isolates x. The set of functions is said to *isolate all of* S, if for every element $x \in S$, there is some function h_i in the set that isolates x.

假设我们选择散列函数 h1, h2, ... ,hr, 统一和独立地从H. 这些函数集将S全部隔离的概率是多少? 答案取决于S的大小与T的大小和r的值. 直觉上,如果S足够小于T,h足够大,概率很高,随机选择的一组函数会将S全部隔离. 另一方面,如果S比T大,r也足够小,那么T没有多少空间给S中所有元素"单独座位". 对于适合选择的QQTQ和r,我们可以使隔离概率为零.

Lemma 5.16(孤立莱玛) 让H成为S到T的散列功能的普世家族. 让h1,h2,。 . . .. ,hr从H,r > 1. 随机统一独立选择。

1. *If* $|S| \geq r|T|$, *then*

$$\Pr_{h_1,h_2,\ldots,h_r}[\{h_1, h_2, \ldots, h_r\} \textit{ isolates all of } S] = 0.$$

2. *Suppose* $|S|^{r+1} \leq |T|^r$. *Then,*

pr h1,h2,...,hr[{h1,h2,]. . . .. , hr} 孤立所有 S] > 1 +
−|S|r+1 .

Proof. The first part of lemma is quite obvious. Any hash function $h \in \mathcal{H}$ can isolate at most $|T| - 1$ elements in S (it can assign "separate seats" for some $|T| - 1$ elements in S and then map all the other elements to the last "seat" in T). So, any set of r hash functions can together isolate at most $r(|T| - 1)$ elements. As $|S| \geq r|T|$, no set of r hash functions can isolate all of S.

We next prove the second part of the lemma. Fix any distinct elements $x, y \in S$. From the properties of universal family of hash functions, the probability that x and y collide under h is $1/|T|$. So, for any element $x \in S$,

$$\begin{aligned} \Pr_{h\in\mathcal{H}}[h \text{ does not isolate } x] &\leq \sum_{y\in S-\{x\}} \Pr_{h\in\mathcal{H}}[x \text{ and } y \text{ collide under } h] \\ &< \frac{|S|}{|T|} \end{aligned}$$

由此可见,对于S中的任何x,如果我们选择h1,h2. . . .. , 统一和独立地随机,

$$\text{pr h1,h2,... hr [无一hi隔离 x]} < \left(\frac{|S|}{|T|}\right)^r.$$

最后,随机选择的r散列函数不孤立所有S的概率可以通过对S中所有可能元素的总结来限定:

$$\begin{aligned} & \Pr_{h_1,h_2,\ldots,h_r\in\mathcal{H}}[\{h_1,h_2,\ldots,h_r\} \text{ does not isolate all of } S] \\ \leq & \sum_{x\in S} \Pr_{h_1,h_2,\ldots,h_r\in\mathcal{H}}[\text{none of } h_1,h_2,\ldots,h_r \text{ isolate } x] \\ < & |S| \times \left(\frac{|S|}{|T|}\right)^r \end{aligned}$$

我们的结论是:

$$\Pr_{h_1,h_2,\ldots,h_r\in\mathcal{H}}[\{h_1,h_2,\ldots,h_r\} \text{ isolates all of } S\ ] > 1 - \frac{|S|^{r+1}}{|T|^r}$$

♣

5.7 BPP $\subseteq \Sigma$2 - 使用隔离莱玛的又一证据

本节显示如何使用隔离lemma将 BPP 放到 \ p . We first present the proof idea informally. Let L be a language in BPP, via a randomized algorithm that uses m random bits, where m is polynomial in the input length. We assume that the algorithm has been suitably amplified. We will fix parameters r and size of the target set T suitably in Lemma 5.16. For any input x, let $W_x \subseteq \{0,1\}^m$ be the set of witness strings on which the algorithm accepts the input x. The parameters will be chosen appropriately, so that if $x \in L$, W_x will be large enough that no set of r hash functions $h_1, h_2, \ldots, h_r$ will isolate all of W_x; on the other hand, if $x \notin L$, W_x will be small enough so that, with high probability, randomly chosen r hash functions $h_1, h_2, \ldots, h_r$ will isolate all of W_x, and in particular there will exist some set of r hash functions that isolates all of W_x. Hence, $x \notin L$ iff there exist a set of r hash functions $\{h_1, h_2, \ldots, h_r\}$ that isolates all of W_x. The following condition can be expressed as a Σ2 上游,从而将 L 置于 QQp 2. 下面是正式证据。

THEOREM 5.17 BPP $\subseteq \Sigma_2^p \cap \Pi_2^p$.

Proof. Let $L \in$ BPP. We prove that $L \in \Pi_2^p$. As BPP is closed under complementation, the lemma follows. Without loss of generality, there is a deterministic polynomial time boolean predicate $D(\cdot,\cdot)$ such that for any input $x \in \{0,1\}^n$,

$$x \in L \implies \Pr_{y\in\{0,1\}^m}[D(x,y)=1] \geq \frac{1}{2}$$

$$x \notin L \implies \Pr_{y\in\{0,1\}^m}[D(x,y)=1] \leq \frac{1}{4m}$$

where the number of random bits used m is polynomial in the input length n. Wolog we can assume m is a power of 2. Fix $r = m$, and let T be a set of size $2^m/(2m)$. (As it is usually the case in such proofs, the above choices of parameters such as the extent of amplification, value of r and size of $|T|$ are not crucial. We can fix them in many ways to make the proof work! One such setting is given above). Let $\mathcal{H}$ be a universal family hash functions from $\{0,1\}^m$ to T. For any input x, let $W_x = \{y|D(x,y)=1\}$. Suppose $x \in L$. Then,

$$|W_x| \geq 2^{m-1} = r \cdot |T|.$$

So, Lemma 5.16 shows that no set of r hash functions $\{h_1, h_2, \ldots, h_r\}$ from $\mathcal{H}$ isolates all of W_x. On the other hand, suppose $x \notin L$. Then, from our choice of parameters,

$$\frac{*}{|T|} \leq \frac{1}{4m}.$$

Lemma 5.16同样暗示 如果我们选择h1,h2, ... , 统一和独立地从H,

$$\Pr_{h_1,h_2,\ldots,h_r}[\{h_1,h_2,\ldots,h_r\} \text{ isolates } W_x] \geq 1 - \frac{}{4m}.$$

特别是,由此,

$$x \in L \implies \forall h_1,h_2,\ldots,h_r[\{h_1,h_2,\ldots,h_r\} \text{ does not isolate } W_x]$$
$$x \notin L \implies \exists h_1,h_2,\ldots,h_r[\{h_1,h_2,\ldots,h_r\} \text{ isolates } W_x]$$

所以,我们有,

$$x \notin L \iff \exists h_1,h_2,\ldots,h_r[\{h_1,h_2,\ldots,h_r\} \text{ isolates } W_x].$$

特别的H1,h2, ... ,hr,上游词“{h1,h2,..... . . .. , hr} 隔离 Wx”可以表示为

$$(\forall y \in W_x)(\exists 1 \leq i \leq r)(\forall y' \in W_x - \{y\})[h_i(y) \neq h_i(y')].$$

乍一看,我们似乎需要写一个"Π" predicate to express the above condition. But, we can do better and express it as a coNP predicate, because $\exists 1 \leq i \leq r$ is a bounded

修饰并可以消灭. 这是逻辑上的一项一般原则,但为了完全具体,我们有{h1,h2,>. . . .. , hr] 隔离 Wx ,如果并且只有在

$$(\forall y \in W_x)\{(\forall y_1 \in W_x - \{y\})[h_1(y) \neq h_1(y_1)] \vee \ldots \vee (\forall y_r \in W_x - \{y\})[h_r(y) \neq h_r(y_r)]\}$$

等于

$$(\forall y \in W_x)(\forall y_1, \ldots, y_r \in W_x - \{y\})[(h_1(y) \neq h_1(y_1)) \vee \cdots \vee (h_r(y) \neq h_r(y_r)].$$

We conclude that, $x \notin L$ if and only if

$$\begin{aligned}&(\exists h_1, \ldots, h_r)(\forall y \in W_x)(\forall y_1, \ldots, y_r \in W_x - \{y\})\\ &\quad [(h_1(y) \neq h_1(y_1)) \vee (h_2(y) \neq h_2(y_2)) \vee \cdots \vee (h_r(y) \neq h_r(y_r)].\end{aligned}$$

We have shown that $L \in \Pi_2^p$. ♣

5.8 使用隔离法进行大致计数

使用 Sipser 的隔离 Lemma, Stockmeyer 演示了如何在 PΣp 中进行大致计算。

2. 国家事实上,他的技术已经大致计算到菲律宾国民党的水平。 让我们更加确切地说。

Definition 5.18 (#P) *#P is a function class. A function $f : \{0,1\}^* \to \mathbf{N}$ is in #P, iff there is a p-time NTM M, such that $f(x) = \#$ of accepting paths of $M(x)$.*

因此,NP langugae通常有一个“计数”版本;例如,#SAT(XXX)是公式QQ的满意任务数;#HAM(G)是G中的汉密尔顿电路数。 人们可以很容易地为这些功能形成一个多名制时间缩短的概念。 这并不奇怪。 这是因为库克的减量是令人感到羞耻的,即它们保留了解决方案的数量。 (例如,在库克从通用NP langugae 减少到SAT 中,每个接受计算都与令人满意的任务相关联。)

Less obvious, yet also #P-complete, is the permanent function: For any n by n matrix $A = (a_{ij})$,

$$\mathrm{per}(A) = \sum_{\sigma \in S_n} a_{1,\sigma 1} a_{2,\sigma 2} \ldots a_{n,\sigma n},$$

where the sum is over all permutations $\sigma : i \mapsto \sigma i$. In other words, the permanent function is defined much as the determinant function, except there are no more minus signs. For a 0-1 matrix, $\mathrm{per}(A)$ counts the number of perfect matchings of the bipartite graph with matrix A.

Variant显示永久函数也是#P-完成的(尽管图匹配的决定问题在P.

定义5.19(PP) PP代表概率多元时间。 L在PP中,如果有一个布尔的上游D(...,),在确定多诺时可以计算,以便:

$$\begin{aligned} x \in L &\implies \Pr_{y \in \{0,1\}^m}[D(x,y) = 1] \geq 1/2 \\ x \notin L &\implies \Pr_{y \in \{0,1\}^m}[D(x,y) = 1] < 1/2, \end{aligned}$$

其中任意位数,m,是输入的QXXXXXX的长度的多名数.

Note that, unlike BPP and RP, every polynomial time NTM defines a PP language, whereas not every polynomial time NTM defines a language in BPP or RP. BPP, RP and ZPP are "promise classes" in the sense that a polynomial time predicate (or equivalently a polynomial time NTM) defines a language in BPP, RP or ZPP only if it satisfies some global conditions. These conditions are, moreover, not decidable in general. Therefore we do not have an enumeration of these classes simply by an enumeration of their acceptors. This implies that we do not have a universal language, nor a complete language by this process. It is an open problem whether such complete languages exist. (Of course if BPP = P, then they indeed exist.) For PP, one can easily enumerate the class, and complete languages exist. For example, the set of Boolean formulae on n variables having at least 2^{n-1} satisfying assignments is such a language.

也很容易看出

定理 5.20

$$\mathrm{P}^{\mathrm{PP}} = \mathrm{P}^{\#P}.$$

Next we discuss approximate counting. Given a formula φ over n variables, the goal is to approximately count the number of satisfying truth assignments of φ. Let $S = \{\sigma | \varphi(\sigma) = 1\}$ be the set of all such assignments. We want to compute the first $c \cdot \log n$ bits of $|S|$, where

2oracle,如斯托克迈尔所示. 事实上,他的技术只要一个NPoracle就可以完成同样的任务,如果我们准备好为随机化的多诺米时间算法下定决心.

我们的主要工具是隔离lemma, Lemma 5.16. 我们首先用更合适的格式 重写lemma:

LEMMA 5.21 *Let $S \subseteq \{0,1\}^n$, and let $\mathcal{H}$ be a family of 2-universal hash functions from $\{0,1\}^n$ to $\{0,1\}^k$. For all $m \geq k$, choose $h_1, h_2, \ldots, h_m$ independently at random from $\mathcal{H}$.*

1. *if $|S| \leq 2^{k-1}$ then*

$$\Pr_{h_1,\ldots,h_m}[\forall x \in S \text{ some } h_i \text{ isolates } x\,] \geq 1 - \frac{1}{2^{m-k+1}}$$

2. if $|S| > m2^k$ *then*

$$\Pr_{h_1,\ldots,h_m}[\forall x \in S \text{ some } h_i \text{ isolates } x\,] = 0.$$

The idea is to try all values of k from 1 to n, and attempt to find a k such that $|S| \approx 2^k$. (We may assume our $S \neq \emptyset$; at any rate with one query to SAT we can verify this.) For any $\emptyset \neq S \subseteq \{0,1\}^n$, there is some k_S, where $1 \leq k_S \leq n$, such that $2^{k_S-1} \leq |S| \leq 2^{k_S}$. If we take every k in the range $1 \leq k \leq n+1$, and randomly pick $m = 2n$ hash functions $h_1, \ldots, h_m : \{0,1\}^n \to \{0,1\}^k$, then for each $k \geq k_S + 1$, we would get *isolation* with probability $\geq 1 - \frac{1}{2^n}$. For each k we ask the SAT oracle, whether the chosen set of $h_1, \ldots, h_m$ has the property that "$\forall x \in S$, one of h_i isolates x". Since there are only $m = 2n$ hash functions this is a SAT query. We pick the least k_0 such that the oracle confirms *isolation*. Then $k_0 \leq k_S + 1$, with probability $\geq 1 - \frac{1}{2^n}$. (We abort if for no k the chosen hash functions achieve *isolation*; this happens with exponentially small probability.) Also by the second part of Lemma 5.21, we know definitely $|S| \leq 2n2^{k_0}$. Denote by $U = 2n2^{k_0}$, then with high probability,

$$\frac{U}{8n} \leq |S| \leq U.$$

这给了我们一个随机化的多名时间算法,使用SAToracle来在O(n)范围内大致的QQS|. 我们想做得更好。 要做到这一点,我们用一个小把戏来扩大准确性。

首先我们造了一套S'这样

$$S' = \underbrace{S \times S \times \cdots \times S}_{\text{毫倍数}} \quad \subseteq \{0,1\}^{nm}$$

m为单数n。 然后我们运行了之前的算法 在S ' 集得到一个估计 U ' 的 S ' 这样

$$\frac{U'}{8nm} \leq |S'| \leq U'.$$

Now we set $e(S) = (U')^{1/m}$. It follows that

$$\frac{e(S)}{(8nm)^{1/m}} \leq |S| \leq e(S).$$

通过选择 m 足够大的多诺米 在n,我们可以得到

$$e(S) \cdot \left(1 - \frac{1}{n^c}\right) \leq |S| \leq e(S),$$

用于任何常数 c.

5.9 独特的满足性:Valiant-Vazirani定理

Assuming NP $\neq$ P, SAT cannot be solved in polynomial time. One may suspect that it is difficult to design a polynomial time algorithm for SAT because the input formula may have myriad truth assignments and it is hard to get hands on one of them. This suspicion leads to the following interesting problem, called USAT (Unique SAT). We are given a formula φ which is guaranteed to be either unsatisfiable or has exactly one satisfying truth assignment. Is USAT solvable in polynomial time? Here we show some evidence that it is unlikely. We prove that if USAT is solvable in polynomial time, then NP = RP.

5.22定理假设有一个多名时间算法A, 对于给定的布尔公式 ,答案是

$$A(\varphi) = \begin{cases} \text{如果 没有满意的指派;如果 有一次满意的指派;如果 有一次以上满意的指派。} \end{cases}$$

Then NP = RP.

证得. 注意,如果算法A得到的公式超过一个令人满意的真理任务,它可以输出任何(YES或NO). 假设A在多诺时段运行,我们为SAT设计一个RP算法. 其理念是使用一些硬币抛掷,将 QQ 转换成更受约束的公式 QQ,这样如果 QQ 无法满足, QQ 也会无法满足. 如果...可以调和,有非三角概率,那么... 一旦我们成功获得这样的一个QQ,我们可以应用算法A来达到它.

To start with assume that φ is satisfiable and let $\#\varphi$ be the number of satisfying truth assignments of φ. In order to convert φ to φ', the algorithm needs an estimate for $\#\varphi$. It is computationally hard to estimate $\#\varphi$. So, we simply pick a k with $1 \leq k \leq n$ at random! The hope is that $2^{k-1} \leq \#\varphi \leq 2^k$. The probability that k satisfies the above condition is $1/n$, as long as $\#\varphi \neq 0$. Assume that we are lucky and k indeed satisfies the above condition. Next we choose a set T of size 2^{k+1}. If $2^{k-1} \leq \#\varphi \leq 2^k$, then $2(\#\varphi) \leq |T| \leq 4(\#\varphi)$, meaning $|T|$ is neither too big nor too small compared to $\#\varphi$. We then setup a universal family of hash functions H from the set of all assignments $\{0,1\}^n$ to T, and we randomly pick a hash function h from H and an element α from T. We will show that, with non-trivial probability, there will be a *unique* satisfying truth assignment x such that $h(x) = \alpha$. So, we consider the question: "Is there a truth assignment t such that $\varphi(t) = 1$ and $h(t) = \alpha$". By Cook's theorem this can be converted to a SAT question φ', and it will have a unique satisfying assignment iff there is a unique t satisfying φ, and is mapped to α by h. If we are lucky in choosing the "correct" k, h and α, φ' will have exactly one satisfying truth assignment.

SAT的算法: 输入:一个公式 $+ +$n变量。

1. Choose a number $1 \leq k \leq n$ uniformly at random.
2. Let $T = \{0,1\}^{k+1}$, let H be a universal family of hash functions from $\{0,1\}^n$ to T.
3. Choose $h \in H$ and $\alpha \in T$ uniformly at random.
4. Let φ' be the Boolean formula from Cook's theorem, encoding the NP predicate "$(\exists t \in \{0,1\}^n)[(\varphi(t) = 1) \wedge (h(t) = \alpha)]$".
5. Run the procedure A for USAT on φ'. *Accept* φ iff $A(\varphi') =$ accept and the truth assignment extracted using A via self-reducibility indeed satisfies φ.

It is clear that, if φ is unsatisfiable then our algorithm will reject it with probability one. So, assume that φ is satisfiable. Then over the random choices of k, h and α, we show that our algorithm accepts φ with probability $\Omega(1/n)$. This probability can then be easily amplified, making it an RP algorithm.

First of all suppose $2^{k-1} \leq \#\varphi \leq 2^k$. The probability that k satisfies the above condition is $1/n$. Under this assumption, we have $2(\#\varphi) \leq |T| \leq 4(\#\varphi)$. We say that two *satisfying* distinct truth assignments a and b (with $a \neq b$) collide under h, if $h(a) = h(b)$. Over the random choice h, let C be the random variable that counts the number of collisions. We first show that the expected number of collisions $\mathrm{E}[C]$ is small. For any fixed $a \neq b$, the probability that a and b collide is $1/|T|$ (this follows from the definition of universal family of hash functions). For any pair of satisfying truth assignments $a \neq b$, let $X_{a,b}$ be a 0-1 random variable such that $X_{a,b} = 1$, if a and b collide under h and $X_{a,b} = 0$, if they don't collide. Its expectation is $\mathrm{E}[X_{a,b}] = 1/|T|$. The number of collisions C is the sum of over all $X_{a,b}$. So, expectation of C is

$$\mathrm{E}[C] = \sum_{(a,b)}$$

以上归纳的范围涵盖所有符合要求的对事实的任务。 这样的对数是

. . .. 因此,

$$\mathrm{E}[C] = \binom{\#\varphi}{2} \frac{1}{|T|} \leq \frac{\#\varphi}{4}.$$

利用马尔科夫的不平等,我们

$$\Pr\left[C \geq \frac{\#\varphi}{3}\right] \leq \frac{E[C]}{\#\varphi/3} \leq \frac{3}{4}.$$

So, with probability at least $1/4$, the number of collisions C is at most $\#\varphi/3$. In general, if there are c collisions, at most $2c$ satisfying truth assignments can participate in collisions. (This can be shown easily by the inductive argument: Take any such x and pick any $y \neq x$ such that $h(x) = h(y)$. Now remove x and y. There can be at most $c-1$ collisions left among the remaining points, and thus at most $2c-2$ points.) Thus, in our case, at most

$2/3 \cdot \#\varphi$ satisfying truth assignments can participate in some collision. It follows that at least $\#\varphi/3$ satisfying truth assignments are mapped to a unique image. Call these $\geq \#\varphi/3$ images in T *good*: i.e., there is a unique satisfying assignment a such that $h(a) = t$. Recall that $|T| \leq 4(\#\varphi)$. So, at least $1/12$ fraction of elements in T are good. Therefore, with probability at least $1/12$, the randomly picked element α is good.

We can now lower bound the probability that φ' has a unique satisfying truth assignment. Assume that φ is satisfiable. Then, with probability $1/n$, the number k chosen by the algorithm satisfies $2^{k-1} \leq \#\varphi \leq 2^k$. Assuming k satisfies the above condition, with probability at least $1/4$, the number of collision C is at most $\#\varphi/3$. Assuming $C \leq \#\varphi/3$, with probability at least $1/12$, the randomly chosen α is good. If α is good, then φ' has exactly one satisfying truth assignment. Putting together,

$$\Pr_{k,h,\alpha}[\varphi' \text{ has a unique satisfying truth assignment}] \geq \frac{1}{n} \cdot \frac{1}{4} \cdot \frac{1}{12} = \frac{1}{48n}$$

正如我们已经指出的,如果 \\ 是不满意的, 我们的算法有 零概率接受它。 如果 QQ 是可匹配的, 我们的算法至少以概率 1/(48n) 接受它。 因此我们的算法是RP算法,成功概率为1/(48n). 我们可以像往常一样,通过多次运行算法来放大这种成功概率. 例如,通过运行算法96n倍,成功概率可以放大为1/2. ♣

5.10 高效扩大

Suppose we have some language L in RP. Thus membership of x in L can be determined by running over witness strings $y \in \{0,1\}^n$, for some $n = |x|^{O(1)}$, and achieving

$$\begin{aligned} x \in L &\implies \Pr_y[D(x,y) = 1] \geq 1/2 \\ x \notin L &\implies \Pr_y[D(x,y) = 1] = 0. \end{aligned}$$

A naive method of amplification of this probability can be obtained by running independent $y_1, \ldots, y_k \in \{0,1\}^n$. Thus, with kn random bits, we can achieve error probability $1/2^k$. A similar process with majority vote can be carried out for BPP langugaes, justified by the Chernoff bound. This probability is the ratio of the number of "bad" coin flips ($2^{k(n-1)}$) over the total number of coin flips (2^{kn}). Is there any way to achieve this reduction of error probability more efficiently?

5.10.1 乔尔-戈德里奇发电机

It turns out that there are several methods. Here is a first try. Let $\{h_s\}$ be a family of universal hash functions from $\{0,1\}^n \to \{0,1\}^n$. Now by randomly choosing s, we simply take the pairwise independent samples $y_i = h_s(i)$ as "witness" strings, for $i = 1, \ldots, k$. (We

consider $i = 1, \ldots, k$ as embedded in $\{0,1\}^n$, as long as $k \le 2^n$.) For our RP langugae L, clearly if $x \notin L$ then all $D(x, y_i) = 0$. Suppose $x \in L$, we estimate the error probability that all $D(x, y_i) = 0$. We do this by using the Chebechev bound. Let Z_i be the 0-1 r.v. such that $Z_i = 1$ iff $D(x, y_i) = 1$.

THEOREM 5.23 *If $x \in L$, then*

$$\Pr_s[(\forall 1 \le i \le k) D(x, y_i) = 0] \le 1/k.$$

The proof is a simple application of the Chebechev bound. Let $x \in L$. Then Z_i are pair-wise independent with expectation $\mu = \mathrm{E}[Z_i] = \Pr_y[D(x,y) = 1] \ge 1/2$, and as a 0-1 r.v. Z_i has variance $\le 1/4$. Then

$$\Pr_s[(\forall 1 \le i \le k) D(x, y_i) = 0] = \Pr_s\left[\sum_{i=1}^{k} Z_i = 0\right] \le \Pr_s\left[\left|\sum_{i=1}^{k} Z_i - k\mu\right| \ge k/2\right] \le 1/k.$$

This is called the Chor-Goldreich generator, which achieves error $1/k$ with $2n$ bits. Of course one can not use this generator to achieve exponentially small error probability since

这被称为Chor-Goldreich生成器,它以2n位元实现错误1/k. 当然,人们不能使用这种生成器来达到指数小的误差概率,因为这需要对散列函数进行指数化的很多评价。

5.10.2 Hash Mixing Lemma and Nisan's Generator

5.10.2 哈什·米辛·莱玛和尼桑发电机

ıded computations. But it can also be used for deterministic amplification.

尼桑的下列发电机主要用于空间范围计算。 但也可以用于决定增殖.

uniformly and independently $s_1, \ldots, s_k$, and consider the following recursive definition of $G_i(y; s_1, \ldots, s_i)$, for $0 \le i \le k$. First $G_0(y) = y$, and for $i \ge 0$,

$$G_{i+1}(y; s_1, \ldots, s_{i+1}) = G_i(y; s_1, \ldots, s_i) \circ G_i(h_{s_{i+1}}(y); s_1, \ldots, s_i),$$

where $\circ$ denotes concatenation. Thus, $G_1(y; s_1) = y \circ h_{s_1}(y)$, $G_2(y; s_1, s_2) = y \circ h_{s_1}(y) \circ h_{s_2}(y) \circ h_{s_1}(h_{s_2}(y))$, etc. It may appear that in $G_1(y; s_1)$, it is not very clever since it uses $3n$ bits to produce only $2n$ (not even quite uniform) bits. The genius of this generator comes when you realize that this generator exponentially stretches its input seed. For G_k, it takes $(2k+1)n$ bits, and produces $2^k n$ bits. Also note that for any particular bit position, it is very simple to compute this bit in a random access manner, without recursion and without even having to compute the earlier bits sequentially. For G_k, the i th block of n bits can be directly obtained as follows. Let the k-bit binary representation of i be $i_1 i_2 \ldots i_k$ then the i th block of n bits is $h_{s_1}^{i_1} h_{s_2}^{i_2} \ldots h_{s_k}^{i_k}(y)$, where $h_{s_j}^1$ denotes h_{s_j} and $h_{s_j}^0$ denotes the identity function. With read-only access to the hash function seeds $s_1, s_2, \ldots, s_k$, this can be computed in space $O(k+n)$. These considerations will become important for its primary

sj 表示身份函数。 以只读权限访问散列函数 s1, s2, ... , sk, 这可以在空间 O(k + n) 中计算。 这些考虑将变得对它的首要任务十分重要。

用于空间范围计算的目的;但对于我们在此扩大的目的而言,这些目的并不重要。

这些东西有多好? 以下lemma是解决这一问题的技术工具.

LEMMA 5.24 (HASH MIXING LEMMA) *Let $\epsilon = 2^{-n/3}$. Then for all subset $E \subseteq \{0,1\}^n \times \{0,1\}^n$, for all but an $\epsilon/4$ fraction of s,*

$$| \Pr_{y \in \{0,1\}^n}[y \circ h_s(y) \in E] - \mu[E] | < \epsilon,$$

where, $\mu[E]$ is the probability measure of the set E, i.e., $\mu[E] = \Pr_{y,z \in \{0,1\}^n}[y \circ z \in E]$.

证得. 我们要估计的分数是"差"的:

$$| \Pr_{y \in \{0,1\}^n}[y \circ h_s(y) \in E] - \mu[E] | \geq \epsilon. \tag{5.1}$$

对于任何固定y,我们定义指示随机变量(随机数大于s)

$$\text{日语}_{\breve{}} = \begin{cases} 1 & \text{if } y \circ h_s(y) \in E \\ 0 & \text{otherwise.} \end{cases}$$

We can write $\Pr_{y \in \{0,1\}^n}[y \circ h_s(y) \in E]$ as a sum of pair-wise independent random variables (random over s)

$$\frac{1}{2^n} \sum_{y \in \{0,1\}^n} \mathbf{1}_{[y \circ h_s(y) \in E]} = \frac{1}{2^n} \sum_y \text{日语}_{\breve{}},$$

然后它就有希望了,

$$\frac{1}{2^n} \sum_{y \in \{0,1\}^n} \mu'[E_y] = \mu[E],$$

where E_y is the fibre-set over y, $E_y = \{z \mid y \circ z \in E\}$, and $\mu'[E_y]$ is the 1-dimensional probabiliy measure $\mu'[E_y] = |E_y|/2^n$.

因此,我们可以利用切别切夫的不平等来估计这种偏差. 因此,"坏"事件(5.1)最多有几率

$$\begin{array}{ll} \textbf{1 2} \\ \text{瓦尔} \end{array} \quad \left[\frac{1}{2^n} \sum_y \text{日语}_{\breve{}} \right] = \frac{1}{\epsilon^2} \frac{1}{2^{2n}} \sum_y \text{变量 [Zhs]}_{\breve{}}] \tag{5.2}$$

$$\leq \frac{1}{\epsilon^2 2^n 4}, \tag{5.3}$$

其中,第一个平等使用Zhs y是双向独立的(比另一个Y),第二个不平等来自一个小事实,即一个0-1随机变量最多有1/4的差异。

Hence, for $\epsilon = 2^{-n/3}$, for all but a $\epsilon/4$ fraction of s,

$$| \Pr_{y \in \{0,1\}^n}[y \circ h_s(y) \in E] - \mu[E] | < \epsilon.$$

♣

We now return to our problem of deterministic amplification for RP languages. Suppose $x \in L$, and the witness set is $W_x \subseteq \{0,1\}^n$. Assume $n \geq 9$ and $\mu(\overline{W_x}) \leq 1/2$ is the error probability of a single random try.

THEOREM 5.25 *If $x \in L$ then*

$$\Pr[G_k(y; s_1, \ldots, s_k) \subseteq \overline{W_x}] \leq (\mu(\overline{W_x}))^{2^k} + (k/4 + 2)\epsilon,$$

where $\epsilon = 2^{-n/3}$.

证得. 直观的想法是使用 kε(kε) 一词来照顾k“ 圆形” 中“ 坏” 散列函数 hsi 的特殊概率, 对于“ 好” 散列函数, 估计值应该像统一一样, 错误的 O(ε)。

For $1 \leq i \leq k$, fixing any $s_1, \ldots, s_{i-1}$, define the set

$$A_i = \{y \mid G_i(y; s_1, \ldots, s_{i-1}) \subseteq \overline{W_x}\}$$

We will apply the Hash Mixing Lemma to the set $E_i = A_i \times A_i$. We say that s_i is bad for E_i (with respect to the fixed $s_1, \ldots, s_{i-1}$) if s_i belongs to the $\epsilon/4$ fraction of exceptional s in the Hash Mixing Lemma. Note that which s_i is bad depends on $s_1, \ldots, s_{i-1}$. However, for any $s_1, \ldots, s_{i-1}$ there can be at most $\epsilon/4$ fraction of s that can be bad, the rest are “good”.

我们用综合标记 $\int$ 超过S和Y的概率测量值。 磕□

$$\Pr[G_k(y; s_1, \ldots, s_k) \subseteq \overline{W_x}] \tag{5.4}$$

$$= \int_{\text{s1 差错}} + \int_{\text{s1 不错}} \int + \cdots + \int_{\text{s1 不错}} \int \cdots \int_{\text{-1 不错}} \int \tag{5.5}$$

$$+ \int_{\text{s1 不错}} \int \cdots \int_{\text{好样的}} \int_{y}. \tag{5.6}$$

i − 1 已经超过概率度量。

现在,我们修复S1, ... SKSSI对Ei来说是不错的S1 . . .. si-1, 并考虑

$$\int_y = \Pr_y[G_k(y; s_1, \ldots, s_k) \subseteq \overline{W_x}].$$

Write $\beta = \mu(\overline{W_x})$. Then $\beta \le 1/2$. We will prove by induction that

$$\int_y \le \beta^{2^k} + 2\epsilon.$$

此概率上的任何上方约束都会成为 s1 不错 好样的
y,作为每个
i 的概率度量。

For $k = 0$, trivially $\int_y = \beta$. For $k = 1$, by the Hash Mixing Lemma

$$\int_y \le \Pr_{y,z}[G_1(y; s_1) \circ G_1(z; s_1) \subseteq \overline{W_x}] + \epsilon \le \beta^2 + \epsilon.$$

Similarly, for $k = 2$,

$$\int_y \le (\beta^2 + \epsilon)^2 + \epsilon \le \beta^{2^2} + 2\epsilon,$$

since $2\beta^2 + \epsilon \le 1$. For the general G_{k+1}, with $k \ge 2$,

$$\int_y \le (\beta^{2^k} + 2\epsilon)^2 + \epsilon \le \beta^{2^{k+1}} + 2\epsilon.$$

Here we used $n \ge 3$ and therefore $\epsilon \le 1/8$.

这证明

$$\Pr[G_k(y; s_1, \ldots, s_k) \subseteq \overline{W_x}] \le (\mu(\overline{W_x}))^{2^k} + (k/4 + 2)\epsilon,$$

For $k \ge 3$, a cleaner expression holds:

$$\Pr[G_k(y; s_1, \ldots, s_k) \subseteq \overline{W_x}] \le (\mu(\overline{W_x}))^{2^k} + k\epsilon.$$

♣

就实现错误概率2-k所需的随机位数而言,Nisan的生成器使用O(n log k)位数.

5.10.3 剩余Hash Lemma和Impagliazzo-Zuckerman发电机

Take ℓ and k as integer parameters. (A reasonable choice will be $\ell \approx k \approx \sqrt{n}$ to balance out the bounds.) This generator produces $k + 1$ blocks of n-bits each, using a source of $3n + k\ell$ bits. Let $\{h_s\}$ be a family of universal hash functions from $\{0,1\}^n \to \{0,1\}^{n-\ell}$. The generator G is defined from $\{0,1\}^{2n} \times \{0,1\}^n \times (\{0,1\}^\ell)^k \to (\{0,1\}^n)^{k+1}$ as follows:

$$G(s, Y_1, Z_1, \ldots, Z_k) = Y_1 \circ Y_2 \circ \cdots \circ Y_{k+1},$$

where $\circ$ denotes concatenation, $s \in \{0,1\}^{2n}$, $Y_1 \in \{0,1\}^n$, $Z_i \in \{0,1\}^\ell$, and

$$Y_{i+1} = h_s(Y_i) \circ Z_i,$$

for $1 \leq i \leq k$.

应该直观地理解这一点:我们首先把Y1照原样。 当我们使用Y1来测试RP语言的会籍时,并非Y1中任意性的所有n位. 因此,我们应该使用散列函数来“提取”一些“剩余”随机性,从而获得 hs(Y1). 我们提供一些新鲜的真随机位数Z1来组成Y2. 然后在Y2重复这个过程,直到Yk.

这个发电机有多好? 解决这一问题的技术工具是以下的"剩餘的哈斯·勒玛"(Leftover Hash Lemma). 在此之前我们需要定义。

对于任何在限值集X上的离散概率分布P和Q,l1-距离的定义是:

$$||P - Q||_1 = \sum_{x \in X} | \; - \mathrm{Q} \; - \mathrm{Q}$$

where P_x and Q_x denote the probability of $x \in X$ under P and Q respectively. For any event $E \subseteq X$, the difference in probability $|P(E) - Q(E)| = |\sum_{x \in E} P_x - \sum_{x \in E} Q_x| \leq ||P - Q||_1$. In fact separating out those $P_x > Q_x$ from $P_x \leq Q_x$, it is easy to see that

$$||P - Q||_1 = 2\,\mathrm{m}^{\mathrm{E}\,+}$$

即是概率的最大差值的两倍。 (这一数量最大E QQP(E) QQ(E)) 称为P和Q之间的统计距离。 然而,一些作者也称QQP QQ1为他们的统计距离.).

A distribution P is caled ϵ-uniform if $||P - U||_1 \leq \epsilon$, where U is the uniform distribution on the same set (same support) as P.

同样,我们可以定义l2距离

$$||P - Q||_2 = \sqrt{\sum_{x \in X} \text{(Px x)2.}}$$

碰撞概率c(P)定义为:

$$c(P) = \Pr_{x, x' \in X}[x = x'] = \sum_{x \in X} P_x^2,$$

根据P, x 和 x ' 独立取用。

接下来的lemma是Cauchy-Schwarz不平等的简单应用.

LEMMA 5.26 *If* $c(P) \leq$ [illegible]

证得. 由考奇-施瓦兹(Cauchy-Schwarz)主演,

$$||P-U||_1^2 \le |X| \sum_{x\in X} (P_x - \frac{1}{|X|})^2 = |X| \left(c(P) - \frac{1}{|X|} \right) \le \epsilon^2.$$

♣

在下一个lemma中,当我们对y有一些限制(例如y取自一些(非)证人的RP语言集)时,我们想计算s∘hs(y)的碰撞概率.

LEMMA 5.27 *Let s and y be taken uniformly and independently from $\mathcal{S}$ and $Y' \subseteq Y$, and let h_s be a hash function from Y to T. Let P be the probability distribution $s \circ h_s(y)$, and let $X = \mathcal{S} \times T$. Then*

$$c(P) = \frac{1 + \frac{|T|}{|Y'|}}{|X|}.$$

Proof. We simply compute, taking $s, s' \in \mathcal{S}$ and $y, y \in Y'$ uniformly and independently,

$$\begin{aligned} c(P) &= \Pr[(s = s') \wedge (h_s(y) = h_{s'}(y'))] \\ &= \frac{1}{|\mathcal{S}|} \Pr[h_s(y) = h_{s'}(y') | s = s'] \\ &= \frac{1}{|\mathcal{S}|} \left(\Pr[y = y'] + \Pr[y \neq y'] \cdot \Pr[h_s(y) = h_s(y') | y \neq y'] \right) \\ &\le \frac{1}{|\mathcal{S}|} \left(\frac{1}{|Y'|} + 1 \cdot \frac{1}{|T|} \right) \\ &= \frac{1}{|X|} \cdot \left(1 + \frac{|T|}{|Y'|} \right) \end{aligned}$$

♣

加上上面的注解,按照顺序应用前两个lemma,

LEMMA 5.28 (LEFTOVER HASH LEMMA) *The distribution $s \circ h_s(y)$ is ϵ-uniform, where $\epsilon = \sqrt{|T|/|Y'|}$.*

现在我们准备证明

THEOREM 5.29 *For $x \in L$ so that $\mu(\overline{W_x}) \le 1/2$,*

$$\Pr[G(s, Y_1, Z_1, \ldots, Z_k) \subseteq \overline{W_x}] \le (\mu(\overline{W_x}))^{k+1} + 2^{-\ell/2}.$$

Proof. The intuitive idea of the proof is as follows. We first charge $\mu(\overline{W_x})$ for the first $Y_1 \in \overline{W_x}$. Then on the condition that $Y_1 \in \overline{W_x}$, the hashed value $h_s(Y_1)$ with appended

Z2使分布相当接近于统一,在用n真随机位取代hs(Y1)◦Z2时,QQY2我们只付出小价钱,一个引申论占据了位置.

现在,我们给出证据。 修补我,我们进入K。 让

$$e_k = \Pr[G(s, Y_1, Z_1, \ldots, Z_k) \subseteq \overline{W_x}],$$

where the probability is over all the unform and independent random choices of $s, Y_1, Z_1, \ldots, Z_k$. Clearly $e_0 = \mu(\overline{W_x})$. For $k \geq 1$,

$$e_k = \mu(\overline{W_x}) \Pr[G(s; Y_2, Z_2, \ldots, Z_k) \subseteq \overline{W_x} \mid Y_1 \in \overline{W_x}],$$

where $Y_2 = \langle h_s(Y_1) \circ Z_1 \rangle$. The probability is still over all $s, Y_1, Z_1, \ldots, Z_k$, except now we have the condition that $Y_1 \in \overline{W_x}$.

因此,我们可以把Wx视为我们莱玛5.27中的子集Y,并估算分布之间的统计距离.

$$\langle s \circ Y_2 \circ Z_2 \circ \ldots \circ Z_k \rangle$$

$$\langle s \circ \hat{Y_2} \circ Z_2 \circ \ldots \circ Z_k \rangle$$

where $\hat{Y_2}$ is independent and uniform over $\{0,1\}^n$, the same length as $Y_2 = \langle h_s(Y_1) \circ Z_1 \rangle$. Let $\epsilon = \sqrt{2^{n-\ell}/(\mu(\overline{W_x})2^n)} = 1/\sqrt{2^\ell \mu(\overline{W_x})}$ as in Leftover Hash Lemma, then

$$\Pr[G(s, Y_2, Z_2, \ldots, Z_k) \subseteq \overline{W_x} \mid Y_1 \in \overline{W_x}] \leq \Pr[G(s, \hat{Y_2}, Z_2, \ldots, Z_k) \subseteq \overline{W_x}] + \epsilon/2,$$

因为无论发生什么事件,例如G下的行为,这两个分布下的概率,只能以l1-Norm距离的一半的最多XQ/2来区别.

由此可见,

$$e_k \leq \mu(\overline{W_x})(e_{k-1} + \epsilon/2).$$

我们只能假设

$$e_{k-1} \leq (\mu(\overline{W_x}))^k + 2^{-(\ell+1)/2},$$

然后我们有了

$$e_k \leq \mu(\overline{W_x}) \left((\mu(\overline{W_x}))^k + 2^{-(\ell+1)/2} + \frac{1}{2\sqrt{2^\ell \mu(\overline{W_x})}} \right) \leq (\mu(\overline{W_x}))^{k+1} + 2^{-(\ell+1)/2}.$$

The last inequality follows from the fact that $\mu(\overline{W_x}) \leq 1/2$, and $\sqrt{\mu(\overline{W_x})/2^\ell} \leq 2^{-(\ell+1)/2}$.

♣

5.10.4 扩大式混合雷玛

Let $G = (V, E)$ be a d-regular undirected graph with $n = |V|$ nodes. The adjacency matrix of G is a symmetric 0-1 matrix defined as

$$M(i,j) = \begin{cases} 1 & \text{if } \{i,j\} \in E \\ 0 & \text{otherwise.} \end{cases}$$

Any real symmetric matrix has n real eigenvalues $\lambda_0, \lambda_1, \ldots, \lambda_{n-1}$ (counting multiplicity) and n corresponding orthogonal unit eigenvectors $v_0, v_1, \ldots, v_{n-1}$. Thus the inner product $\langle v_i, v_j \rangle = \delta_{ij}$, and $Mv_i = \lambda_i v_i$. It is clear that the all-1 vector is an eigenvector with eigenvalue d. Normalizing we let $v_0 = \frac{1}{\sqrt{n}}(1, 1, \ldots, 1)^T$, and $\lambda_0 = d$. Let

$$\lambda = \max_{1 \le i \le n-1} |\lambda_i|$$

表示绝对值的第二大单位值。 d和QQ之间的差距控制了G的扩展属性.

We can express every vector v in terms of the eigenvectors $v = \sum_{0 \le i \le n-1} \alpha_i v_i$, then $Mv = \sum_{0 \le i \le n-1} \alpha_i \lambda_i v_i$.

For any two sets $A, B \subseteq V$, let $E(A, B)$ denote the set of ordered pairs of vertices which are also edges of G,

$$E(A,B) = \{(a,b) \mid a \in A, b \in B, \{a,b\} \in E\}.$$

What should the size of $E(A, B)$ be approximately? If we fix A, and randomly pick a subset B of a certain cardinality $|B|$, then $|E(A, B)|$ is a sum of random variables $X_1 + \ldots + X_{|B|}$, where each X_j is the number of ordered pairs of the form (a, b_j), such that $\{a, b_j\} \in E$, where $a \in A$ and b_j is the j th element of the set B. These X_j are not independent; but each b_j is still uniformly distributed on V. Each X_j can be further expressed as a 0-1 sum of $|A|$ many 0-1 random variables, $X_j = X_{1,j} + \ldots + X_{|A|,j}$, where each $X_{i,j}$ is the contribution of the i th element a_i of A to X_j. Clearly then, the expectation $\mathbf{E}[X_{i,j}] = d/n$, and $|E(A, B)|$ has expectation $d|A||B|/n$.

以下lemma表示QQE(A,B)QQ接近这个平均值,这个接近的程度由QQ控制.

Lemma 5.30 (Expander Mixing Lemma) *For all $A, B \subseteq V$,*

$$\left| \le \lambda \sqrt{|A||B|} \le \lambda n. \right.$$

证得. 表达精巧的功能 QQA和QQB 以编辑者,

$$\chi_A = \sum_{0 \le i \le n-1} \alpha_i v_i, \quad \text{and} \quad \chi_B = \sum_{0 \le i \le n-1} \beta_i v_i,$$

where $\alpha_i = \langle \chi_A, v_i \rangle$ and $\beta_i = \langle \chi_B, v_i \rangle$. Clearly $\alpha_0 = |A|/n$ and $\beta_0 = |B|/n$.

我们的证明试图把“主要术语”和其余部分分开

$$\begin{aligned} |E(A,B)| &= \sum_{i \in A, j \in B} \\ &= \chi^{\top} \\ &= \left(\sum_{0 \le i \le n-1} \alpha_i v_i \right) \cdot \left(\sum_{0 \le i \le n-1} \beta_i \lambda_i v_i \right) \\ &= \sum_{0 \le i \le n-1} \lambda_i \alpha_i \beta_i \\ &= \frac{d|A||B|}{n} + \sum_{1 \le i \le n-1} \lambda_i \alpha_i \beta_i \end{aligned}$$

因此,

$$\begin{aligned} \left| \right. &\le \lambda \sum_{1 \le i \le n-1} |\alpha_i \beta_i| \\ &\le \lambda ||\alpha||_2 ||\beta||_2 \\ &= \lambda ||\chi_A||_2 ||\chi_B||_2 \\ &= \lambda \sqrt{|A||B|} \\ &\le \lambda n \end{aligned}$$

♣

There is an alternative way to state this result that brings home the similarity to the Hash Mixing Lemma. Consider either (1) taking a uniform $x \in V$ and then taking a uniform neighbor $e_s(x)$ (where s picks one of d values uniformly), or (2) taking two independent and uniform points x and y in V. Then for all $A, B \subseteq V$,

$$|\Pr_{x,s}[x \in A, e_s(x) \in B] - \Pr_{x,y}[x \in A, y \in B]| \le \lambda/d.$$

请注意,第一个术语是 QQEA,B)和第二个学期是 QQAB。

12,那么,请说

时间轴:

$\frac{\lambda}{d}$.

Expander grap $_0$. One can explicitly construct expander graph families with $n \to \infty$, and with constant d, and $\lambda \le 2\sqrt{d}$.

For all n' and d', one can construct expander graphs with $n' \le n \le 2n'$ and $d' \le d \le 2d'$, and achieves $\lambda \le d^{9/10}$.

A simple application is the following Karp-Pippenger-Sipser generator. Consider an expander graph with vertex set $\{0,1\}^n$, and degree d. Assume $\lambda \leq d^{9/10}$. With "seed" $z \in \{0,1\}^n$, $G(z)$ produces all d neighbors of z: $G(z) = \{z_1, \ldots, z_d\}$.

THEOREM 5.31 *For $x \in L$ so that $\mu(\overline{W_x}) \leq 1/2$,*

$$\Pr[G(z) \subseteq \overline{W_x}] \leq 2\left(\frac{\lambda}{d}\right)^2.$$

Proof. Let A be the set of vertices all of whose neighbors are in $\overline{W_x}$. Let $B = W_x$. Then $E(A,B) = \emptyset$. By Expander Mixing Lemma,

$$\left| \text{0 -d -A -Wx - = = = 翻译:} \right| \leq \lambda\sqrt{}$$

Since $|W_x| \geq 2^n/2$, we get

$$\frac{|A|}{2^n} \leq 2\left(\frac{\lambda}{d}\right)^2.$$

♣

5.10.5 阿杰泰-科姆尔-塞默尔-埃迪发电机

The idea of using expander graphs can be significantly furthered. The intuitive idea is that, after the initial random vertex, one can take a random walk on the expander graph. If the graph has vertex set $\{0,1\}^n$, and constant degree d, then each additional step of the random walk takes only $O(1)$ random bits. So if we take k steps, we will use only $n+O(k)$ bits. In the meanwhile the names of the vertices the random walk visited form k blocks of n-bits each. Conditional on the previous vertices being in (or not in) the witness set $W_x \subseteq \{0,1\}^n$, the properties of the expander graph ensure that the random walk tends to "disperse" quickly, and therefore these names of the vertices on the random walk in a way simulate true random samples.

Consider an expander graph with vertex set $\{0,1\}^n$, and constant degree d. We will assume $\lambda/d \leq 1/4$; this can be constructed. Take the seed bits $z \in \{0,1\}^n$, and $s_1, s_2, \ldots, s_k \in \{1, \ldots, d\}$. We take $y_0 = z$, and for $i \geq 1$, y_i is the neighbor of y_{i-1} indexed by s_i. Then

$$G(z; s_1, s_2, \ldots, s_k) = \{y_0, y_1, \ldots, y_k\}.$$

THEOREM 5.32 *For $x \in L$ so that $\mu(\overline{W_x}) \leq 1/2$,*

$$\Pr[G(z; s_1, s_2, \ldots, s_k) \subseteq \overline{W_x}] \leq 2^{\Theta(k)}.$$

证得. 证明思想是在扩展图的光谱(egenvalues and eigenvectors)中随机行走的i步骤后表达概率分布, 并以l2-诺姆来分析概率(用l1-Norm表示).

Write $N = 2^n$. Choosing uniformly $z \in \{0,1\}^n$ gives the uniform distribution, which can be represented by a vector in $\mathbf{R}^N$ as $p_0 = (1/N, \ldots, 1/N)^T$. Let $\hat{M} = \frac{1}{N}M$ be the probability transition matrix of the Markov chain defined by the random walk, where M is the adjacency matrix of G. If p is a distribution on G, then after one step of the random walk, the distribution is $\hat{M}p$. Define $P = P_{\overline{W_x}}$ to be the projection matrix, which has a 1 in every diagonal entry corresponding to an element of $\overline{W_x}$, and 0 elsewhere else. Then

$$\Pr[y_1 \in \overline{W_x}] = ||P\hat{M}p_0||_1.$$

Denote by $w_i = \Pr[\{y_1, \ldots, y_i\} \subseteq \overline{W_x}]$. We show by induction that

$$w_k = ||(P\hat{M})^k p_0||_1.$$

Inductive Hypothesis: We assume the distribution of the i th step y_i, conditional on $\{y_1, \ldots, y_{i-1}\} \subseteq \overline{W_x}$, is given by the vector

$$\frac{1}{w_{i-1}}$$

$$w_{i-1} = ||(P\hat{M})^{i-1}p_0||_1.$$

很明显

$$\Pr[\{y_i \in \overline{W_x} \mid \{y_1, \ldots, y_{i-1}\} \subseteq \overline{W_x}] = \frac{1}{w_{i-1}}||(P\hat{M})^i p_0||_1,$$

$$\begin{aligned} w_i &= \Pr[\{y_1, \ldots, y_i\} \subseteq \overline{W_x}] \\ &= \Pr[\{y_1, \ldots, y_{i-1}\} \subseteq \overline{W_x}] \cdot \Pr[\{y_i \in \overline{W_x} \mid \{y_1, \ldots, y_{i-1}\} \subseteq \overline{W_x}] \\ &= w_{i-1} \cdot \frac{1}{w_{i-1}}||(P\hat{M})^i p_0||_1 \\ &= ||(P\hat{M})^i p_0||_1. \end{aligned}$$

我们也可以写

$$\Pr[\{y_i \in \overline{W_x} \mid \{y_1, \ldots, y_{i-1}\} \subseteq \overline{W_x}] = \frac{w_i}{w_{i-1}}.$$

The distribution of y_{i+1}, conditional on $\{y_1, \ldots, y_i\} \subseteq \overline{W_x}$, can be represented by the vector

1wi/ wi−1 (韩语) $\hat{M}P\left(\frac{1}{w_{i-1}}$ˆ(P ˆ)i−1p0 $\right) = \frac{1}{w_i}$M(P ˆ M)ip0. ˆ.

这完成了上岗培训。

(We can also show $\Pr[\{y_0, y_1, \ldots, y_k\} \subseteq \overline{W_x}] = ||(P\hat{M})^k \hat{M} p_0||_1$. But this only contributes another factor of 1/2.)

Now write any vector v in terms of the eigenvectors v_i, $v = \sum_{0 \le i \le n-1} \alpha_i v_i$, then

$$\begin{aligned} ||P\hat{M}v||_2 &\le ||P\hat{M}\alpha_0 v_0||_2 + ||P\hat{M} \sum_{1 \le i \le n-1} \alpha_i v_i||_2 \\ &\le ||P\alpha_0 v_0||_2 + ||\hat{M} \sum_{1 \le i \le n-1} \alpha_i v_i||_2. \end{aligned}$$

Here we used the fact that $\hat{M}v_0 = v_0$, and the orthogonal projection matrix P has ℓ_2^2-norm $||P||_2 \le 1$. More concretely, for any vector u, $||Pu||_2^2 = \sum_{i \in \overline{W_x}} u_i^2 \le \sum_{1 \le i \le N} u_i^2 = ||u||_2^2$.

Recall $v_0 = \frac{1}{\overline{N}}(1, \ldots, 1)^T$. Note that $||Pv_0||_2^2 = \frac{|\overline{W_x}|}{N} = \mu(\overline{W_x})$, and $\hat{M}v_i = (\lambda_i/d)v_i$, it follows that

$$\begin{aligned} ||P\hat{M}v||_2 &\le \sqrt{\mu(\overline{W_x})}|\alpha_0| + ||\sum_{1 \le i \le n-1} (\lambda_i/d)\alpha_i v_i||_2 \\ &= \sqrt{\mu(\overline{W_x})}|\alpha_0| + \sqrt{\sum_{1 \le i \le n-1} (\lambda_i/d)^2 |\alpha_i|^2} \\ &\le \sqrt{\mu(\overline{W_x})}||\alpha_0 v_0||_2 + (\lambda/d)||\sum_{1 \le i \le n-1} \alpha_i v_i||_2 \\ &\le \left(\sqrt{\mu(\overline{W_x})} + \frac{\lambda}{d}\right)||v||_2, \end{aligned}$$

最后一个不平等的原因是α0v0和P

$\sum_{1 \le i \le n-1} \alpha_i v_i$ are orthogonal projections

终于,由考奇-施瓦兹,

$$\begin{aligned} w_k &= \Pr[\{y_1, \ldots, y_k\} \subseteq \overline{W_x}] \\ &= ||(P\hat{M})^k p_0||_1 \\ &\le \sqrt{\mathrm{N}}||(\mathrm{P}\ )^{\mathrm{k}}\mathrm{p0}||2 \\ &\le \sqrt{N}\left(\sqrt{\mu(\overline{W_x})} + \frac{\lambda}{d}\right)^k ||p_0||_2 \\ &\le \sqrt{N}\left(\sqrt{1/2} + 1/4\right)^k \frac{1}{N} \\ &= 2^{-\Theta(k)} \end{aligned}$$

为了实现错误概率2−k,AKS生成器使用n + O(k)真随机位.

第6章

Hartmanis 猜想

在这场讲座中,我们讨论了伯曼-哈特马尼斯异形主义猜想.

6.1 导言

The Berman-Hartmanis conjecture, also known as the *isomorphism conjecture* is related to NP vs. P question. We start with a definition. Two languages A and B are said to be p-isomorphic (polynomial time isomorphic), if there is a polynomial computable, polynomial time invertible, 1-1 and onto reduction from A to B. Formally, we need a function $f : \Sigma^* \mapsto \Sigma^*$, such that i) f is 1-1 and onto, ii) $x \in A \iff f(x) \in B$, iii) f polynomial time computable, iv) f^{-1} is polynomial time computable. Berman and Hartmanis observed that all "known" NP-Complete problems are p-isomorphic to each other. Here, "known" refers to, for example, all problems in Garey and Johnson. We will get into to the issue of proving this claim, shortly. Based on this claim, they formulated the now-famous conjecture:

Berman-Hartmanis isomorphism conjecture: Any two NP-Complete sets are p-isomorphic to each other.

The most interesting aspect of the conjecture is that, if it is true then NP $\neq$ P. Because, if NP = P, then, even finite sets would be NP-Complete. But, a finite set cannot be isomorphic to a infinite set like SAT. The conjecture has been studied extensively in the past two decades. If the conjecture is true, then not even sparse sets can be NP-Complete, because SAT is an "exponentially" dense set and there cannot be a polynomial time isomorphism from a dense set to a sparse set. This raises an interesting issue of whether sparse sets can be NP-Complete. Mahaney's theorem (which we proved in an earlier lecture) shows that no sparse set can be NP-Complete, unless NP=P. The isomorphism conjecture remains open. Some recent evidence shows that it may be false. We will discuss this evidence at the end of the lecture.

我们现在回到Berman和Hartmanis的说法:所有“已知”的NP-Complete问题都是相互同质的。 索赔的依据是他们遵循的定理。 假设存在可计算,不可逆的多诺时间,1-1以及长度从A到B和B到A的递减. 然后,A和B彼此是p-异形的. (我们稍后将更正式地说明这一定理。)他们注意到,所有“已知”的NP-完成问题都存在这种削减。 以上定理的证明基于坎托尔的定理. 我们先证明坎托尔的定理,然后继续讨论异态主义猜想。

6.2 坎托尔定理

定理6.1 如果A和B的设定中存在A至B的1 − 1地图和B至A的1 − 1地图,那么A和B之间就存在1 − 1对应(即1 − 1和上).

证得.

让 f 成为 A 到 B 的 1 - 1 地图, g 成为 B 到 A 的 1 - 1 地图。

如果f或g也被输入,则为1 - 1 个信件。 从此以后,我们将假定f和g都没有出现,并证明仍然有1 -1份信函。

Consider the following two sequences of subsets of A and B: $A_0 = A$, $B_0 = B$, and the rest of the A_i and B_i are given by:

$$A_i = g(B_{i-1}) \qquad B_i = f(A_{i-1})$$

这两个序列在图6.1中以图片形式显示.

The first thing to note about this sequence is that $A_i \subset A_{i-1}$ (i.e., A_i is a proper subset of A_{i-1}) for all $i \in \mathbf{N}$, and the same is true for the B_i. To see this, observe first that it is clear from the definitions of A_i and B_i that $A_i \subseteq A_{i-1}$ and $B_i \subseteq B_{i-1}$. We will show (by induction on i) that successive sets are not equal.

Observe that $A_1 = g(B_0)$. Since g is not onto, there exists an element in A_1 that is not in A_0. Similarly, $B_1 \subset B_0$.

Now, assuming that $B_i \subset B_{i-1}$, let x be any element of $B_{i-1} - B_i$. Note first that $g(x) \in A_i$, since $A_i = g(B_{i-1})$. We claim that $g(x) \notin A_{i+1}$. If $g(x) \in A_{i+1}$, then $\exists y \in B_i$ such that $g(y) = g(x)$. Since $x \notin B_i$, this would imply that two distinct elements (x and y) of B are mapped to the same element of A, contradicting the stipulation that g is $1 - 1$. This completes the induction and proves that $A_i \subset A_{i-1}$ for all $i \in \mathbf{N}$. A similar argument can be made for the B_i.

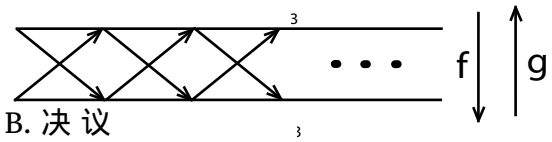


图6.1:应用函数f(x)和g(x)生成的序列的图示. 爱人被列为红宝石, Bi也一样。

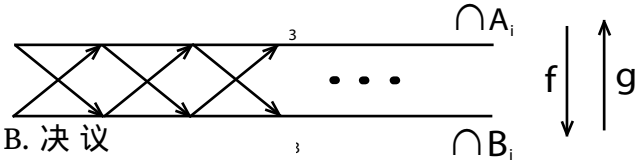


图6.2:6.1的图片描述修改后,包括了TQQ套集.

Now, with these facts in hand, we note that the set A can be decomposed as follows:

现在,在掌握这些事实后,我们注意到,A组可以分解如下:

$$A = (A_0 - A_1) \cup (A_1 - A_2) \cup \cdots \cup (A_0 \cap A_1 \cap A_2 \cap \cdots)$$

$$= \left(\bigcup_{i=0}^{\infty} (A_i - A_{i+1}) \right) \cup \left(\bigcap_{i=0}^{\infty} A_i \right)$$

(Ai-Ai+1) (中文(简体)).

Similarly,

同样,

$$B = \left(\bigcup_{i=0}^{\infty} (B_i - B_{i+1}) \right) \cup \left(\bigcap_{i=0}^{\infty} B_i \right)$$

(Bi-Bi+1) (中文(简体)).

Thus, for any element $x \in A$, either $x \in (A_i - A_{i+1})$ for some (unique) i, or $x \in A_i$ for all i (see figure 6.2). We'll define a function $F : A \to B$, which will be a $1-1$ correspondence, as follows: if $x \in \bigcap_{i=0}^{\infty} A_i$, then $F(x) = f(x)$. Otherwise, $x \in (A_i - A_{i+1})$ for some i. In this case,

$$F(x) = \begin{cases} f(x) & \text{if } i \text{ is even} \\ g^{-1}(x) & \text{if } i \text{ is odd} \end{cases}$$

QQ f(x) 如果偶数为数

That is, an element x of A is mapped by f or g^{-1}, depending on which subset(s) of A it falls into. Note that for all $i > 0$, A_i contains only elements that are in the

也就是说,A的一个元素x被f或g−1映射到B,这取决于它属于哪一个子集。 注意,对于所有i > 0,Ai只包含在g图像中的元素,因此g−1(x)在我们使用它的所有情况下都得到了很好的定义. 我们仍然要表明F是1 -1,并进入。

We start with $1-1$. We first consider an element $x \in \bigcap_{i=0}^{\infty} A_i$ and argue that no other element of A maps to $F(x)$ under F. Here we have $F(x) = f(x)$. Moreover, note that $f(x) \in \bigcap_{i=0}^{\infty} B_i$. Consider any element $x' \in A$ with $x' \neq x$. If $x' \in \bigcap_{i=0}^{\infty} A_i$, $F(x') = f(x')$ and as f is 1-1, $F(x) \neq F(x')$. If $x' \notin \bigcap_{i=0}^{\infty} A_i$, then x' is mapped to some element in $B_i - B_{i-1}$, for some i. Hence, such an x' cannot be mapped to an element in $\bigcap_{i=0}^{\infty} B_i$. So, $F(x) \neq F(x')$. Next consider an element $x \in (A_i - A_{i+1})$ for some i. We have two cases: if i is even, then $f(x) \in (B_{i+1} - B_{i+2})$ – i.e., the largest j for which $f(x) \in B_j$ is odd; if i is odd, then $g^{-1}(x) \in B_{i-1} - B_i$ – i.e., the largest j for which $g^{-1}(x) \in B_j$ is even. Thus, an element of A mapped by f never collides with an element mapped by g^{-1}. Since f and g are both $1-1$, elements mapped by f never collide with each other, and likewise for g^{-1}. Thus, F is $1-1$.

It may be obvious at this point that F is also onto, but we will prove it for good measure. Let y be an element of B. Suppose $y \in \bigcap_{i=0}^{\infty} B_i$. Clearly, $y \in B_1$ and hence, for some $x \in A$, $f(x) = y$. For any i, notice that $y \in B_i = f(A_{i-1})$ and f is 1-1 and hence, $x \in A_{i-1}$. Thus, $x \in \bigcap_{i=0}^{\infty} A_i$, and hence, $F(x) = f(x) = y$. Suppose, $y \in B_i - B_{i+1}$ for some i, then we again look at two cases: if i is even, then y is mapped to by $g(y)$; if i is odd, then i is mapped to by $f^{-1}(y)$, which is guaranteed to exist, since every odd B_i is the image under f of some A_j. Thus, F is $1-1$ and onto.

6.3 Myhill's Theorem

6.3 Myhill的定理

roven using a similar proof is Myhill's theorem.

使用类似证据可以证明的一个有趣的定理是Myhill的定理。

every other RE-complete set.

6.2 定理 6.2 每个RE完成的集都有1-1到递归,不可逆的映射到其他每个RE完成的集.

any two undecidable, r.e. sets are just computable renaming of each other. We will not prove the theorem in this class.

这个定理表明,基本上只有“一个”的RE-完成集。 换句话说,任何两个不可更改的,r.e. 组合只是相互的可计算重命名。 我们不会证明这个班里的定理

6.4 The Berman-Hartmanis Conjecture

6.4 伯曼-哈特马尼斯猜想

are p-isomorphic to each other. The conjecture is based on the following theorem.

Berman和Hartmanis猜想,所有的NP - 完成问题彼此是p-异形的. 猜想基于以下定理.

$g : \Sigma^* \mapsto \Sigma^*$ *that satisfy*

定理 6.3 让L1和L2成为两套,这样就有函数 f: Σ*7→Σ*和 g: Σ*7→Σ*满足

1. 国家 f和g为多诺可计算时间。

2. 国家 f和g为1-1。

3 (中文(简体)). f和g的长度在增加。 也就是说,对于任何x, +|f(x)| > +|x|和+|g(x)| > +|x|.

4 , 4 , 4 , 4 , 4 , 4 , 4 f 是从L1减为L2,g是从L2减为L1.

5. *f and g are polynomial time invertible. That is, there is a polynomial time algorithm that given x either outputs the (unique) y such that $f(y) = x$ or says no such y exists. Similarly, such an algorithm exists to compute g^{-1}.*

然后,L1和L2是p-异态的. 也就是说,有一个多诺时数可计算,多诺时数不可逆的F从L1到L2的还原是1-1和上.

Proof. We follow the proof of Cantor's theorem. Let $A_0 = B_0 = \Sigma^*$. Then, as in Cantor's proof, define

$$A_i = g(B_{i-1}) \text{ and } B_i = f(A_{i-1})$$

首先要注意的是,长度n的字符串 x不能在An+1中,因为f和g的长度(严格)都在增加. 因此,TQQ

$$F(x) = \begin{cases} \text{QQ f(x) 如果偶数为} \\ \text{g}-1\text{(x) 如果我是奇} \end{cases}$$

That completes the definition of F. We数 satisfies the required properties:
这完成了F的定义. 我们需要指出,F满足了所需的属性:

1. *F is 1-1 and onto:* The proof is same as Cantor's proof.
1. 国家 F是1-1, 与Cantor的证明相同。
2. *F is a reduction from L_1 to L_2 :* For any x, $F(x)$ is either $f(x)$ or $g^{-1}(x)$. The claim
2. 国家 F是从L1到L2的还原:对于任何x,F(x)是f(x)或g−1(x). 索赔的根据是,f是从L1减少到L2,g是从L2减少到L1。

3. *F is polynomial time computable :* Given an x, all we need to do is, determine whether we are in the even or odd case. After that, we need to either compute $f(x)$ or $g^{-1}(x)$ and both these can be done in polynomial time. Notice that $x \in A_1$ iff $g^{-1}(x)$ exists. Similarly, $x \in A_2$ iff $f^{-1}(g^{-1}(x))$ exists and $x \in A_3$ iff $g^{-1}(f^{-1}(g^{-1}(x)))$ exists. In general, to determine whether $x \in A_k$, we need simply apply g^{-1} and f^{-1} alternatively k times and check whether that element exists or not. Now our goal is to find the unique d such that $x \in A_d - A_{d+1}$. We already noted that $x \notin A_n$, where $n = |x|$. Hence, $d \leq n$. Recall our assumption that f^{-1} and g^{-1} are polynomial time computable. So, the idea is to start with x and apply g^{-1} and f^{-1} alternatively: compute

$$x_0 = x, x_1 = g^{-1}(x), x_2 = f^{-1}(x_1), x_3 = g^{-1}(x_2), x_4 = f^{-1}(x) \ldots$$

Notice this, the strings in this sequence get shorter as we move on (i.e. $|x_{i+1}| < |x_i|$),
注意到这一点,这个序列中的字符串在我们继续前进时会缩短(即. |xi+1| < |xi|xi|),因为f和g的长度在增加. 过了几步,我们就得停下来, 泡泡

either f^{-1} (or g^{-1}) does not exist or we end up with a string of length 1. Suppose we computed till x_i succesfully and x_{i+1} does not exist. Then, $x \in A_i$ but $x \notin A_{i+1}$, (i.e. $d = i$). As f^{-1} and g^{-1} are computable each step of this process can be carried out in polynomial time. As $d \leq n$, we will need at most n steps.

利用这一定理,Berman和Hartmanis观察到,任何两套"已知的"NP-Complete是相互的p-异形。 因为,对于任何"已知的"NP-完成问题,NP-完成的减少可以很容易地被修改,使它达到1-1,长度增加和多诺米时间无反转! 作为例子,我们将考虑SAT和汉密尔顿电路。

Let L be any lnaguage in NP and f be the reduction from L to SAT. We will construct a new reduction f' from L to SAT which is length increasing and 1-1. Let $x = a_1 a_2 \dots a_n \in \{0,1\}^*$ be the input string of length n. Reduction f' first computes $\varphi = f(x)$. Then construct a new formula φ'. φ' uses all the boolean variables of φ and n new variables $z_1, z_2, \dots, z_n$. New formula $\varphi' = \varphi \wedge (l_1 \vee l_2 \vee \dots \vee l_n)$, where the lieteral l_i is z_i, if $a_i = 1$ and $\overline{z_i}$ if $a_i = 0$. It is clear that φ is satisfiable iff φ' is satisfiable and hence, f' is a reduction from L to SAT. Moreover, $|\varphi'| > |x|$. Finally, given φ', we can easily extract the string x. Thus, f' is polynomial time invertible.

A similar trick can be used in case of Hamiltonican circuit (HC). Let L be any language in NP and f be a reduction from L to HC. We construct a new reduction f'. Given input $x = a_1 a_2 \dots a_n$, f' first computes the graph $G = f(x)$. f' will output a new graph G'. G' is obtained by first adding $3n+1$ new vertices $s_1, s_2, \dots, s_{n+1}$, $u_1, u_2, \dots, u_n$, and $v_1, v_2, \dots, v_n$ to G. For each $1 \leq i \leq n$, if $a_i = 0$ then add the edges $s_i \to u_i \to v_i \to s_{i+1}$; if $a_i = 0$ then add the edges $s_i \to v_i \to u_i \to s_{i+1}$. Thus, we have created some "chain" of vertices, that depend on input x. Finally, take any edge (s,t) in G and "replace" it with the chain constructed from x. Namely, remove the egde (s,t) and add edges (x, s_1) and (s_{n+1}, t). It is clear that G has a Hamiltonican circuit iff G' has one. Moreover, $|G'| > |x|$ and given G', we can extract x easily.

对于所有"已知的"NP-完成问题,比如在Garey和Johnson列出的问题,我们可以做这样简单的技巧,使缩短的长度增加,1-1。 然后,上面的定理意味着所有已知的NP-Complete问题彼此是p-异形的. 基于这一证据,伯曼和哈特马尼斯猜想,所有的NP-Complete问题都相互具有p-异形性. 但是,现在一天,人们普遍认为它是法斯尔。 约瑟夫-杨猜想为这个信念提供了一些证据.

6.5 约瑟夫-杨猜想

Joseph and Young猜想中说,有一种NP-Complete语言A,因此没有从SAT到A的不可逆时间。 这种语言不符合定理6.3的要求。 语言A的构建基于存在单向函数的假设. 随便说说,一个功能是单向的,如果它

is computable in polynomial time, but not invertible in polynomial time. It is not known whether one-way functions exist (existence of one-way functions imply NP $\neq$ P). But, there are some candidate functions believed to be one-way. One example is multiplication. Given two numbers a and b, it is easy to compute their product ab. But the inverse of multiplication, namely, factoring is believed to be hard. Suppose we have a one-way function f. It is easy to show that for any 1-1 function g, the language g(SAT) is NP-Complete. In particular, $f(SAT)$ is NP-Complete. Now, the conjecture is that there is no invertible reduction from SAT to $f(SAT)$.

第7章

交互式证明系统

在第一章中,我们定义了NP语言的验证模式. 让我们以SAT作为NP的范例语言. 假设证明人想证明 给定的配方是可喜的 他可以通过将令人满意的真相指定为证明来完成这项工作。 核查可在多名制时核实指定网是否确实符合公式。 假设这个公式其实是不可改变的. 证明人如何提供简洁的证明,证明公式是不可替代的? 换句话说,是否有一种不满足的证明系统? 接下来的两个章节为NP以外的语言开发了证明系统. 在这些证明系统中,证明人和验证人有许多回合的互动. 我们以互动证明系统为例。

7.1 互动证据-实例

We explain IP with an example. We say that two graphs $G = (V, E)$ and $H = (V', E')$ are isomorphic, if relabeling G would give H. Formally, G and H are isomorphic, if there is a one-one and onto function $\sigma : V \to V'$, such that $(u, v) \in E$ iff $(\sigma(u), \sigma(v)) \in E'$. If G and H are isomorphic, we write it as $G \cong H$. For a permutation σ on V, we denote $\sigma(G) = (V, E_\sigma)$, where, $E_\sigma = \{(\sigma(u), \sigma(v))|(u, v) \in E\}$. Define the following two languages:-

$$GI = \{(G, H)|G \cong H\}$$

$$GNI = \{(G, H)|G \ncong H\}$$

It's easy to see that GI is in NP. We can simply guess a σ and verify whether $\sigma(G) = H$. What about GNI? It's in co-NP, by definition. But what about a proof system for GNI? No concise proof for GNI is known. Here, we describe what's called an interactive proof for GNI.

L的交互式证明系统(IP)由两名玩家组成,证明人和验证人. 证明人声称一个字符串x在L,验证人必须验证. 证明人

has no computational limitations, he is all powerful, whereas, the verifier is a probabilistic polynomial time player. They can interact with each other. The verifier can toss some coins and ask the prover some question and prover answers it. This process of interaction can go on for some polynomial number of rounds. At the end, the verifier accepts or rejects the input. If $x \in L$, then we require that the verifier accept with probability $\geq 3/4$ and if $x \notin L$, then he accepts with probability $\leq 1/4$. As usual, the values $3/4$ and $1/4$ are arbitrary and can be amplified.

We now describe an IP for GNI. So, let G and H be the two input graphs and the prover claims that $G \ncong H$. The verifier tosses a coin. If HEAD, he chooses G else chooses H. Let the chosen graph be S. Then, he chooses a random permutation σ and sends $S_\sigma = \sigma(S)$ to the prover. The prover has to send the original coin toss of the verifier. If G and H were really non-isomorphic, then, S_σ will be isomorphic to exactly one of G or H and the prover can find it easily and correctly guess the coin toss. On the other hand, if he were cheating and $G \cong H$, S_σ be isomorphic to both G and H. So the prover, can at most make a wild guess at what the coin toss was. He will have probability only $1/2$ to succeed. We can run this process a polynomial number of times to make the probability of cheating to be exponentially small.

It's easy to see that $NP \subseteq IP$. The verifier keeps quiet! On input x, the prover sends him the certificate that $x \in L$. The verifier verifies whether it's a valid certificate.

7.2 阿瑟-梅林游戏

Arthur-Merlin游戏是IP的限制版本. 在AM中,我们只允许一回合的互动 贝文的证明和验证。 这里我们称证明人为梅林,而马鞭草人称亚瑟. 梅林是个强大的魔术师 而亚瑟是个卑微的多诺米时代君主 梅林想证明一个字符串x在L. 和IP一样,他通过和亚瑟互动来完成这项工作.

Definition (AM): $L \in AM$ iff $\exists$ a polynomial time predicate P such that

$$x \in L \Rightarrow Pr_y[\exists z, P(x, y, z) = 1] \geq \frac{3}{4}$$

$$x \notin L \Rightarrow Pr_y[\exists z, P(x, y, z) = 1] < \frac{1}{4}$$

字符串y和z以x的长度为多边界限。

Remarks: We can see that $AM \subseteq IP$. AM is a restricted version of IP. Arthur simply tosses coins and chooses a random string y and sends it to Merlin. Merlin sends back a certificate z. Arthur, evaluates the polynomial predicate $P(x, y, z)$. If $x \in L$, for most coin tosses y of Arthur, Merlin will have some valid proof z. On the other hand, if $x \notin L$, for most y, Merlin will not have a valid proof z.

同往常一样,我们可以扩大成功的可能性。 亚瑟,选择许多随机弦,y1,y2,. . . .. 把他们都送到梅林那里 梅林以z1,z2回覆. . . .. 兹尔. 上游P被修改为P ' ,将模拟P(x, y1, z1). . . .. P(x, yr, zr) (英语). 那就拿多数 我们可以使用Chernoffbounds来显示成功概率在r中成倍放大. 因此,我们可以替换3个

4乘1 − 1 2n,1 4乘1 2poly(n).

Now suppose we have a university president who wants no error. He wants probabilities 1 and 0, respectively. For example, if $x \notin L$, then $\forall y \forall z$ it is false. Indeed, we can consider, four possibilities here.

(1.) $\begin{cases} x \in L & \text{then for most } y, \text{ there exists } z \text{ that makes it true} \\ x \notin L & \text{then for most } y, \text{ for all } z \text{ it is untrue} \end{cases}$

(2.) $\begin{cases} x \in L & \text{then for all } y, \text{ there exists } z \text{ that makes it true} \\ x \notin L & \text{then for most } y, \text{ for all } z \text{ it is untrue} \end{cases}$

(3.) $\begin{cases} x \in L & \text{then for all } y, \text{ there exists } z \text{ that makes it true} \\ x \notin L & \text{then for all } y, \text{ for all } z \text{ it is untrue} \end{cases}$

(4.) $\begin{cases} x \in L & \text{then for most } y, \text{ there exists } z \text{ that makes it true} \\ x \notin L & \text{then for all } y, \text{ for all } z \text{ it is untrue} \end{cases}$

声明(1.)只是AM的定义。 陈述(3.)和(4.)都是属于国民党的案件。 声明(2)定义了我们所称的AM1,一个侧式错误版本的AM. 我们很快会讨论的

We note that we can always transform $\exists z_{|z| \leq n^{O(1)}} P(x, y, z) = 1$ to a satisfiability issue by using Cook's theorem. Therefore, the general form of AM is:

For $x \in L$, then for most y, we get $\varphi_{x,y} \in$ SAT.

For $x \notin L$, then for most y, we get $\varphi_{x,y} \notin$ SAT.
where the map $x, y \mapsto \varphi_{x,y}(z)$ is computable in polynomial time.

We can think of AM as follows: We think of AM by writing AM = BP $\cdot NP$.

我们定义AM1为陈述(2. 形式上

Definition (AM_1): $L \in AM_1$ iff $\exists$ a polynomial time predicate P such that

$$x \in L \Rightarrow Pr_y[\exists z, P(x, y, z) = 1] = 1$$

$$x \notin L \Rightarrow Pr_y[\exists z, P(x, y, z) = 1] \leq \frac{1}{4}$$

AM_1 can be thought of as $co-RP.NP$. It is immediate that $\text{AM}_1 \subseteq$ AM. We have the following surprising theorem. We now prove that $AM = AM_1$. Equivalently, BP.NP=coRP.NP.

Theorem: $AM = AM_1$.
Proof: Let L be in AM and A be the underlying polynomial predicate. We amplify the success probability such that,

$$x \in L \Longrightarrow P_{y \in \{0,1\}^m}[\exists z \in \{0,1\}^r, A(x, y, z)] \geq 1 - \frac{1}{m}$$

$$x \notin L \Longrightarrow P_y[\exists z, A(x,y,z)] \leq \frac{1}{m}$$

Recall that a set $E \in \{0,1\}^m$ is called thin if $|E| \leq 2^m/m$. E is called fat if E^c is thin. A set may be neither thin nor fat. Let $k \sim \log m$.

我们已经证明了以下关于脂肪和薄片的莱玛: - Lemma:如果E是脂肪,那么,

$$Pr_{s_1,s_2,\ldots s_m \in \{0,1\}^m}[\bigcap_{i=1}^{m}(E \oplus s_i) \neq \phi] = 1$$

如果E是瘦的,那么,

$$\text{Prs1,s2,... ...sm } [\bigcap_{i=1}^{m} E \oplus s_i \neq \phi] \leq \frac{1}{2^{m(k-1)}}$$

Fix an input x. Let $E_x = \{y | \exists z, A(x,y,z) = 1\}$. If $x \in L$, then, E is fat. If $x \notin L$, E is thin. We shall construct a new one sided error protocol for L as follows:- Suppose Merlin claims $x \in L$. He will prove this claim by showing that E_x is fat. To do this, Arthur will give him set of random shift strings $s_1, s_2, \ldots s_m$. Merlin will produce a $y \in \bigcap_{i=1}^{m} E \oplus s_i$. By above lemma, for any set of random strings, such a y exists. But, Merlin has to prove that y, indeed, has this property. Equivalently, for $1 \leq i \leq m$, $y \oplus s_i \in E$. To prove this, he produces, $z_1, z_2, \ldots z_m$ such that, $A(x, y \oplus s_i, z_i) = 1$. On the other hand, if $x \notin L$, by the second part of the above lemma, the probability that such a y exists is very small.

Formally, we now present a polynomial predicate A' as required by the definition of AM_1. $A'(x,s,z')$ takes three arguments, where x is the input string, $s =< s_1, s_2, \ldots s_m >$ will be a string of length m^2; $z' =< y', z_1, z_2, \ldots z_m >$ will be a string of length $m + mr$, where r is the length of z in the original AM protocol.

$$A'(x, < s_1, s_2, \ldots s_m >, < y', z_1, z_2, \ldots z_m >) = \bigwedge_{i=1}^{m}(A(x, y' \oplus s_i, z_i)$$

正如已经解释的,在"肥胖"lemma的帮助下,我们可以很容易地核实A ' 是否拥有所需的属性。 ♣

我们现在比较IP和AM。 一个主要的区别是,在IP中,亚瑟的硬币掷币是私人的,而在AM中,硬币是公开的. 在IP中,梅林无法看到亚瑟的硬币掷掷物,而在AM中,他可以看到它们. (在我们对AM的定义中,Merlin可以看到"y"然后说出他的证据"z"。)对Merlin没有什么隐瞒。 为了更好地解释这一点,请回顾一下我们的"图非异形"知识产权协议。

Suppose $G_1 \ncong G_2$. The original interactive proof system worked as follows:- the verifier takes $b \in_R \{0,1\}$ and $\sigma \in_R S_n$. Then, sends over $\sigma(G_b)$ to the prover (and hides his coin toss). Then the prover sends back b' and claims that $G_{b'} \cong \sigma(G_b)$. The verifier then verifies

this. We can't use this protocol straightaway in the Arthur-Merlin setup. In AM, Merlin invites Arthur to send the bits, i.e., σ and b. Now Merlin can easily cheat Arthur.

But, we can design a clever AM protocol for GNI. Play this game. $\mathrm{Aut}(G)$ is defined to be the set of all graphs isomorphic to G. Look at $\mathrm{Aut}(G_0) \cup \mathrm{Aut}(G_1)$. This set is large, in fact much larger than one of $\mathrm{Aut}(G_0)$ or $\mathrm{Aut}(G_1)$. If $G_0 \cong G_1$ then these sets are the same. Otherwise, we have twice the size for this larger set. Merlin tells us which one of the sets is larger. Merlin engages in a proof of the size of these.

7.3 梅林 - 亚瑟运动会

MA与AM相似,只是梅林先动. 他向Arthur(一个概率论的多诺时间玩家)提出一个证明,他用动词来形容证明是否有效.

定义(MA):如果存在一个可计算出来的多元时间(.,.)上游A(.,.),则L在MA中,

$$x \in L \Longrightarrow \exists z Pr_y[A(x,y,z)=1] \geq \frac{3}{4}$$

$$x \in L \Longrightarrow \forall z Pr_y[A(x,y,z)=1] < \frac{1}{4}$$

,其中的Q|y|和Q|z|是多语种在X|x|的边界.

Remarks: Here, if $x \in L$, a single proof "z" should work for most of the coin tosses "y" of Arthur. If $x \notin L$, no matter what proof Merlin presents, Arthur is unlikely to accept it. MA is like N.BPP. Similar to NP, but the verfication procedure is in in BPP. The proof "z" is a *publishable proof*. You can write it down and then check it. It's like a relaxed way of proving.

再说一遍,我们可以从3号放大这个 4至1 - 2 - q(n),用于任何多诺的q(n).

Proposition: $BPP \subseteq MA$.
Proof: Arthur simply ignores Merlin, and just tosses some coins and simulats the BPP algorithm. As we remarked, MA is like $N.BPP$, so it's quite obvious that $BPP \subseteq N.BPP = MA$. ♣

It's easy to see that,
Proposition: $NP \subseteq MA$. ♣

如AM一样,我们可以定义MA的一个侧式错误版本,即MA1:- 定义(MA1):如果存在一个可计算出来的多诺时间上游A(.,.),则L在MA中,

$$x \in L \Longrightarrow \exists z Pr_y[A(x,y,z)=1] = 1$$

$$x \in L \Longrightarrow \forall z Pr_y[A(x,y,z)=1] < \frac{1}{4}$$

,其中的Q|y|和Q|z|是多语种在X|x|的边界.

We saw that $AM = AM_1$. We can prove a similar result for MA:-
Theorem: $MA = MA_1$. Proof is left as a homework. ♣

MA is like $N.BPP$ and MA_1 is like $N.coRP$. We have shown above that $N.BPP = N.coRP$. Can we employ similar techniques to show that BPP $\subseteq$ coRP? Unfortunately, we can't. Because we do not allow the intervening of the prover. In the proof that MA $\subseteq$ MA_1 we allowed the prover to give the proof and the shifts, so we did have the intervening of the prover.

We now establish a link between MA and AM. We show that $MA \subseteq AM$. Let $L \in MA$ via a polynomial predicate $A(.,.,.)$ that meets the requirements mentioned in the definition of MA. One naive attempt to prove $L \in AM$ would be to use the same predicate A for the AM protocol. This would work, when $x \in L$. As per MA, there is some proof z_0 that works for most of Arthur's coin tosses y. In AM, irrespective of Arthur's coin tosses, Merlin can give this same proof z_0. This would satisfy, $Pr_y[A(x, y, z_0) = 1] \geq 3/4$. But, this idea fails in case $x \notin L$. A satisfies the condition that, for any z, only a small fraction y would work. But, for each y, some z may work. So, Merlin can cheat Arthur when $x \notin L$. Our proof will make use of this naive idea, but, by amplifying sufficiently, the success probability of MA, we avoid the pitfall described above.

Theorem: $MA \subseteq AM$.
Proof: Let $\in MA$. As we know that $MA = MA_1$, $L \in MA_1$. Furthermore, we can amplify the success probability so that, we have a polynomial predicate $A(.,.,.)$ that satifies,

$$x \in L \iff \exists z \left[Pr_y[A(x, y, z) = 1] = 1\right]$$

$$x \notin L \iff \forall z \left[Pr_y[A(x, y, z) = 1] \leq \frac{1}{2^{m+1}}\right]$$

where, $m = |z|$, and m is polynomial in $|x|$.

Now we can use the same predicate A for our AM protocol. The main point is that, when $x \notin L$, for any z, most of the y's do not work. By a counting argument, we see that, for half of the y's, no z works:-

$$Pr_y[\exists z, A(x, y, z) = 1] \leq \sum_{z \in \{0,1\}^m} [Pr_y[A(x, y, z) = 1] \leq \sum_{z \in \{0,1\}^m} \frac{1}{2^{m+1}} = \frac{1}{2}$$

On the other hand, when $x \in L$, there is some z_0, that satisfies, $Pr_y[A(x, y, z_0) = 1] = 1$. This z_0 can serve as a proof for Merlin, for all the coin tosses of Arthur. We have shown that,

$$x \in L \iff Pr_y[\exists z, A(x, y, z) = 1] = 1$$

$$x \notin L \iff Pr_y[\exists z, A(x, y, z) = 1] < \frac{1}{2}$$

2 我们可以通过反复运行来降低被骗的概率,从1/2.到较小的值,A. ♣

7.4 上午多回合

在AM和MA,我们进行了两轮互动,一轮由亚瑟,一轮由梅林. 我们可以概括地包括更多回合的互动。 例如,我们有MAM。 梅林先动,然后亚瑟动,然后是梅林.

定义(MAM):

$$x \in L \Longleftrightarrow \exists z_1 \left[Pr_y[\exists z_2, A(x, y, z_1, z_2)] \geq \frac{3}{4} \right]$$

$$x \in L \Longleftrightarrow \forall z_1 \left[Pr_y[\exists z_2, A(x, y, z_1, z_2)] \geq \frac{3}{4} \right]$$

We can think of MAM in two ways. One way is MAM=N.AM, just like NP, but the verfication procedure is an AM predicate (instead of a polynomial predicate. To get the second way, first think of MA as MA(P). The undelying predicate is a polynomial predicate. Then, MAM is MA(NP). In other words, take the original definition of MA, and replace the phrase, "where A is a polynomial predicate" by "where A is a NP predicate".

As with MA and AM, we can also have a one sided error version of MAM, named MAM_1. We can prove the following theorem, using, essentially, the same technique we used to prove that $AM = AM_1$.

Theorem : $MAM = MAM_1$. ♣

Then, we proceed as in the proof of $MA \subseteq AM$, to show that $MAM \subseteq AMM$. But, what is AMM? This means, that Arthur picks the random string y_1. Then, Merlin gives a proof z_1. Then, again Merlin gives a proof z_2. But, Merlin can combine these two moves into a single move and give $z_1 \cdot z_2$. Thus, AMM is same as AM. In other words, we extend the proof $MA \subseteq AM$ to show that $MAM = MA(NP) \subseteq AM(NP)$. But what is AM(NP)? Observe that it is same as AM. We have proved that:-

Theorem: $MAM = AM$. ♣

We can generalize further to involve more rounds mixing M's and A's. For example, MAAAMMAAAMM. But we can certainly, replace a streak of A's by a single A and a streak of M's by a single M. In the above example, we have MAAAMMAAAMM⊆ MAMAM. Just like MAM is same as AMM, we can make MAMAM to be AMMAM, by changing the starting MA to be AM. Now again compress the "MM" into "M" to get AMAM. We can continue this process of turning "MAM" into "AM" then, compressing streaks of M's into a single M (and streak of A's into single A) to get AM. Define, AM[k] to be AM with k rounds of interaction. We have "shown" that,

Theorem: AM[k]=AM. ♣

正式证明可以通过使用感应法给出. 以上定理只有在 k 是常数时才有效。 如果弹数取决于输入长度呢? 意思,如果

输入是x, 我们允许多(QXXXXXXX) 回合的反射。 将AM[poly]定义为这样的类. 那么,人们知道

Theorem: AM[poly]=PSPACE. ♣

我们证明了

$$MA_1 = MA \subseteq AM_1 = AM = MAM$$

In fact, we showed that, any finite (fixed number of) iterations of M and A is $\subseteq$ AM.

7.5 AM类和其他复杂类

我们现在研究AM与其他古典复杂类的关系。 记得,据说L在...
2,如果存在多号上游D(.,.),则

$$x \in L \Longleftrightarrow \exists y \forall z[D(x, y, z) = 1]$$

在那里 和 \ 和 \ \ \ \ \ 和 \ \ \ \ \ \ 和 \ \ \ \ \ \ 和 \ \ \ \ \ 和 \ \ \ \ 和 \ \ \ 和 \ \ \ \ 和 \ \ \ \ \ 和 \ \
\ \ 和 \ \ \ 和 \ \ \ \ 和 \ \ \ 和 \ \ \ 和 \ \ \ \ 和 \ \ \ 和 \ \ \ \ 和 \ \ 和 \ 属于 \ \ \ \ \ \ \ \ 的 \ 和 \ \ 的 \ 的。 据说L语言在QQP

2,如果它的赞美L在ΣP

if there is a polynomial predicate A such that, Π_2^P. L is in Π_2^P

我们知道,AM包含在QQP . This is easy to see that $\mathrm{AM}_1 \in \Pi_2^P$

$$x \in L \Longleftrightarrow \forall y \exists z[A(x, y, z) = 0]$$

$$x \notin L \Longleftrightarrow \exists y \forall z[A(x, y, z) = 1]$$

Verify that $\mathrm{AM}_1 \subseteq \Pi_2^P$. If $x \in L$, for all coin tosses y of Arthur, Merlin has some proof z. Whereas, if $x \notin L$, we have some y (in fact, most of the y), for which, all proofs of Merlin are invalid. As $\mathrm{AM}_1 = \mathrm{AM}$, we have proved the theorem,

Theorem: $\mathrm{AM} \subseteq \Pi_2^p$ ♣

We know that $NP \subseteq$ AM. We also know that graph non-isomorphism is in AM. Can we generalize this to prove that, $coNP \subseteq$ AM? But it is unlikely, as shown by the following theorem.

Theorem: If $coNP \subseteq$ AM, the polynomial hierarchy (PH) collapses to AM. $PH = AM$. ♣

Proof: The proof is a simple application of our theorem that $MAM = \mathrm{AM}$. From the definition of Σ_2^P, we have that $\Sigma_2^P = \mathrm{NP}^{\mathrm{AM}}$. As $\mathrm{AM} \subseteq \Pi_2^P$, we have that $\Sigma_2^P \subseteq \Pi_2^P$. Thus, $\Sigma_2^P \subseteq \Pi_2^P$. We now prove that if $\Sigma_2^P \subseteq \Pi_2^P$ then $PH = \Sigma_2^P$. From the definition of Σ_2^P, we see that $\Sigma_2^P \subseteq \mathrm{AM}$. As $\mathrm{AM} \subseteq \Pi_2^P$, we have that $\Sigma_2^P \subseteq \Pi_2^P$. Thus, $\Sigma_2^P = \Pi_2^P$. We now provide an idea of how the entire hierarchy collapses to this level. We already have that $\Sigma_2^P = \Pi_2^P$. We can show that $\Sigma_3^P = \mathrm{NP}^{\Sigma_2^P} = \mathrm{NP}^{\Pi_2^P} = \Sigma_2^P$. Thus, $\Sigma_3^P = \Sigma_2^P$. In general, we can show that $\Sigma_k^P = \Sigma_2^P$ for all $k \geq 2$. Thus, $PH = \Sigma_2^P = \mathrm{AM}$.

Theorem: $MA \subseteq ZPP^{NP}$.

Proof: Let $L \in MA$. There exists a polynomial predicate D, such that,

$$x \in L \iff \exists y \left[Pr_z[D(x,y,z) = 1] \right] = 1$$

$$x \notin L \iff \forall y \left[Pr_z[D(x,y,z) = 1] \right] \leq \frac{1}{m}$$

where, $|y| = |z| = m$, and m is a polynomial in $n = |x|$. Such an amplification can easily be achieved.

Recall that, a set $E \subseteq \{0,1\}^m$ is called thin, if $|E| < 2^m/m$. E is called fat, if E^c is thin. That is, $|E| \geq (1 - 1/m)2^m$. Call E to be perfect if E consists of all strings in $\{0,1\}^m$.

Recall our proof of $BPP \subseteq ZPP^NP$. There given input x, we tested, whether the witness set E_x, is fat or thin, in a zero error manner, using a SAT oracle. We are going to do the same, with little more sophistication.

Fix an input x. We say that y beats z if $D(x,y,z) = 1$. For a y, let E_y be the set of z's that y beats. The main observation is that, if $x \in L$, there there is some y, such that, E_y is perfect. If $x \notin L$, then for all y, E_y is thin. These are the two conditions that our ZPP machine will test (with the help of an oracle.)

我们首先要找出 是否有一个Y和完美的Ey。 我们随机选择一些多名齐。 那问问oracle,有没有人能打败这些家伙 如果答案是“不”,我们可以安全地拒绝x。 如果答案是「是」, 然后我们用自我减压 来找到打败这些Zi的y0 我们问oracle, 如果是这样,我们可以安全接受x。 如果不是,我们接下来就试试看Ey是胖还是瘦. 做这个,我们随便选个弦S1 . . .. 斯马。 并询问oracle,如果Ey0被这些字符串转移时,会覆盖所有z的. 验证这是 NP 问题。 如果Ey0是胖子, 另一方面,如果Ey0是薄的,概率为0. 我们还知道,如果x不在L, 对于所有y0,Ey0是薄的。 因此,如果我们从oracle那里得到“是”的答案,我们就能够安全地接受。 如果我们得到“不”的答案,我们就有麻烦了。 有三个案件是可能的:

- Case 1: $x \in L$, but the oracle gave us a y_0 with a thin E_{y_0};
- Case 2: $x \in L$, the oracle gave us a a good y_0, but our shift strings were not good enough to cover (a low probability event).
- Case 3: $x \notin L$, and so the y_0 is thin.

因此,如果我们得到“不”的答案,我们就放弃。 我们认为,第2号案例是一个低概率事件。 例1和例2只有在有薄的y0,才能击败我们的随机字符串{zi}时才会发生. 但是Ey0瘦了,你怎么能打败这么多Zi? 我们稍后正式提出这个论点。

形式上,ZPP机M,给定SAToracle的工作如下: - 让x成为输入字符串.

1. 国家 选择 m2 字符串 z1。 . . .. zm2 随机从{0, 1} m.

2. Ask the oracle if there is a y such that for all i, $D(x, y, z) = 1$. If the answer is 'no', then REJECT. If 'yes', use self-reducibility to get a y_0 that beats all these z_i.

3 (中文(简体)). 选择 m 字符串 s1,。 . . .. sm, 随机从{0, 1} m.

4 , 4 , 4 , 4 , 4 , 4 , 4 问oracle, 当Ey0被这些随机字符串移动时, 是否会覆盖所有 z。 也就是说,问下一个前提是否真实

$$\forall z \left[z \in \bigcup_{i=1}^{m} E_{y_0} \oplus s_i \right]$$

如果上面的上游是真实的,那么 Accept. 如果没有,产出....."?"

Correctness: In step 2, when we reject, we have that for any y, it fails to beat at least one z_i. So, there cannot be any perfect y. We are correct in rejecting x. In step 4, suppose we ACCEPT. We claim that we are again correct in accepting x. By contradiction, suppose $x \notin L$. Then every y is bad. For a bad y, $E_y < 2^m/m$. So shifting by m strings cannot cover all of $\{0,1\}^m$.

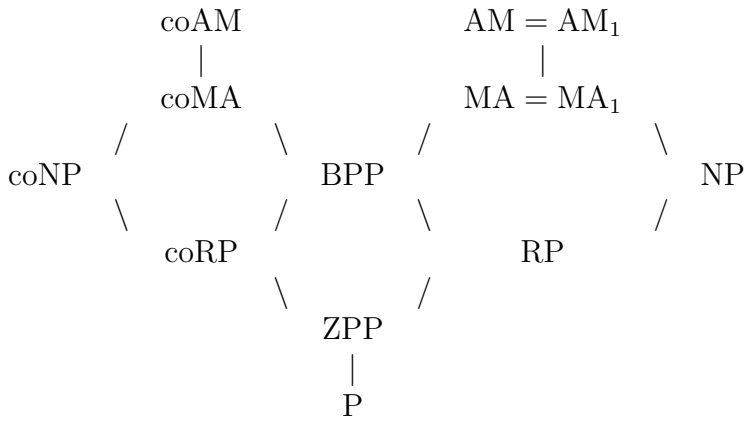
Now we argue that the probability that we output "?" is low. First let's bound the probability that we will have a non-good y_0 at the end of step 2. Fix a y_0 which is not good. That is $Pr_z[y_0 \text{ beats } z] \leq (1 - 1/m)$. We picked up m^2 strings at random. The probability that y_0 beats all these strings is $\leq (1 - 1/m)^{m^2} \sim e^{-m}$. There are at most 2^m such y's. So, the probability that there exists some y which is not good, but beats all these m^2 strings is $\leq 2^m/e^m$. This probability is exponentially small. So, with high probability, there are no non-good witnesses that can beat all the m^2 strings. (So, if $x \notin L$, we would stop at step 2, with high probability).

Assume that we proceeded to step 4, and y_0 is good. Then, with high probability, the shifting by m strings would cover all of $\{0,1\}^m$. This we proved already in proving that $BPP \in ZPP^{NP}$. Recall the lemma that if $S \in \{0,1\}^m$ has cardinality at least $2^m(1-1/m)$, then,

$$Pr_{s_1 \ldots s_m \in \{0,1\}^m} \left[\bigcup_{i=1}^{m} S \oplus s_i = \{0,1\}^m \right] \geq 1 - \frac{1}{2^{m(k-1)}}$$

- 哪里,K log m。 - 哪里,K log m。 因此,我们输出的概率"?" 以(2m/em)+(2−(k−1))为界。 ♣

复杂类的结构,到目前为止是: /* 与ZPPNP比较好 成为DRAWN */



7.6 LFKN永久议定书

We start the discussion by reviewing the definition of permanent of a matrix. Let $M = (a_{ij})$ be a $n \times n$ matrix where each entry is n bits long. We define,

$$\mathrm{Perm}(M) = \sum_{\sigma=(\sigma_1,\ldots,\sigma_n)\in S_n} a_{1\sigma_1} \cdots a_{n\sigma_n}$$

其中 Sn 是设定 {1 } 上所有排列的一组。 . . .. n.

还有其他方式来表达永久。 这和我们通常计算确定者的方式相似,

$$\mathrm{Perm}(M) = \sum_{i=1}^{n} \text{a1i – 珀尔姆(M1i)}$$

中,M1i是(n − 1) × (n − 1)通过删除第1行和一栏获得的子矩阵。 这两个定义的等同性证明是,每个术语在两个表达式中都完全出现一次.

We are now ready to discuss the interactive proof system, The input is a $n \times n$ matrix M, with each entry n-bits. Since permanent has $n!$ terms, $|\mathrm{Perm}M| \leq 2^{n^{O(1)}}$. The prover P first chooses a huge prime $p > 2^{n^{O(1)}}$ and sends it to the verifier V. Using the recent polynomial time AKS algorithm for primality testing, the verifier can check that p is indeed a prime number. The prover next sends the $\mathrm{Perm}(M)$ modulo p. As p is large enough, taking modulo p is fine. Hereafter, all the actions take place modulo p.

So the prover has produced a number α with the claim that $\mathrm{Perm}(M) = \alpha \bmod p$. If prover's claim is true, we want the verifier to get convinced with probability 1. If not, the verifier has to be convinced with only a small probability.

协议围绕以下多诺米语进行. 定义

$$F(x) = \sum_{i=1}^{n} \frac{(x-1)(x-2)\cdots\widehat{(x-i)}\cdots(x-n)}{(i-1)(i-2)\cdots\widehat{(i-i)}\cdots(i-n)} M_{1i}.$$

以上注解是指在总和的直线词中,(x－i)在数字中缺失,(i－i)在分母中缺失. 聚氮化物的基质为(n－1)×(n－1)基质。 多元体有以下重要财产.

If we set $x = i$, of the n terms in the summation, except the i^{tn} term, all the other terms evaluate to 0. The ith term evaluates to M_{1i}. Thus we have $F(1) = M_{11}$, and in general, for $1 \leq i \leq n$, $F(i) = M_{1i}$. $F(x)$ has been constructed using the matrix analog of the well-known Lagrange interpolation The polynomial can be viewed in two ways. It is a polynomial whose coefficients are $(n-1) \times (n-1)$ matrices. It is also a $(n-1) \times (n-1)$ matrix whose entries are polynomials (in x) of degree $n-1$. We will view $F(x)$ mainly using the second method.

Think of $F(x)$ as a $(n-1) \times (n-1)$ matrix whose entries are polynomials of degree $(n-1)$. What is the permanent of $F(x)$? It is a polynomial of degree $\leq (n-1)^2$. Denote this polynomial as $g(x) = \text{Perm}(F(x))$. The polynomial $g(x)$ has many useful properties. First of all it can used to calculate the permanent of the minors M_{1i} easily:

$$\text{Perm}[M_{1i}] = \text{Perm}[F(i)] = Perm[F(x)]|_{x=i} = g(i).$$

第二,我们可以计算M的永久位置:

$$\text{Perm}[M] = \sum_{i=1}^{n} a_{1i} \cdot \text{Perm}[M_{1i}] = \sum_{i=1}^{n} \text{a1i} \cdot \text{g(i).}$$

The beauty of the polynomial $g(x)$ is that it can do more! We saw that, for any $1 \leq i \leq n$, $g(i) = \text{Perm}(M_{1i})$. But, in fact, $g(x)$ can be used to compute the permanent of many other matrices! Choose any $r \in Z/p$. Evaluate F at r to obtain the $(n-1) \times (n-1)$ matrix $N_r = F(r)$. If $1 \leq r \leq n$, N_r is nothing but M_{ir} and for other values, N_r is some matrix! The main idea of the proof is that $g(r) = \text{Perm}(N_r)$, for any $r \in Z/p$. Thus g can be used to compute permanent of some p many matrices (and $p >> n$). Still g is very short: it is only polynomially long in n. Now let us get back to the discussion of the interactive proof system and put our observations into use.

So the prover has sent a value α with claim that $\alpha = \text{Perm}(M)$. Next he sends a polynomial $g'(x)$, with the implicit claim that $g'(x) = g(x)$ (of couse, he could be lying!). The verifier first checks whether these two claims are consistent: check if

$$\alpha = \sum_{i=1}^{n} \text{a1i} \text{——} \text{g} \atop \text{' (—).}$$

如果上述测试失败,核查人将停止并驳回证明人的索赔。 假设以上测试通过.

我们可以做一个简单的观察:

$$\alpha \neq \text{Perm}(M) \Longrightarrow g \neq g'.$$

Even if the prover is a liar, he has to be consistent with his lies! If he lies and gives $\alpha \neq \text{Perm}[M]$, he has to lie again and give a g' different from g. Otherwise, the above consistency check will fail and verifier will catch him!

The next step is the heart of the proof. The verifier computes the polynomial $F(x)$. He can compute $F(x)$ easily, without the help of the prover. Next the verifier chooses a random number r from Z/p and computes $N_r = F(r)$. As we saw already, permanent of N_r can be computed

$$\text{Perm}(N_r) = \text{Perm}[F(r)] = g(r).$$

Of course, the verifier cannot compute the polynomial g. But, he has the g' given by the prover, who claimed that $g' = g$. So the verifier simply computes $g'(r)$ and assumes that $g'(r) = \text{Perm}(N_r)$.

There are two possible cases to analyze: $g' = g$ and $g' \neq g$. If $g' = g$, then clearly, $g'(r) = \text{Perm}(N_r)$. On the other hand, suppose $g' \neq g$. What is the probability (over the random choice of r) that $g'(r) = \text{Perm}(N_r)$? We know that g and g' are both polynomials of degree $(n-1)^2$. So they can agree on at most $(n-1)^2$ many choices of $x \in Z/p$. As r was a random value chosen from Z/p,

$$\Pr_{r \in Z/p}[g'(r) = g(r)] \leq \frac{(n-1)^2}{p}.$$

We chose p to be exponential in n. So except for a (exponentially) small probability, if $g' \neq g$, then $g'(r) \neq \text{Perm}(N_r)$. Let us summarize the discussion so far: if $\alpha \neq Perm[M]$, then $g' \neq g$ and hence, with probability exponentially close to 1, $g'(r) \neq \text{Perm}(N_r)$. But, by claiming that $g = g'$, the prover has, implicitely, claimed that $\text{Perm}(N_r) = g'(r)$.

The situation the same as the one with which the story started, but it is "smaller". Initially, the provr claimed that $\alpha = \text{Perm}(M)$, where M was a $n \times n$ matrix. Now, he has claimed that $g'(r) = \text{Perm}(N_r)$, where N is a $(n-1) \times (n-1)$ matrix. And if the prover' first claim was a lie, then the second claim is also a lie, with very high probability! How can the verifier validate the claim? Simply, by repeating the same process!

After $(n-1)$ rounds, the two players would be left with a matix of size 1×1, say (x), and the prover would have (implicitely) claimed its permanent to be some β. The verifier can esily check this: Is $\text{Perm}[(x)] = x = \beta$? If not he rejects the original claim of the prover that $\alpha = Perm(M)$. If $\alpha \neq \text{Perm}(M)$, then probability that $\beta \neq \text{Perm}[(x)]$ is at least $1 - n \cdot (n-1)^2/p$. (The factor n comes from the n rounds). We conclude that a honest prover will win with probability 1. And a dishonest prover will win with probability exponentially small.

Exact protocol: M is the $n \times n$ input matrix. The prover claims that $Perm(M) = \alpha$. The first thing prover does is to choose a prime number $p \sim 2^{n^3}$, (note: $p >> Perm(M)$). He sends it to the prover along with a proof that it's prime (using Pratt's theorem that Primes$\in$ NP). All the computation hence forth take place modulo p.

1. Prover's claim is that $Perm(M) = \alpha$, where $M = (a_{ij})$ is a $n \times n$ matrix.
2. If $M = (x)$ is a 1×1 matrix, the verifier checks $\alpha = x$. If they do not match he *rejects* else *accepts*.

3 (中文(简体)). 核查员计算

$$F(x) = \sum_{i=1}^{n} \frac{(x-1)(x-2)\cdots\widehat{(x-i)}\cdots(x-n)}{(i-1)(i-2)\cdots\widehat{(i-i)}\cdots(i-n)} M_{1i}.$$

4. Prover sends a polynomial $g'(x)$ degree $(n-1)^2$ (with the implicite claim that $g'(x) = \mathrm{Perm}[F(x)]$).

5 国 务 院 验证检查

$$\alpha = \sum_{i=1}^{n} \text{a1i} \text{——} \text{g}$$

' (一).

如果他们不相容,

6. Verifier picks $r \in_R Z/p$ and sends it to prover.

7. Verifier computes $N_r = F(r)$, a $(n-1) \times (n-1)$ matrix.

8. 与《公约》有关的其他事项 校验器设置为 M ←−Nr, n ←−n − 1和
α ←−g ' (r).

9. 国家 回到第1步去 继续

If the $\alpha = Perm(M)$, the prover can provide correct polynomials g and the verifier will accept with probability 1. Suppose $\alpha \neq \mathrm{Perm}(M)$. The verifier will reject in step 4 of some round. If that does not happen, to make verifier accept in the end in step 1, the verifier has to be lucky in at least one round, where the random r satisfies $g'(r) = g(r)$. Probability for this to happen in any particular round is $\leq (n-1)^2/p$. So probability for it to happen in one of the n rounds is $\leq n(n-1)^2/p$. As $p \sim 2^{n^3}$, this error probability is exponentially small. ♣

第8章

IP=PSAPCE

第9章

去随机化

章节大纲:德兰多姆化演绎. 普苏多随机生成器. 下位预测和伪随机等效. 单向功能. 弱单向意味着强单向. 戈德里奇·莱文硬核位。 使用 GL 硬核位构造伪随机生成器。

需要在这里实现一个理想化。

9.1 Pseudorandom 发电机

非正式地说,假随机生成器G是一种多诺时算法,它将一个叫做种子的短随机字符串伸展为长假随机字符串. G是成功的,如果它的输出不能与任何有效的对手真正随机的弦区分开来. 换句话说,G的输出在计算上与真正的随机字符串是无法区分的. 我们正式确定两种模型中的不可区分性概念,并显示它们的等同性。

生成器是多诺时可计算,决定函数G:{0, 1}n → {0, 1}l(n) 的l(n) > n. 敌方是概率论的多名时间Turing machine.

Pseudorandom Generators:我们把形式主义的第一个模型称为伪随机模型. 在这方面,我们通过下列公约。 一个对手**A,**给定一个输入字符串,如果认为其输入是伪随机的,输出**1,**如果认为其输入是真正随机的,输出**1.** 关于生成器**G,**我们把对手**A**的成功定义为:

$$\delta_n^A = |\Pr_{x\in\{0,1\}^n}[A(G(x)) = 1] - \Pr_{y\in\{0,1\}^{l(n)}}[A(y) = 1]|$$

如果对手的区别良好,则n的大小更大。

Definition (Pseudorandom Generator): We say that G beats A if $\forall$ polynomials $p(\cdot)$, $\forall$ sufficiently large n, we have δn < 1 p(n). G被定义为假随机生成器,如果 G

打败每个对手

说明:根据上述定义,如果存在p(...),则A击G,这样,我们拥有的N(A)就无穷无尽。

$p(\cdot)$

下一个比特不可预测性:在这个第二个模型中,对手A先得到生成器输出的I比特. 它必须预测(i + 1)第1位. 对于对手A来说,从长远来看,成功与否是衡量:

$$\beta_n^A = \Pr_{\substack{x\sim U_n \\ i\in_R[0..l(n)-1]}}[A(G(x)_{1,\ldots,i}) = G(x)_{i+1}] - \frac{1}{2}$$

因此,我们平均占所有i。 任何对手都很容易通过猜测获得1/2的成功. 因此,我们已经减去1/2,以恢复正常。

Definition (Next-bit unpredictable generator): We say that G beats A if $\forall p(\cdot)$, $\forall$ sufficiently large n, we have $\beta n^A < \frac{1}{p(n)}$. G is defined to be a next-bit unpredictable if G beats every adversary.

第一种型号应归姚,第二种型号应归布卢姆和米卡利. 姚氏所显示的等同性.

定理(Yao):G是假冒的,如果并且只有在G是下位可预见的情况下. 证明:等效地,G如果且只有G不是下位可预选时,才不是假冒. 因此,我们显示,一些对手在假冒模型中击败G, 如果并且只有在一些对手在下一个位模型中击败G。

第一部分 假设A是下一个模式中的对手 我们想要找到一个对手A '

在假冒模型中表现同样出色

修无. A ' ' 以 l(n) 位字符串作为输入,并必须预测其是否真正随机或伪随机. A ' 把它当作oracle。 A ' on introduction y1, (英语). . . .. yl(n),有下列规定:

1. First choose a random $i \in [0..l(n)-1]$.
2. If $A(y_1, \ldots, y_i) = y_{i+1}$ then output 1, i.e., guess pseudorandom. Otherwise output 0, i.e., guess truly random.

假设A的成功,在长度n是βA .

$$Pr_{x\sim U_n}[A'(G(x)) = 1] = Pr_{\substack{x\sim U_n \\ i\in_R[0..l(n)-1]}}[A(G(x)_{1,\ldots,i}) = G(x)_{i+1}] = \beta_n^A + \frac{1}{2}$$

如果输入A ' 的输入是真正随机的,无论A ' 选择的i,A可以输出yi + 1,概率只有1/2. 这么说

$$Pr_{\quad\quad}[A(y) = 1] = \frac{1}{}$$

2 因此,

$$\delta_n^{A'} = Pr_{x\in\{0,1\}^n}[A(G(x)) = 1] - Pr_{y\in\{0,1\}^{l(n)}}[A(y) = 1] = \beta_n^A + \frac{1}{2} - \frac{1}{2} = \beta_n^A$$

编号因此,如果A能在下位模型中击败G,A'就可以在伪随机模型中击败G.

第二部分:现在假设A在假冒随机模型中是G的对手,并取得了成功 A
长度为n. 在下一个模式中,我们构建一个对手A'。

修无. 让A ' b1输入。 . . .. ,双。 A'必须预测下一个比特比 + 1. A ' 做下列事情: -

1. 国家 选择 l(n) -i 随机位数 ri + 1。 . . .. ,rl(n) 随机。
2. Compute $a = A(b_1, \ldots, b_i, r_{i+1}, \ldots, r_{l(n)})$.
3. If $a = 1$ output r_{i+1}, else output $\overline{r_{i+1}}$.

The intuition is that A outputs 1 more often when given all pseudorandom bits. And outputs 0 more often when given all truly random bits. We input A with a mix of i pseudorandom bits and $(l(n) - i)$ truly random bits. b_{i+1} is either r_{i+1} or its compliment. If $r_{i+1} = b_{i+1}$ then A gets one more pseudorandom random bit and is more likely to output 1. To put it in a reverse Bayesian way, if A outputs 1, we are more likely to have $r_{i+1} = b_{i+1}$. To get a formal proof, we first quantify the behavior of A when given a mix of pseudorandom and truly random bits. Suppose we input A with i pseudorandom bits and $l(n) - i$ truly random bits. The probability that A outputs 1, for such a mix, is denoted by p_i. Formally,

$$p_i = Pr_{\substack{x \sim U_n \\ r_{i+1}, \ldots, r_{l(n)} \in_R \{0,1\}}} [A(G(x)_{1,\ldots,i} \circ r_{i+1}, \ldots, r_{l(n)}) = 1$$

首先,我们发现,

$$p_0 = Pr_{y \in_R \{0,1\}^{l(n)}}[A(y) = 1]$$
$$p_{l(n)} = Pr_{x \in_R \{0,1\}^n}[A(G(x)) = 1]$$

Thus, $p_{l(n)} - p_0 = \delta_n^A$.

接下来我们用"A"来计算A的成功 . For any $0 \leq i < l(n)$, we denote the success of A' as $\beta_{n,i}^{A'}$. We let $w = b_1, \ldots b_i, r_{i+1}, \ldots, r_{l(n)}$. Then,

$$\begin{aligned}
\beta_{n,i}^{A'} &= Pr[A'(b_1, \ldots b_i) = b_{i+1}] \\
&= Pr[A(w) = 1 \text{ and } r_{i+1} = b_{i+1}] + Pr[A(w) = 0 \text{ and } r_{i+1} \neq b_{i+1}] \\
&= Pr[r_{i+1} = b_{i+1}] \cdot Pr[A(w) = 1 | r_{i+1} = b_{i+1}] + Pr[r_{i+1} \neq b_{i+1} \cdot Pr[A(w) = 0 | r_{i+1} \neq b_{i+1}] \\
&= \frac{1}{2}\left(Pr[A(w) = 1 | r_{i+1} = b_{i+1}] + Pr[A(w) = 0 | r_{i+1} \neq b_{i+1}]\right)
\end{aligned}$$

The first quantity inside the parenthesis, is easy to compute. When $r_{i+1} = b_{i+1}$, we are giving one more pseudorandom bit to A. So,

$$Pr[A(w) = 1 | r_{i+1} = b_{i+1}] = p_{i+1}$$

. . .. 计算第二个数量:-

$$\begin{aligned} p_i &= Pr[A(b_1,\ldots,b_i,r_{i+1},\ldots,r_{l(n)})=1] \\ &= Pr[r_{i+1}=b_{i+1}]\times Pr[A(w)=1|r_{i+1}=b_{i+1}]+Pr[r_{i+1}\neq b_{i+1}\times Pr[A(w)=1|r_{i+1}\neq b_{i+1}] \\ &= \frac{1}{2}p_{i+1}+\frac{1}{2}Pr[A(w)=1|r_{i+1}\neq b_{i+1}] \end{aligned}$$

So,

$$Pr[A(w)=0|r_{i+1}\neq b_{i+1}]=p_{i+1}-2p_i+1$$

替换βA的方程式 不,我,我们得到,

$$\beta_{n,i}^{A'}=\frac{1}{2}\left(p_{i+1}+p_{i+1}-2p_i+1\right)=\frac{1}{2}+p_{i+1}-p_i$$

现在,我们可以计算成功 βA 随机选择的

$$\begin{aligned} \beta_n^{A'} &= \text{编号(n)X}\ \frac{1}{l(n)}\ \text{l(n)βA'号} \\ &= \text{编号(n)X}\ \frac{1}{l(n)}\left(\frac{1}{2}+\text{pi}+1-\text{pi键}\right) \\ &= \frac{1}{2}+\frac{1}{l(n)}\ \text{编号(n)X}\ \text{pi}+1-\text{pi键} \\ &= \frac{1}{2}+\frac{1}{l(n)}(p_n-p_0) \\ &= \frac{1}{2}+\frac{\delta_n^A}{l(n)} \end{aligned}$$

A ' 的成功概率与A的几率是多极性的。 因此,如果A在假冒模型中击败G,A'在下位模型中击败G. ♣

9.2 单向函数

在本次讲座中,我们定义了弱和强的单向函数. 非正式地,一个函数f是单向的,如果它很容易计算给x的f(x),但很难计算给x的f(x). 弱的单向概念要求任何试图反转f的概率性多诺时算法至少对相当一部分输入失败. 强单向的概念要求任何概率性多诺时算法都未能反转f,只有一小部分输入除外. 我们使这些概念精确,然后表明,单向功能薄弱的存在意味着存在强大的单向功能.

定义(微弱的单向函数): 一个函数 f :{0, 1}n\\\0, 1}p(n) 叫做弱单向函数, 如果存在 l > 0, 则所有概率多时算法 A, 对于足够大的 n, A 未能在输入的至少1nl 的一小部分中倒置 f。 形式上

$$Pr_{x\sim U_n}\left[f(A(f(x))\neq f(x)\right]\geq\frac{1}{n^l}$$

定义(强单函数):函数 f :{0, 1}n →{0, 1}p(n) ,称为强单函数,如果对所有概率多诺时算法A, 对于所有多诺时算法p, 对于足够大的n,

$$Pr_{x\sim U_n}\left[f(A(f(x))=f(x)\right]\leq\frac{1}{p(n)}$$

换句话说,A只能将输入的微小部分反转f.

定理(Yao):如果存在薄弱的单向函数,则存在强大的单向函数. 证明:假设 f : {0, 1}n {0, 1}p(n) 是一个弱的单向函数. 我们构建一个强大的单向函数 F : {0, 1} mxxn {0, 1} mxp(n)。 m是一个参数 polynomial in n, 我们稍后会修正它。 F的定义是:

$$F(x_1,x_2\ldots x_m)=<f(x_1),f(x_2),\ldots f(x_m)>$$

索赔是F强烈的单向。 直觉如下:- 假设一个算法试图以天真的方式颠倒F,通过颠倒其输入的每个组件. 它有成功的可能性

success probability $\leq\left(1-\frac{1}{n^l}\right)^m$ in inverting F. If we choose, $m=n^{l+1}$, this probability will be $\leq e^{-m}$. We next prove the claim formally.

By contradiction, assume that F is not strong one-way. Then there is an algorithm A can invert F on $\geq e^{-n}$ 其投入的nk部分,为一些k. 我们要在 > 1 - 1 上反转 f 我们2n输入的零分之一。

Our algorithm A' to invert f is the following: Upon input $y=f(x)$, first randomly pick $i\in_R\{1,\ldots,m\}$. Then, randomly (uniformly) pick $x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_m\in\{0,1\}^n$. Construct,

$$Y=\langle f(x_1),\ldots,f(x_{i-1}),y,f(x_{i+1}),\ldots,f(y_m)\rangle$$

where $y=f(x)$ is in the ith position. Ask A to invert Y. Let it's output be $X=\langle x_1,\ldots,x_m\rangle$. If $y=f(x_i)$, output x_i. If A' succeeded in inverting F at Y, we would have succeeded in inverting y. We repeat the process some polynomial (to be fixed later) number of steps to enhance our success probability. If we are unsuccessful in all these

(一) 国家 如果A'成功在Y颠倒F,我们就会成功颠倒y。 我们重复这一过程,采取一些多名(稍后将固定)步骤,以提高我们的成功概率。 如果我们在所有这些多方面的审判中不成功,我们就放弃。

我们现在认为,我们的算法在它输入的1毫升以上的一小部分中倒置。 定义一边2n顶和另一边2mn顶的双方图。 {0, 1}n 中的字符串是左侧 L 中的顶点,而 {0, 1}m n 中的字符串是顶点

right side R. Add an edge between x and $(x_1, \ldots, x_m)$ if $x = x_i$, for some i. The graph is a multigraph (meaning, there may be more than one edge between a pair of vertices). For a node x, we let $N(x)$ denote the multi-set of its neighbors. Let $G \subseteq R$ are the points at which A can invert F.

Now let's review our algorithm A'. Given $f(x)$, A' (implicitly) chooses a random element $r \in_R N(x)$, then computes $Y = F(r)$ and then asks A to find r. A' wants to find x=i^{th} component of r. If $N(x) \cap G$ is large, then with high probability Y will be in G. If $Y \in G$, then A' will invert Y successfully and hence A' will invert $f(x)$ successfully. We shall argue that for "large" fraction of x, $N(x) \cap G$ is "large".

To make the above argument formal, let's first state some simple facts. $|L| = 2^n$. $|R| = 2^{m \cdot n}$. The degree of each node in R is m. The degree of each node in L is $m2^{n(m-1)}$. Our bipartite graph is an "expander", in the following sense.

A bipartite graph has (ϵ, δ)-expansion property, if for all $S \subseteq L$, with $|S| \geq \epsilon|L|$, we have that $|N(S)| \geq (1 - \delta)|R|$. In other words, for any small set in L, its neighbor set is large. Our graph has good expansion properties.

Let $S \subseteq L$ be of size $|S| \geq \epsilon|L|$. What's the size of $N(S)$? Pick a random $Y \in R$. Let $Y = (x_1, x_2, \ldots x_m)$. What's the probability that $Y \notin N(S)$? So,

$$Pr[Y \notin N(S)] = Pr[x_1 \notin S, \ldots x_m \notin S] \leq (1 - \epsilon)^m$$

We want this probability to exponentially small. So, choose $m = n/\epsilon$. Then,

$$Pr[Y \notin S] \leq (1 - \epsilon)^m \leq e^{-n}$$

还有

$$|N(S)| \geq (1 - e^{-n})|R|$$

我们现在就把参数修好。 修补:

$$\epsilon_0 = \frac{1}{n^l}; m = \frac{n}{\epsilon_0} = n^{l+1}; \delta_0 = e^{-n}$$

因此,我们的图表有(ε0,δ0)扩展财产。

我们假设对手A在e -n+1中颠倒F ; fraction of the inputs. Let $\alpha = \frac{1}{n^k}$. Then $|G| \geq (\delta_0 + \alpha)|R|$.

Let $S \subseteq L$, be any set with $|S| \geq \epsilon_0|L|$. As our graph has (ϵ_0, δ_0)-expansion property, $|N(S)| \geq (1 - \delta_0)|R|$. So,

$$|G \cap N(S)| = |G| - |G \cap \overline{N(S)}| \geq |G| - |\overline{N(S)}| \geq (\alpha + \delta_0)|R| - \delta_0|R| = \alpha|R|$$

So $|G \cap N(S)| \geq \alpha|R|$. The left degree of the graph is $m2^{n(m-1)}$. So, the total number of edges incident on S is $|S|m2^{n(m-1)}$. How many of these are good? Meaning, how many have the other vertex in G? From each one of these vertices in $|G \cap N(S)|$, at least one edge goes

改为S. 因此,在S上的好边缘事件数量至少是QQR|. S的好边缘事件平均有多少?

$$\frac{\#\text{good edges}}{\text{total \# of edges}} \geq \frac{\alpha|R|}{Sm2^{n(m-1)}} = \frac{\alpha 2^{n\cdot m}}{Sm2^{n(m-1)}} \geq \frac{\alpha}{m}\frac{2^{nm}}{2^n 2^{n(m-1)}} = \frac{\alpha}{m}$$

所以,对于任何大小至少为%=0=L的S集,分数好的边缘大于α. .

现在想想这一套

$$S = \left\{ x \in L \;\middle|\; \frac{|\{(x,g) \mid g \in G\}|}{|N(x)|} < \frac{\alpha}{m} \right\}$$

对于此集,好边缘的分数为 $< \alpha$ 说吧 因此它的大小应该是 $< \varepsilon 0|L|$。 所以,这个集

$$D = \left\{ x \in L \;\middle|\; \frac{|\{(x,g) \mid g \in G\}|}{\deg(x) = |N(x)|} \geq \frac{\alpha}{m} \right\}$$

has cardinality $|D| \geq (1-\epsilon_0)|L|$.

We are given $f(x)$ as input to invert. We randomly pick a neighbor r of x, compute m,r将是在G,我们将成功。 如果我们重复这个过程 m

main point is that α was assumed to be $1/n^k$, a inverse polynomial. So m/α is polynomial. As $|D| \geq (1-\epsilon_0)|R|$, $Pr_x[x \in D] \geq (1-\epsilon_0) = 1 - \frac{1}{n^l}$. Thus, we will invert f with probability $\geq 1 -$ 没有。 这与f为1/nl-hard的假设相矛盾. ♣

9.3 戈德里奇-勒文硬核比特

让我们做个强势的单向运动 f的硬核上游是一个布尔函数 g, 从而很难计算 g(X, Y), 给 f(X) 和 Y, 形式上, 对于任何概率多诺时算法, 对于所有多诺时算p, 对于足够大的n,

$$Pr_{x,y \in_R \{0,1\}^n}[A(f(x), y) = g(x,y)] \leq \frac{1}{2} + \frac{1}{p(n)}$$

We show that such predicates can be constructed for any strong one-way permutation f. The construction is due to Goldreich and Levin. Define $g(X,Y)$ to be the inner product $X \cdot Y$ of X and Y. That is, if $X = x_1 x_2 \ldots x_n$ and $Y = y_1 y_2 \ldots y_n$, then, $X \cdot Y = \sum x_i y_i$, where the summation is over mod 2.

我们声称g是f的硬核上游. 相反,假设有一个算法A可以计算g(X,Y),给定f(X)和Y,概率1 2+1 纳克. 我们显示,使用A作为黑匣子,f可以被高概率反转.

For two string $X = x_1 \ldots x_n$ and $Y = y_1 \ldots y_n$, we define $X + Y$ to be the bit-wise XOR of X and Y. That is, $X + Y = z_1 \ldots z_n$, where $z_i = x_i + y_i \pmod 2$. Define $\widehat{X} = \bar{x_1} x_2 \ldots x_n$,

the string X with first bit flipped. We have $X \cdot (Y+Z) = X \cdot Y + X \cdot Z$. So, $X \cdot Y + X \cdot \widehat{Y} = x_1$. This last property plays an important role in our construction. Our goal is to compute x_1 correctly, with high probability. Then use the same procedure to compute other bits of X, there by inverting f.

Let $m = O(\log n)$. We will fix its exact value later. Choose m strings $Y_1, Y_2, \ldots Y_m \in \{0,1\}^n$ at random. Let $\epsilon_1, \epsilon_2, \ldots, \epsilon_m$ be some fixed m bits, not all 0. Define $Y_{\epsilon_1\epsilon_2\ldots\epsilon_m} = \sum \epsilon_i Y_i$. The above one is a random variable. It will be uniformly distributed over $\{0,1\}^n$. Take any $\delta_1\delta_2\ldots\delta_m \in \{0,1\}^m - \{0\}^m$, such that $\epsilon_1\ldots\epsilon_m \neq \delta_1\ldots\delta_m$. For some i, $\epsilon_i = 1$ and $\delta_i = 0$ (or vice versa). Y_i is chosen uniformly from $\{0,1\}^m$ and it contributes only to $Y_{\epsilon_1\ldots\epsilon_m}$. So the random variables $Y_{\epsilon_1\ldots\epsilon_m}$ and $Y_{\delta_1\ldots\delta_m}$ are independent. Thus the set of $2^m - 1$ random variables $\{Y_{\epsilon_1\ldots\epsilon_m}\}$, for $\epsilon_1\ldots\epsilon_m \in \{0,1\}^m - \{0\}^m$, are pairwise independent.

Now we are ready to present our construction. We first choose $Y_1, Y_2, \ldots, Y_m \in_R \{0,1\}^n$. Then guess the values of $X \cdot Y_1, X \cdot Y_2, \ldots, X \cdot Y_m$ to be $b_1, b_2, \ldots, b_m$. Assume that we guessed correctly! Then, for any $\epsilon_1, \ldots, \epsilon_m$ we can compute $X \cdot Y_{\epsilon_1\epsilon_2\ldots\epsilon_m}$ correctly:-

$$X \cdot Y_{\epsilon_1\epsilon_2\ldots\epsilon_m} = X \cdot (\epsilon_1 Y_1 + \cdots + \epsilon_m Y_m) = \sum \epsilon_i (X \cdot Y_i) = \sum \text{齐比}$$

Fix some $\epsilon_1, \ldots, \epsilon_m$ and compute $X \cdot Y_{\epsilon_1\epsilon_2\ldots\epsilon_m}$ as above. Then use the black-box A to compute $X \cdot \widehat{Y_{\epsilon_1\epsilon_2\ldots\epsilon_m}}$. Assume that A computed this correctly. We can now compute the first bit x_1 of X, $x_1 = X \cdot Y_{\epsilon_1\epsilon_2\ldots\epsilon_m} + X \cdot \widehat{Y_{\epsilon_1\epsilon_2\ldots\epsilon_m}}$. (Recall that for any Y, $x_1 = X \cdot Y + X \cdot \widehat{Y}$.) How to make the correct guesses for $b_1, \ldots, b_m$? As $m = O(\log n)$, total number of possible values for $b_1 \ldots b_m$ is polynomial in n. So we can run the above procedure for every possible sequence of guesses. And one of the guesses would be correct. To make the black-box work well with high probability, we run the above procedure over all possible (polynomial number of) $\epsilon_1, \ldots \epsilon_m$. Then, we take the majority. We would get x_1 this way. Similarly, compute other bits. Using Chebyshev inequality, we shall show that with high probability (over the random choices for $Y_1, Y_2, \ldots, Y_m$), we would have got the correct $x_1, x_2, \ldots, x_m$. We can verify whether we are correct by checking $f(x_1 x_2 \ldots x_m) = f(X)$. If we are not correct, we run the experiment again, by choosing at random another set of $Y_1, Y_2, \ldots, Y_m$.

接下来我们把算法全部呈现出来 输入为f(X). 它试图计算X。

Choose $Y_1, Y_2, \ldots, Y_m \in_R \{0,1\}^n$.
For all 2^m choices of $b_1 \ldots b_m \in \{0,1\}^m$ do:-
    For $1 \leq k \leq n$ do:-
        For each $\epsilon_1 \ldots \epsilon_m \in \{0,1\}^m - \{0\}^m$ do:-
            使用 A 来计算投票
                $V_{\epsilon_1,\ldots,\epsilon_m} = \sum_{i=1}^m \epsilon_i b_i + A[f(X), Y_{\epsilon_1\epsilon_2\ldots\epsilon_m} + 0^{k-1}10^{n-k}]$
        Set x_k to be the majority of the above $2^m - 1$ votes.
    If $f(x_1, \ldots, x_k) = f(X)$, output $x_1 \ldots x_k$ and EXIT.

The above procedure runs in polynomial time, as $m = O(\log n)$. In the above procedure, as we run through all possible $b_1, \ldots, b_m$, in some iteration we would achieve $\forall i[b_i = X \cdot Y_i]$. Consider that particular iteration. What's the chance that we would get x_1 correctly? This

depends upon the what's the majority of the votes. Each $\epsilon_1, \ldots, \epsilon_m$ casts a vote for the value of x_1. The vote $V_{\epsilon_1,\ldots,\epsilon_m}$ will be correct, if A succeeds in computing $X \cdot \widehat{Y_{\epsilon_1\epsilon_2\ldots\epsilon_m}}$ correctly. We already observed that the $2^m - 1$ random variables $Y_{\epsilon_1\epsilon_2\ldots\epsilon_m}$ are uniformly distributed over $\{0,1\}^n$ and they are pairwise independent. So, by our assumption that A can compute the 2+1 nc,每张选票的正确率至少是这个概率. 我们现在对这2米 -1票申请Chebyshev。 让QQ成为随机变量, 计算出在这2米~1票中正确的票数. QQ是2m－1对偶的独立0-1随机变量的总和. 每一个随机变量都有成功概率,至少1

2+1 纳克. 对...

$$E[\chi] \geq (2^m - 1)\left(\frac{1}{2} + \frac{1}{n^c}\right)$$

Due to pairwise independence, var χ is the sum of variances of the individual 0-1 random variables. The variance of any 0-1 random variable is at most 1. So, var $\chi \leq 2^m - 1$. By Chebyshev,

$$\begin{aligned} Pr[\text{ majority of } 2^m - 1 \text{ votes wrong }] &= Pr\left[\chi < \frac{2^m - 1}{2}\right] \\ &\leq Pr\left[|\chi - E[\chi]| > \frac{2^m - 1}{n^c}\right] \\ &\leq \frac{\text{var } \chi}{\left(\frac{\text{nc (2米－1)}}{\text{(中文(简体)).}}\right)^2} \\ &= \frac{(2^m - 1)n^{2c}}{(2^m - 1)^2} \\ &= \frac{n^{2c}}{2^m - 1} \end{aligned}$$

以 m +(2c 对数 n + 2) 为单位。 那么x1不正确的概率最多为1n2. x1, x2 的任何一种概率。 . . .. . . 无。 xn 不正确的最多为 1 通过重复上述实验,我们可以提高我们的成功概率.

9.4 建造Pseudorandom发电机

显示如何使用 gl- hardcore 位元构建 PRG。

第10章

电路计算

接下来的几个讲座,我们将处理电路的复杂性。 首先我们将集中研究小深度电路。 由于一个电路的深度与平行计算时间相对应,因此小深度电路的复杂性捕捉高效的平行计算. 在我们证明电路下限之前, 电路下限不能用小深度电路计算, 我们应该对这些电路所能做的有一定的认识。 我们首先展示了这些电路的一些计算力。 我们从一个最简单的计算开始:整数加法。

10.1 二进制

Given two binary numbers, $a = a_{n-1} \ldots a_1 a_0$ and $b = b_n b_{n-1} \ldots b_1 b_0$, we can add the two using the *school method* – adding each column from right (the least significant bit) to left (the most significant bit), with carry bit along the way. In other words, $r = a + b$, where

$$\begin{array}{cccccc} & a_{n-1} & \ldots & a_1 & a_0 \\ + & b_{n-1} & \ldots & b_1 & b_0 \\ \hline r_n & r_{n-1} & \ldots & r_1 & r_0 \end{array}$$

can be accomplished by first computing $r_0 = a_0 \oplus b_0$ ($\oplus$ is exclusive or, which can be further written out in terms of $\vee$ and $\wedge$) and computing a carry bit, $c_1 = a_0 \wedge b_0$. Then, we can compute $r_1 = a_1 \oplus b_1 \oplus c_1$ and $c_2 = (c_1 \wedge (a_1 \vee b_1)) \vee (a_1 \wedge b_1)$, and in general, for $k \geq 1$ we have

$$\begin{aligned} r_k &= a_k \oplus b_k \oplus c_k \\ c_k &= (c_{k-1} \wedge (a_k \vee b_k)) \vee (a_k \wedge b_k) \end{aligned}$$

当然,上述操作可以在多语种时间内完成。 主要问题是,我们能否同时更快地完成这项工作? 注意上述计算是按顺序进行的。 之前,我说

计算 rk, 需要计算所有以前的输出位数。 当然,人们总是可以以CNF(或DNF)形式表达比特尔克. 这种线路只有2个深度,因此可以快速平行地增加。 但是,通过CNF(或DNF)这个论点会给出一般的指数大小的电路.

It turns out that there is indeed a way to do this in fewer levels (i.e. low depth or fast parallel time) and polynomial size. This is usually called *Carry-Look-Ahead*. In computing $r_k = a_k \oplus b_k \oplus c_k$, the main problem is computing c_k and what we would like is to be able to compute the carry bit c_k in one step for any k. Notice that for c_1 to be 1, we must have $a_0 = b_0 = 1$. In general, for c_k to be 1, a carry bit has to be generated at some position $i < k$, and should be propagated all the way through to position k. To be precise,

$$c_k = \exists i(0 \leq i < k) \left[\text{ii} \wedge \text{bi} \text{语} \bigwedge \forall j(i < j < k)(a_j \vee b_j) \right]$$

可以重写为

$$c_k = \bigvee_{0 \leq i < k} \left[\bigwedge_{\text{电子邮件:}} (\text{aj} \vee \text{bj}) \right]$$

注意这种计算可以经常深入进行,因为其中任何内容都不取决于以前的结果. 当然,为了不断深入地进行上述计算,我们需要有任意的粉丝的大门。

10.2 净捐助国和空调类

接下来我们来定义ACI和NCII的班级. 首先,我们定义AC0.

Definition. AC^0 is the set of languages that have constant depth, polynomial size circuits (unbounded fan-in). Formally, $L \in \text{AC}^0$ if for every n, there exists a circuit C_n with unbounded fan-in gates AND, OR and NOT, such that

1. $\forall x \in \{0,1\}^n, C_n(x) = L(x)$
2. $|C_n| < n^{O(1)}$
3. $\text{depth}(C_n) = O(1)$

我们显示,添加是在AC0中(形式上,我们应该定义一个语言版本的添加,例如总和的ITh比特). 我们说一种语言是统一的AC0

如果电路Cn在对数空间中可以计算。 意思是,有一个Turing machine,它给1n作为输入,输出电路Cn和机器运行在空间O(logn)中. 接着,我们概括了定义ACi的概念.

定义。 **ACi是语言L的集合,所以对于每个n,有一个带无约束的扇形门的电路Cn,并且,或者和NOT,这样可以**

1. $\forall x \in \{0,1\}^n, C_n(x) = L(x)$
2. $|C_n| < n^{O(1)}$
3. $\text{depth}(C_n) = O[(\log n)^i]$

ACi允许电路使用无约束的扇形门. 类NCi与ACi相似,只是只有有界的扇形门被NCi允许. 更确切地说,我们要求每个布尔 \\\\\\\\\\\\\\\\\\\\\\\\是二进制。 以下封装(统一NC和AC类)很容易看到:

$$\text{AC}^0 \subseteq \text{NC}^1 \subseteq \text{AC}^1 \subseteq \text{NC}^2 \subseteq \ldots \subseteq \text{P}$$

As with other hierarchies in complexity theory, we know very little about whether this hierarchy is strict or if it collapses. One thing we know is that $\text{AC}^0 \neq \text{NC}^1$. This is because, one can do parity in NC^1, but not in AC^0. It is easy to see that we do parity in NC^1, by divide and conquer. Later, we will show that parity cannot be done in AC^0.

10.3 乘法

现在,我们已经显示二进制加法在AC0,我们把注意力转向二进制乘法. 同样,学校方法包括将一个数字乘以另一个数字的每个数字,并添加所有结果以获得最终结果。 但是,因为我们正在做二进制乘法, 每个数字是1或0。因此,我们要么将数字(适当转移)写成,要么不写,然后把这些数字(大多数为n)加在一起。 因此,乘法只不过是这些n数字的加法,每个数字最多2n−1比特,如下例所示.

			1	0	0	1
×			1	1	0	1
			1	0	0	1
	1	0	0	1		
1	0	0	1			
1	1	1	0	1	0	1

So the question is how to add m binary numbers, each of at most m bits. To add these numbers quickly, we introduce a new trick called 3-2 trick. This trick is to take 3 numbers a, b, c, and output 2 new numbers d, e, such that the sum of the 2 new numbers is the same as the sum of the 3 input numbers: $a+b+c = d+e$. To do this, we imagine doing an addition at every bit position i separately. Given a_i, b_i and c_i, we can produce the sum as two bits, $uv \in \{00, 01, 10, 11\}$. The school method would have set the leading bit u as the carry bit and propogated onward. Instead we write down v as the ith bit of d, and u as the $i+1$th bit of e, where d and e are the two output numbers. More precisely, for $0 \leq i < n$ in parallel, we take the three bits a_i, b_i, and c_i, and set $d_i = a_i \oplus b_i \oplus c_i$ and $e_{i+1} = (a_i \wedge b_i) \vee (a_i \wedge c_i) \vee (b_i \wedge c_i)$.

现在,我们把增加的3个数字转换为增加2个数字的恒定深度,带有边框的扇形电路(它们有时被称为NC0).

为了利用它来增加m数,我们应用分割和征服. 在O(log m)平行回合的3-2技巧之后,我们剩下两个数字要加,此时我们用抬看-头法加起来. 因此,在NC1中可以实现加入m-bit数字的平行时间(深度).

也存在于NC1中的其他操作是整数分裂和内产. 使用"中国留守定理",可以显示师在NC1中. (就划分而言,线路的统一性是一个棘手的问题。) 多年来,人们只能实现P-unififiform;但最近已经表明,可以在对流空间统一NC1中实现。)

10.4 内产物、矩阵功率和三角线系

We now turn our attention to matrix computations and solving linear systems of the form $Ax = b$. First, consider multiplying two $n \times n$ matrices A and B. Each entry of the output matrix is an inner product of a row of A and a column of B. All these can be carried out in parallel. Each inner product involves n multiplications and then summing up the products. The multiplications can be carried out in parallel in NC^1. Summing up the n products can be done in a "binary tree fashion" with $\log n$ levels. Thus matrix multiplication can be done in NC^1. How about computing A^m? Using repeated squaring, it is clear we need only $O(\log m)$ levels of matrix squaring and multiplying. In particular, for a $n \times n$ matrix A with each entry an $O(n)$-bit integer, $A^{O(n)}$ can be computed in NC^2.

Our next goal is to solve linear equations. Recall that a linear system is of the form $Ax = b$ where A is a matrix of coefficients, b is a vector of constants and x is a vector of unknowns. To simplify our discussion, suppose that A is a non-singular $n \times n$ matrix.

我们首先考虑的是最简单的情况。 假设A是下三角形,单位对角;即A是形式

$$A = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ a_{2,1} & 1 & 0 & \dots & 0 \\ a_{3,1} & a_{3,2} & 1 & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ a_{n,1} & a_{n,2} & a_{n,3} & \dots & a_{n,n} \end{pmatrix}$$

We first write $A = I - \Delta$ where Δ is strictly lowe triangular (i.e. has zeros along the diagonal as well as above the diagonal). Note that Δ^2 has zeros in not only the upper triangular and

diagonal entries, but also the diagonal just above the main diagonal. Δ is called a *nilpotent* matrix, in that for some m, $\Delta^m = 0$. In fact $\Delta^n = 0$. By analogy of geometric sereis, we write

$$(I - \Delta)^{-1} = \sum_{i=0}^{\infty} \Delta^i$$

As $\Delta^n = 0$, only the first n terms of the above series are non-zero. So,

$$(I - \Delta)^{-1} = I + \Delta + \Delta^2 + \ldots \Delta^{n-1}$$

Of course, so far this argument is dubious, since it is arrived at by analogy. This argument can be made rigorous. However we will be theoretical computer scientist for once, a.k.a. mathematicians in a hurry, and justify this simply by verifying that the finite sum exhibited indeed is the inverse of $I - \Delta$, QED. Thus, if our linear system is such that A is lower triangular, (as A is non-singular, by dividing out the diagonal entries, we can assume it has unit diagonals), we can compute the inverse of A by evaluating the above finite series. This involves only matrix multiplication and addition. Finally, the solution is given by $x = A^{-1}b$. This entire process is certainly in NC^2.

这个过程对于降低三角矩阵作用很大,但我们想要解决任何线性系统. 我们可以使用高斯式的消除,这当然不是平行的,因为每一步骤都取决于在前一步中所做的工作。 我们想以某种方式将一般矩阵倒置的问题降低到低三角矩阵倒置的问题,同时有效地进行。 我们接下来做

10.5 决定方程和线性方程

我们在NC2中演示如何解决一般线性方程.

The goal is to solve a given system of linear equations: $Ax = b$, where A is a non-singular $n \times n$ matrix, b an n dimensional column vector and $x = (x_1, x_2, \ldots, x_n)$ is the vector of unknowns. Our aim is to compute the determinant of A in NC^2. Once we do that, the solution x can be found by using Cramer's rule: $x_i = \det(A_i)/\det(A)$, where A_i is the matrix obtained by replacing the ith column of A by the column vector b.

10.5.1 矩阵追踪

N × n 矩阵A的追踪表示Tr(A),是其对角条目的总和。 这是,

$$\mathrm{Tr}(A) = \sum_{i=0}^{n}$$

我们接下来要说明追踪的一些特性。

PROPOSITION 10.1 *For two $n \times n$ matrices A and B, $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$.*

证得.

$$\mathrm{Tr}(AB) = \sum_i \sum_k a_{ik} b_{ki} = \sum_k \sum_i a_{ik} b_{ki} = \sum_k \sum_i b_{ki} a_{ik} = \mathrm{Tr}(BA)$$

♣

以下的lemma以一个矩阵的分光分解而闻名. 矩阵J(称为约旦正常形式)的更精确的信息是已知的,但我们只需要这种形式。 我们将利用lemma 来建立更多的 追踪属性和精华。

LEMMA 10.2 *Let A be an $n \times n$ matrix. There exist $n \times n$ matrices T and J such that $A = T^{-1}JT$, where J is an upper triangular matrix whose n diagonal entries are the n eigenvalues of A.*

PROPOSITION 10.3 *Let A be an $n \times n$ matrix and $\lambda_1, \lambda_2, \ldots, \lambda_n$ be its eigenvalues. Then, for $k \geq 0$,*

$$\mathrm{Tr}(A^k) = \sum_{i=1}^{n} \lambda_i^k.$$

Proof. Let $A = T^{-1}JT$. Then we have

$$A^k = (T^{-1}JT)^k = T^{-1}J^kT$$

So,

$$\mathrm{Tr}(A^k) = \mathrm{Tr}(T^{-1}J^kT) = \mathrm{Tr}(T^{-1}TJ^k) = \mathrm{Tr}(J^k)$$

第二项平等来自建议10.1。 注意J是一个上三角矩阵,其n对角元素为 1,λ2,. . . .. , λ。 因此,Jk的对角元素是QQk

$$\sum_{i=1}$$

♣

Next, we can express the determinant of a matrix as the product of its eigenvalues.

其次,我们可以将矩阵的决定因素表述为它的精华值的产物。

PROPOSITION 10.4 *For an $n \times n$ matrix A with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, we have*

10.4 用于含有 1,λ2, 2的n×n矩阵A. . . .. , ,我们有

$$\prod_{i=1}^{n}$$

Proof. Consider the decomposition $A = T^{-1}JT$. Then,

$$\det(A) = \det(T^{-1}JT) = \det(J) = \prod_{i=1}^{n} \lambda_i$$

♣

From last section, for $1 \leq k \leq n$, we know how to compute A^k in NC^2. Hence, we can also compute $\mathrm{Tr}(A^k) = \sum_{i=1}^{n} \lambda^k$. Notice that we have computed all these sums without explicitly computing $\lambda_1, \lambda_2, \ldots, \lambda_n$. Our aim is to compute $\det(A)$, the product of eigenvalues. One could explicitly compute these eigenvalues and then take their product. But, computing eigenvalues is at least as hard as computing the determinant. Instead, we will directly compute their product using the known traces $\mathrm{Tr}(A^k)$, $1 \leq k \leq n$.

将eigen值视为未知值或变量. 然后,Tr(Ak)和det(A)是这些变量之上的多名数。 这些多诺是对称的多诺体,我们可以用多诺体Tr(Ak)来表达多诺体的det(A). 为此,我们将稍有分歧,研究对称的多名词。

10.5.2 对称多名称

我们将专注于n变量x1,x2上的多名数. . . .. ,xn. (中文(简体)). 相对于这些变量的多名制是对称的, 如果重新命名变量不会改变多名制. 形式上

Definition 10.5 (Symmetric polynomial) *A polynomial p on $x_1, x_2, \ldots, x_n$ is* symmetric *if $\forall \sigma \in S_n$ (i.e., for all possible permutations of 1 through n), $p(x_1, \ldots, x_n) = p(x_{\sigma_1}, x_{\sigma_2}, \ldots, x_{\sigma_n})$.*

Two families of symmetric polynomials are important for our purpose here: Newton's symmetric polynomials and elementary symmetric polynomials. We first give Newton's symmetric polynomials. For $0 \leq k \leq n$, the kth Newton's symmetric polynomial is given by

$$s_k = \sum_{i=1}^{n} x_i^k$$

It is clear that these polynomials are symmetric. We could define s_k for $k > n$, but we will only need $0 \leq k \leq n$. Note that $s_0 = n$.

We now give the elementary symmetric polynomials. For $1 \leq k \leq n$, the kth elementary symmetric polynomial is given by

$$p_k = \sum_{1 \leq i_1 < i_2 < \ldots < i_k \leq n} x_{i_1} x_{i_2} \ldots x_{i_k}$$

Since every choice of k variables appears as a product exactly once, these polynomials are symmetric. It is convenient to define $p_0 = 1$. Note that $p_n = \prod$

(Notice the correspondence between these polynomials and trace and determinant we discussed before. In particular, $\mathrm{Tr}(A^k) = \sum_{i=1}^{n} \lambda_i^k$ is the kth Newton's symmetric polynomial, whereas, $\det(A) = \prod_{i=1}^{n} \lambda_i$ is the nth elementary symmetric polynomial.)

人们可以用牛顿的多名词来表示基本对称的多名词. 举例来说,

$$p_1 = \sum_{i=1}^{n} x_i = s_1.$$

Similarly, since $s_2 = \sum_{i=1}^{n} x_i^2$,

$$p_2 = \sum_{1 \le i < j \le n} x_i x_j.$$

Thus
因此,

$$s_1^2 = \left(\sum x_i\right)^2 = s_2 + 2p_2.$$

Therefore
因此,

$$p_2 = \frac{1}{2}(s_1^2 - s_2).$$

事实上,任何对称的多诺可以表示为牛顿的多诺,也可以表示为基本对称的多诺:

定理10.6(对称多音的基本定理),x1上的每一个多音节. . . .. , xn 是对称的iffit可以表示为s1的多名,. . . .. , sn (以及p1),. . . .. (原始内容存档于2018-09-29)., pn.

我们可以用这个定理来联系基本对称的多诺米和牛顿的多诺米。 但是,我们需要更具体的公式,把多名制的两个家族联系起来。 因此,接下来我们阐述并证明这两个家庭之间的关系,这种关系在计算NC2中的决定因素方面是有用的.

THEOREM 10.7 *For* $1 \le k \le n$,

$$p_k = \frac{1}{k}(p_{k-1} \cdot s_1 - p_{k-2} \cdot s_2 + \cdots \pm p_0 \cdot s_k)$$

Proof. For $k, m \ge 0$, define the polynomial:

$$f_k^m = \sum_{1 \le i_1 < \ldots < i_k \le n j \notin \{i_1, \ldots, i_k\}} x_{i_1} x_{i_2} \ldots x_{i_k} x_j^m$$

有多少条件可以概括? 注意有两个不同的部分,产品x1xi2. . . .. 坐标:
j . 电话 有个
$\sum_{j=1}^{n} x_j^m = s_m$ and $f_k^{\upsilon} = (n-k) \cdot p_k$.

We now derive a formula for $p_k \cdot s_m$, when $k \geq 1$ and $m \geq 0$:

$$\begin{aligned} p_k \cdot s_m &= \left(\sum_{1 \leq i_1 < \ldots < i_k \leq n} x_{i_1} x_{i_2} \ldots x_{i_k} \right) \cdot \left(\sum_{j=1}^{n} x_j^m \right) \\ &= \sum_{1 \leq i_1 < \ldots < i_k \leq n, j \in \{i_1, \ldots, i_k\}} x_{i_1} x_{i_2} \ldots x_{i_k} x_j^m + \sum_{1 \leq i_1 < \ldots < i_k \leq n, j \in \{i_1, \ldots, i_k\}} x_{i_1} x_{i_2} \ldots x_{i_k} x_j^m \end{aligned}$$

The first sum is f_k^m. What about the second sum? It's a bit tricker, but a moment reflection convinces us that it is nothing but f_{k-1}^{m+1}, as each term in f_{k-1}^{m+1} occurs eactly once in the

$$p_k \cdot s_m = f_k^m + f_{k-1}^{m+1}$$

For $k = 0$, since $p_0 = 1$, $p_0 \cdot s_m = s_m = f_0^m$.

Now consider the alternating sum:

$$\begin{aligned} & p_0 \cdot s_1 - p_1 \cdot s_2 + \ldots + (-1)^k p_0 \cdot s_k \\ = \; & (f_k^0 + f_{k-1}^1) - (f_{k-1}^1 + f_{k-2}^2) + \ldots + (-1)^{k-1}(f_1^{k-1} + f_0^k) + (-1)^k f_0^k \end{aligned}$$

This is a telescoping sum, where all the terms cancel out, except for f_k^0. By definition, $f_k^0 = (n-k)p_k$. Hence,

$$p_0 \cdot s_1 - p_1 \cdot s_2 + \ldots + (-1)^k p_0 \cdot s_k = (n-k)p_k$$

Note that $s_0 = n$ and $p_0 = 1$. So,

$$(-1)^{k-1} p_1 s_{k-1} + \cdots + p_{k-2} s_2 - p_{k-1} s_1 + k p_k = (-1)^{k-1} s_k.$$

♣

10.5.3 Csanky's Algorithm for Determinant

10.5.3 Csanky对决定因素的算法

Let us now see how to compute the determinant of a given non-singular $n \times n$ matrix A. Let its eigenvalues be $\lambda_1, \lambda_2, \ldots, \lambda_n$. We can compute $S_k = \mathrm{Tr}(A^k)$ in NC^2. Our goal is to compute,

$$\det(A) = P_n = \lambda_1 \lambda_2 \ldots \lambda_n$$

Think of the eigenvalues as variables or unknowns. Then, S_k and P_k can be viewed as polynomials. For each $1 \leq k \leq n$, S_k is nothing but the kth Newton's symmetric polynomial and P_k is nothing but the kth elementary symmetric polynomial. We have these equalities relating P_k and S_k. Now, the idea is to treat $\{P_k\}$ as the unknowns and solve for $\{P_k\}$ as a system of linear equations. We can write

$$(-1)^{k-1} S_{k-1} P_1 + \cdots + S_2 P_{k-2} - S_1 P_{k-1} + k P_k = (-1)^{k-1} S_k.$$

记得我们开始解决线性方程系统的问题。 我们是否取得了任何进展? 对! 关键是,现在这个系统是低三角形的! 为了更加谨慎,让我们用矩阵形式写上上述系统:

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -\frac{S_1}{1} & 1 & 0 & \cdots & 0 \\ \frac{S_2}{2} & -\frac{S_1}{2} & 1 & \cdots & 0 \\ \vdots & \vdots & & & \\ \frac{(-1)^{n-2}S_{n-2}}{n} & \frac{(-1)^{n-3}S_{n-3}}{n} & \frac{(-1)^{n-4}S_{n-4}}{n} & \cdots & 1 \end{bmatrix} \begin{pmatrix} \\ \\ \\ \vdots \\ P_n \end{pmatrix} = \begin{pmatrix} S_1 \\ -\frac{S_2}{2} \\ +\frac{S_3}{3} \\ \vdots \\ \frac{(-1)^{n-1}S_n}{n} \end{pmatrix}$$

We already saw how to solve system of linear equations whose coefficient matrix is lower triangular in NC^2. Using that method, we can solve the above system of equations and compute $P_1, P_2, \ldots, P_n$. Finally, $P_n = \det(A)$ is the required answer.

第11章

电路下界

章节大纲:Razborov-Smolensky,决定树,切换莱玛,对等下限,对等不近似结果

11.1 历史注记

讨论电路下限的历史。 富士德-萨克森-西普赛尔,姚,蔡,哈斯塔德.

11.2 拉兹博罗夫-斯莫伦斯基定理

恒定深度大小 2nQQ布尔电路与 Mod 3 闸门无法计算等值。

我们将在今后两节中证明上述结果。 我们首先显示,常深电路可以被低度的多诺米电路所近似. 然后,我们表明,这种多名制不能大致发挥对等功能。

11.2.1 按低度聚子表示的常量深度电路

在本节中,我们显示,常深电路可以被低度的多诺米比GF(3)相近.

THEOREM 11.1 *We can approximate the function* $F(x_1, x_2, \ldots, x_n) = x_1 \vee x_2 \vee \ldots \vee x_n$ *with a low degree polynomial. More precisely, for any positive integer* k*, there is a degree* $\leq 2k$ *polynomial* P *in* $x_1, \ldots, x_n$*, over GF(3) that agrees with* y *on at least* $2^n(1 - ($(第3k段)

任务。 这是,

$$Pr_{\sigma\in\{0,1\}^n}[P(\sigma) = F(\sigma)] \geq 1 - \left(\frac{1}{3}\right)^k$$

这项索赔也适用于V、X和Mod3职能。

$$= Pr\left[\sum_{i=1}^{j} a_i c_i \neq 0\right]$$

lemma. Fix an assignment $\sigma_1 = (c_1, c_2, \ldots, c_n) \in \{0,1\}^n$. Choose $a_1, a_2, \ldots, a_n \in GF(3)$ uniformly at random. Consider the polynomial $\hat{F} = (\sum_{i=1}^{n} a_i x_i)^2$). Note that, as we are working in GF(3)={-1,0,1}, $\hat{F}$ evaluates to 0 or 1 for any input. What is the probability that $\hat{F}(\sigma_1) = F(\sigma_1)$? If $F(\sigma_1)$ evaluates to 0, then $\hat{F}$ also evaluates to 0 (regardless of the random values a_i). On the other hand, suppose $F(\sigma_1) = 1$. Let j be the largest index such that $c_j = 1$. Then

$$= Pr\left[\left(\sum_{i=1}^{j} a_i c_i\right)^2 = 1\right]$$

The last equality is due to the fact that in GF(3), $\left(\sum_{i=1}^{j} a_i c_i\right)^2 = 1 \iff \sum_{i=1}^{j} a_i c_i \neq 0$. We evaluate the last quantity by conditioning on the first $j-1$ terms of the summation. These could sum up to 0,1 or -1. But regardless this value, we have 2/3 chance for the summation to be non-zero, because $c_j = 1$ and a_j is chosen at random from GF(3). To conclude, for any fixed $\sigma_1 \in \{0,1\}^n$,

$$Pr[\hat{F}(\sigma_1) = 1] = Pr\left[\left(\sum_{i=1}^{} \right)^2 = 1\right] = Pr_{a_1,a_2,\ldots,a_n\in GF(3)}[\hat{F}(\sigma_1) = F(\sigma_1)] \geq$$

$$\frac{}{3}$$

有3n方法选择多诺米的系数ai,有2n

x1, x2 变量的任务。 . . .. ,xn. (中文(简体)). 考虑下表,每行代表一个可能的系数矢量,每列代表一个可能的指派。

	Assignment of the x_i				
	σ_1	σ_2	σ_3	$\cdots$	σ_{2^n}
$a^{(1)}$					
$a^{(2)}$					
$a^{(3)}$					
$\vdots$					
$a^{(3^n)}$					

Consider an assignment $\sigma_j = c_1, c_2, \ldots, c_n$ and a coefficient vector $a^{(i)} = (a_1, a_2, \ldots, a_n)$. Place a 1 in the cell $(a^{(i)}, \sigma_j)$ in the above table, if $(\sum_{j=1}^{n} a_i c_j^2 = F(c_1, c_2, \ldots, c_n,$ and place a 0 otherwise. By our probability calculation above, any column of the table has 1 in at least 2/3 of the cells. Thus, at least 2/3 of the entries in the entire table are 1. It follows that there is some row that has a 1 in at least 2/3 of its cells. Let this row be r_1 and let $\hat{F}_1$ be the polynomial corresponding to this row, i.e. $\hat{F}_1(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} a^{(\ )}$

assignment σ at random, we have

$$Pr_{\sigma\in\{0,1\}^n}[\hat{F}_1(\sigma) = F(\sigma)] \geq \frac{2}{3}$$

现在忽略 R1 行有 1 的所有列。 最多仍保留三分之一的栏目。 所有这些柱子至少有2/3的细胞有1个. 同样地,还有一些其他的r2行至少有2/3的剩余栏目中的1个. 我们让F2成为对应R2行的多名体. 接下来忽略这些列。 最多1/3 2

栏目将保留。 我们继续这样重复 最多将保留1/3k栏。 我们本来可以建造多鼻音的 + +F1, + +F2, ... (原始内容存档于2018-09-21). , QQFk.

Being the OR function, F evaluates to 0 for only one assignment, namely $\sigma_0 = (x_1 = 0, x_2 = 0, \ldots, x_n = 0)$. All our k polynomials evaluate to 0 on this assignment and for any other input they all evaluate to 0 or 1. Consider the columns (or assignments) that were ignored. These are "covered"by our polynomials in the sense that if $F(\sigma) = 1$ then at least one of these polynomials F_i also evaluates to 1 on σ. So if we take OR of our polynomials $\hat{F} = \hat{F}_1 \vee \hat{F}_2 \vee \ldots \vee \hat{F}_k$, then for any "covered" assignment σ, $\hat{F}(\sigma) = F(\sigma)$. But $1 - (1/3)^k$ 包括任务。 这么说

$$Pr_{\sigma}[\hat{F}(\sigma) = F(\sigma)] \geq 1 - \left(\frac{1}{3}\right)^k$$

Only one problem remains. How do we OR two polynomials to get another polynomial? We use the trick mentioned in the beginning. We let $\hat{F}_1 \vee \hat{F}_2 = 1 - (1 - \hat{F}_1)(1 - \hat{F}_2)$. So we take OR of all the k polynomials this way and let the required polynomial P to be $\hat{F}_1 \vee \hat{F}_2 \vee \ldots \vee \hat{F}_k$. What is the degree of P? Each $\hat{F}_i$ has degree 2 (because of the squaring). Thus P has degree at most $2k$.

因此,我们构建了最多2k的多名学位,在至少1 -(1/3)k任务上与OR函数F一致。 同样的,我们可以 大致和不起作用。

接下来我们显示,任何大小大于n输入的深度d电路C,都可以被低度的多诺米相近. 我们用上面的lemma 来用2k的多音量来估计每个门。 我们通过结合这些多名体来构建一个多名体,以近似于电路. 例如,考虑两个输入QQgate g,从另外两个门g1和g2(位于较低的深度)接收其输入. Lemma 11.1 建造的门g的多名式为 P(y1,y2). 我们用多诺米取代g1和g2的门,如P中的y1和y2的Fg1和Fg2,以获得门g的多诺米Fg. 这样,我们从输入级别开始,向上移动到电路的输出级别,使多诺米的FC来近似电路. 由于电路的深度为d,而门的多名数(来自Lemma 11.1为2k级),FC的度数将为2k)d.

Choose an input $x_1, x_2, \ldots, x_n \in \{0, 1\}$ at random. What is the probability that F_C does not agree with circuit C on this input? F_C will fail to agree with the circuit C only when at

至少有一个门的圆形失败。 根据Lemma 11.1提供的保证,对于任何特定的大门,这种可能性最多为3k。 因此,至少一个门故障的概率最多为13k。 因此,

$$Pr_{x_1,x_2,\ldots,x_n\in\{0,1\}}[F_C(x_1,x_2,\ldots,x_n)=C(x_1,x_2,\ldots,x_n)]\geq 1-\frac{s}{3^k}$$

我们已经证明,

11.2 任何深度d的电路C和在n布尔变量上的大小s,对于任何整数k,都有一个多诺米F(x1,x2,). . . .. , xn) , 最多2k)d ,超过GF(3),其值等于至少2n(1 − s/3k)输入的C输出。 电路被允许使用AND,OR,NOT和Mod3门.

11.2.2 低学位的多元性

11.3 换换Lemma和对等下限

11.3.1 决定树

决策树用于代表布尔函数。

11.3(决策树) 决策树是根植的二进制树t,每片叶子标注为0或1. 每个内部节点都有xi的标签. 内部节点的两个边缘分别标注为0和1. 我们并不反对从根到叶的路径重复变量。

定义 11.4(决策树深度) 决策树的深度被定义为从根到叶最长路径的边缘数.

图11.1显示了一个决策树的例子。

决定树 T 定义布尔函数如下:

1. if $\text{depth}(T)=0$, then it is a constant function (either 0 or 1)
2. if $\text{depth}(T)=d+1$ $(d\geq 0)$: inductively assume that the left subtree and the right subtree respectively define a boolean function. Call the boolean function defined by the left subtree G and the right subtree as H. Also assume that the root is labeled as x_i and has the left edge labeled 0 and the right edge labeled 1. Then,

$$F(x_1,x_2,\ldots,x_n)=\begin{cases} G(x_1,x_2,\ldots,x_n) & \text{if } x_i=0 \\ H(x_1,x_2,\ldots,x_n) & \text{if } x_i=1 \end{cases}$$

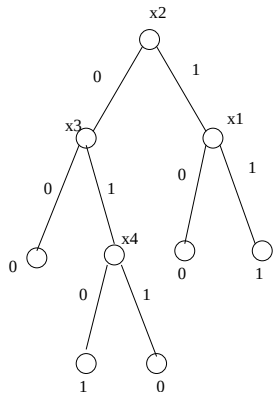


图11.1:决策树的例子

G和H在xi中也都有功能,但是如果我们不允许xi在决策树中从根到叶的任何路径重复,我们可以把它从G和H的功能中离开. 然后,我们可以将G写成x1,x2的函数. . . .. , xi − 1, xi + 1, (中文(简体)). . . .. , xn,它经常被写成G(x1, x2). . . .. ˆxi, ... , xn) (中文(简体)).

因此,职能F事实上是:

$$F = (G \wedge \bar{x_i}) \vee (H \wedge x_i)$$

We can instead choose to write G as $F|_{x_i=0}$ and H as $F|_{x_i=1}$ and rewrite the above equation as:

$$F = (F|_{x_i=0} \wedge \bar{x_i}) \vee (F|_{x_i=1} \wedge x_i)$$

This expansion gives us a way to write the formula in DNF (i.e., $\vee$ of $\wedge$'s). If $F|_{x_i=0}$ and $F|_{x_i=1}$ can be written as a DNF where every conjunct has size at most t, then we get a formula F in DNF in which every conjunct is of size at most $t + 1$.

定义 11.5(决策树复杂度) 对于布尔函数f,决策树复杂度DC(f)是计算 f 的决定树的最小深度.

LEMMA 11.6 $D(f) = D\left(\bar{f}\right)$

证明: 计算 f 的布尔决定树 与 计算 f 的 1-1 个对应。 (在树叶上翻转位数)

如上一节所示,用决定树深度d计算出来的布尔函数可以用除网的正常形式(DNF)表示布尔公式,其中每个连接点最多有大小d。 利用这个想法,我们得出以下必然结果。

COROLLARY 11.7 *A boolean function f and its complement $\bar{f}$ can be expressed in DNF in which every conjunct has size $\leq DC(f)$. So f can be written as a DNF formula with conjuncts of size $\leq DC(f)$, and as a CNF formula with disjuncts of size $\leq DC(f)$.*

上述DNF的特殊情况,即每个连体的大小都受限制,对我们的讨论将是有益的。 以下定义对此作了阐述.

DEFINITION 11.8 (t-AND-OR) *Boolean formula G is a t-AND-OR if $G = G_1 \wedge G_2 \wedge \cdots \wedge G_w$, where each G_i is the OR of at most t literals. So $G_i = \tilde{x}_1 \vee \tilde{x}_2 \vee \cdots \vee \tilde{x}_s$, for $s \leq t$, and $\tilde{x}_i \in \{x_i, \bar{x}_i\}$. (Likewise, a t-OR-AND is a formula in disjunctive normal form with at most t terms in each conjunct.)*

11.3.2 随机限制

DEFINITION 11.9 (RESTRICTION) *A restriction ρ is a partial assignment to variables of boolean formula G. Specifically, it is a mapping $\rho : \{1, 2, \cdots, n\} \mapsto \{0, 1, *\}$. The restriction of G by ρ, written $G|_\rho$, is the boolean function obtained by setting x_i to $\rho(i)$ if $\rho(i) \in \{0, 1\}$ and leaving x_i as a variable otherwise.*

随机的 p- 限制是一个限制 ˆ , 对于每个 i, 一个独立指定 ˆ (i) 这样的限制 :

$$\begin{aligned} \Pr[\rho(i) = *] &= p, \quad \text{and} \\ \Pr[\rho(i) = 0] = \Pr[\rho(i) = 1] \quad &= \quad \frac{1-p}{2} \end{aligned}$$

11.3.3 证明对等的下限线路:概述

在此,我们将对与电路计算对等的深度和大小有关的较低界限进行高级别的验证。 将在以后的讲座中提出正式证据。

在开始之前,让我们审查一些公约。 假设我们有一个使用否定门的电路 通过使用德摩根定律,我们可以推倒否定的大门,直到它们达到输入水平. 因此,任何布尔电路都可以转换成等效电路,使所有否定电门都发生在底层 (即: 任何否定门只从变量中输入). 这种转变可以达到最大两倍于原来的大小,并且不增加深度。 我们假设输入既包括变量,也包括其否定,这样一个电路的大小就是指AND和OR门的数量. 我们不限制扇形或扇形的大门。 因此,在不丧失通俗性的情况下,我们可以假设电路是“平面”的:深度d的电路是d级的,这样一个级别中的所有闸门都是同类型,边缘只在相邻的电位之间. 因此,和OR门将在各个级别交替。 从现在起,我们考虑

只有具有以上属性的电路。 按惯例,输出水平(仅用一个门)为1,底层水平(即大门从变量中输入)为d。

现在我们准备说明下限定理。

THEOREM 11.10 *For sufficiently large n,* Parity_n *cannot be computed by a depth-d circuit of size $\leq 2^{cn^{1/d}}$, where $c > 0$ is some constant.*

One can prove the theorem with $c = 0.143781$. Here we will prove a weaker version where $c \approx 0.1$. The main ingredient of the proof is the switching lemma, which is as follows.

LEMMA 11.11 *[Switching Lemma] Let G be a t-And-Or formula. Let ρ be a random p-restriction. Then, for all $\Delta \geq 0$,*

$$\Pr[DC(G|_\rho) > \Delta] \leq (5pt)^\Delta. \tag{11.1}$$

Assuming the switching lemma, we sketch a proof of Theorem 11.10. Set $p = \gamma_0/t$, where $\gamma_0 > 0$ is a suitable constant fixed later. We will apply the lemma with $\Delta = t$. Then, the lemma says that when we apply a random p-restriction to a t-AND-OR formula G, with probability at least $1 - (5\gamma_0)^t$, the resultant formula $G|_\rho$ has DC $\leq t$. As we noted before, a formula with DC $\leq t$, can be expressed as a t-OR-AND formula. Thus, with high probability, the t-AND-OR formula $G|_\rho$ can be "switched" into an equivalent t-OR-AND formula. Using this ability to "switch", we can prove Theorem 11.10.

We first consider some restricted type of circuits. A circuit C of depth d is said to be of type $C^d(s,t)$ if it satisfies the two conditions: i) the gates at the bottom-most level (i.e. input level) have fan-in $\leq t$; ii) number of gates at levels above the bottom level $\leq s$. Thus, we have a bottom fan-in condition (bfi), which restricts fan-in of the bottom level gates. The gates above the bottom level (or the "internal" gates) are allowed to have any fan-in. At the end, we will relax these conditions and prove the theorem for any d-depth circuit (this part is easy).

Assume, without loss of generality that, the gates at the bottom level are OR gates (if not consider $\neg C$). Each gate at one level above ($(d-1)^{th}$ level) is an AND gate which gets its input from the OR gates at level d. Think of each such AND gates and along with the OR gates as a small circuit. Any such small circuit sc is computing a t-AND-OR formula, because of the bottom fan in condition. Choose a random p-restriction ρ and consider the circuit $C|_\rho$. By the switching lemma, with probability $\geq 1-(5\gamma_0)^t$, the formula computed by the small circuit sc (in $C|_\rho$) has DC $\leq t$. As we discussed, this formula is (also) a t-OR-AND formula. Thus, we can *replace* the AND-OR circuit sc by an OR-AND circuit, where the AND gates have fan-in $\leq t$. Suppose there are s_{d-1} AND gates at level $d-1$. Assume that we were lucky and our random restriction ρ is good so that we are able to replace all these s_{d-1} AND-OR circuits by s_{d-1} OR-AND circuits with bottom fan-in $\leq t$. The resultant

circuit is made of OR gates in level $d-1$, OR gates is level $d-2$ and AND gates in level d. We can now *merge* the levels $d-1$ and $d-2$ in $C|_\rho$ to get a circuit that has only $d-1$ levels. There are a few things to note. The new circuit may have more gates at its bottom level than the original circuit. But, number of internal gates in the new circuits is $s - s_{d-1} \leq s$. Thus, the number of internal gates didn't increase. Next, bottom fan-in of the new circuit is $\leq t$. The new circuit has only $d-1$ levels. To summarize, the new circuit is of type $C^{d-1}(s,t)$ and is equivalent to $C|_\rho$. Of course, the new circuit is not computing the same function as the original one. But, this is not an issue when it comes to the parity function. We will elaborate this shortly.

现在的想法是应用以上切换d－2次的过程:我们选择一个随机p-限制 $1并将其应用到C,然后选择一个随机p-限制 $2并将其应用到C|ρ1等等. 假设我们很幸运能选择这些 D -2 随机限制。 然后,我们将留下一个C2(1,t)型电路。

假设原电路C在n位值上计算等值. 新电路C是什么?

计算? 在应用所有 d－2 随机限制后,一些 n 变量会被分配到 1 或 0 , 其余的将仍然是变量(即 分配到 *)。 让剩下的变量数为 N(这是一个随机变量). 注意C'计算这些N变量的等值.

假设所有 d - 2 随机限制都是好的,假设 N > t。 然后,C'是C2(1,t)型的电路,在N > t变量上计算等值. 这是不可能的。 (中文(简体)).

computes a formula $G_1 \wedge G_2 \wedge \ldots \wedge G_w$, where each G_i is an OR of $\leq t$ literals. In particular, G_1 has some $a \leq t < N$ literals. Set all these literals to 0. Then output of C' is 0. But, since $N > a$, G_1 does not include some variable x. Set all the variables except those in G_1 and x to be 0. Now, set x appropriately, so that the parity is 1. Thus, C' does not compute parity of N variables).

其余的证明是,以非零概率表示两个属性得到满足: (一)d－2随机限制是好的(即. 它们允许我们在所有情况下将 t-OR-OR电路"转换"为 t-OR-AND电路)和 ii) N > t。 我们首先指出,我们不选择d-2随机限制,而是可以选择一个随机限制。 假设我们有一个公式F,我们应用一个随机p1限制,然后是随机p2限制. 注意, 等效的, 我们可以应用随机(p1 ^ p2) 限制。 (两种情况下获得的布尔函数的分布相同). 在我们的假设中,我们不逐个应用d－2随机p－限制,而是可以等效地应用单个随机pd－2-限制.

我们有一个Cd(s, t)型的电路和一个随机的pd－2-限制 $. 我们需要一个估计

on the probability that $C|_\rho$ can be transformed (by switching) into a $C^2(1,t)$ circuit. As C has s internal gates, we will need do s many "switches". The probability of failing in any one of these instances is bounded $\leq (5pt)^\Delta = (5\gamma_0)^t$. Thus, probability that we fail to convert $C|_\rho$ into an equivalent $C^2(1,t)$ circuit is $\leq s \cdot (5\gamma_0)^t$. Formally,

$$\Pr[C|_\rho \text{ is not equivalent to a } C^2(1,t) \text{ circuit}] \leq s \cdot (5\gamma_0)^t \tag{11.2}$$

Now we estimate the random variable N, the number of input variables of $C|_\rho$. $E[N] = n \cdot p^{d-2} = n \cdot (\gamma_0/t)^{d-2}$.

We will fix γ_0 and t to obtain a bound on $E[N]$ and the RHS of (11.2). Set $\gamma_0 = 1/10$ and $t = \gamma_0 n^{1/(d-1)}$ Then,

$$\Pr[C|_\rho \text{ is not equivalent to a } C^2(1,t) \text{ circuit}] \leq s \cdot (5\gamma_0)^t = s \cdot 2^{-(0.1)n^{\frac{1}{(d-1)}}}.$$

We have $E[N] = n^{1/(d-1)}$. Then, using Chernoff bound, we get

$$\Pr[N < t] = \Pr[N < \gamma_0 \cdot E[N]] < e^{-\frac{(1-\gamma_0)^2}{2} \cdot n^{\frac{1}{d-1}}} < e^{-(0.4)n^{\frac{1}{d-1}}}$$

Let $c < 0.1$ be a constant. For sufficiently large n, if $s \leq 2^{-c \cdot n^{1/(d-1)}}$, then

$$s \cdot 2^{-(0.1)n^{\frac{1}{(d-1)}}} + e^{-(0.4)n^{\frac{1}{d-1}}} < 1.$$

让我们总结一下。 让 < 2 - c {n1/(d−1)} 和 C 成为计算 n 位元的 Cd(s, t) 型电路。 然后,用非零概率,CQQ 可以被转换成C2(1,t)电路,在N > t比特上计算平价. 特别是,存在着一种随机限制,即满足这两个特性。 但是,C2(1,t)型的电路不能在 > t比特上计算等值. 一个矛盾。 我们证明了以下的定理。

THEOREM 11.12 *For $s \leq 2^{-c \cdot n^{1/(d-1)}}$, circuits of type $C^d(s,t)$ cannot compute parity on n bits. Here, c can be any constant < 0.1.*

现在很容易证明定理11.10. 观察大小的S和深度的d的电路可以看作是Cd + 1(s, 1)型的电路. 然后使用定理11.12. 稍后,我们将证明切换lemma的更好版本,进行仔细分析,并将整流体c从0.1改进为0.143781.

11.3.4 切换莱玛:证据

在这场演讲中,我们证明了换掉Lemma。

LEMMA 11.13 *Let G be a t-And-Or formula $G_1 \wedge G_2 \wedge \ldots \wedge G_w$. Let ρ be a random p-restriction. Then, for all $\Delta \geq 0$,*

$$\Pr[\ \mathrm{DC}(G|_\rho) \geq \Delta\ ] \ \leq\ (5pt)^\Delta. \tag{11.3}$$

莱玛 11.14 让G成为t-And-Or公式 G1 QQG2 QQ. . . .. ∧. 对于任何β, 0 < β < t,

let ρ *be a random p-restrıctıon, where* $p = \frac{\rho}{t-\beta}$*, and let* $\alpha = \beta/ln\left\lfloor \frac{\cdot \cdot \vee \cdot \cdot \cdot}{2} \right\rfloor$. . . .. 那为所有人 $\Delta \geq 0$*, we have*

$$\Pr[DC(G|\rho) \geq \Delta] \leq \alpha^{\Delta}.$$

证得. 我们一起证明这两个小精灵 我们将证明以下更强烈的说法:

对于任何布尔函数 F,

$$\Pr[\ \mathrm{DC}(G|_\rho) \geq \Delta \mid F|_\rho \equiv 1\ ] \ \leq \ \alpha^{\Delta}, \tag{11.4}$$

α将设为5pt。 (如果条件不能满足,则条件概率被定义为0.) Lemma 11.13随即将F作为常态函数1.

直觉如下. 我们将用参数 p 指定一些随机限制。 注意,p是 ,bfi。 因此,人们期望一个连接点,比如G1,在此 ˆ 之后的 G1 中 ˆ 左值为 0,即所有变量都以某种方式分配。 由于我们以 ± 的概率分配它,并且由于 G1 是 OR,我们期望其中的一个变量 被以正确的方式分配,以便 G1 → 1%. 因此,G1QQ1是一个低信息事件. 我们将“传递”这一信息,然后前往G2。 G1中的某些xi 得到我们想要收取p价格(概率)的“稀有”事件。

在这个过程中,我们将进行越来越多的形式事件:在一个集合子集中,每个子集至少有一个被特定方式分配的词组(0或1). 这种条件的一般形式不亚于说出某种情况,也说出经评估为真实的其他资源情况。 但这只是一个条件,即任意布尔函数在限制下是真实的。 这是迄今在逐一处理Gi的过程中积累的“信息”。

注意,至少直观地,鉴于限制 $ 下的一些任意布尔函数 F 属实,(即 FQQ QQ 1) 并不降低任何变量收到 * 的概率 p。 现在我们正式证明这个定理

The statement (11.4) is trivially true for $\Delta = 0$, since the RHS becomes 1 in this case. Thus, we may assume $\Delta > 0$. We prove (11.4) by induction on w, the number of clauses conjuncted to form G. The base case of $w = 0$ is trivial, since $G \equiv 1$ by definition and the statement holds since the LHS is 0. By induction, assume that (11.4) holds for all conjuncts of up to $w - 1$ clauses.

Let G be a t-And-Or formula $G_1 \wedge G_2 \wedge \ldots \wedge G_w$. Let $G' = G_2 \wedge \ldots \wedge G_w$. Let ρ be a random p-restriction. By renaming literals, we can assume WLOG that $G_1 = \bigvee_{i \in T} x_i$ for some T with $|T| \leq t$.

为了证明这一说法,我们考虑两起案件:G1+++++++

其余证据涉及G1|ρ1案件。 也就是说,我们也想证明以下声明。

$$\Pr[\ \mathrm{DC}(G|_\rho) \geq \Delta \mid F|_\rho \equiv 1, G_1|_\rho \not\equiv 1\ ] \ \leq\ \alpha^{\Delta} \tag{11.5}$$

We have renamed the variables so that $G_1 = \bigvee_{i\in T} x_i$. Then $G_1|_\rho \not\equiv 1$ means that for each $i \in T$, $\rho(i) = 0$ or $*$. For each ρ such that $G_1|_\rho \not\equiv 1$, consider the set of all $i \in T$ such that $\rho(i) = *$. Moreover, if $\rho(i) = 0$ for all $i \in T$, then $G_1|_\rho \equiv 0$ and hence, $G|_\rho \equiv 0$ implying $DC(G) = 0$. Thus, it is not the case that $\rho(i) = 0$ for all $i \in T$. So,

$$\begin{aligned}&\{\rho\ :\ F|_\rho \equiv 1, G_1|_\rho \not\equiv 1, \mathrm{DC}(G|_\rho) \geq \Delta\}\\ &\quad = \bigcup_{\emptyset\neq Y\subseteq T} \{\rho\ :\ \rho(Y) = *, \rho(T-Y) = 0, F|_\rho \equiv 1, \mathrm{DC}(G|_\rho) \geq \Delta\,\}.\end{aligned} \tag{11.6}$$

让

$$a_Y \ \ =\ \ \Pr[\ \rho(Y) = * \wedge \rho(T-Y) = 0 \mid F|_\rho \equiv 1 \wedge G_1|_\rho \not\equiv 1\ ]. \tag{11.7}$$

因此,(11.8)可以写成

$$\begin{aligned}&\Pr[\ \mathrm{DC}(G|_\rho) \geq \Delta \mid F|_\rho \equiv 1, G_1|_\rho \not\equiv 1\ ]\\ &\quad = \sum_{\emptyset\neq Y\subseteq T} a_Y \cdot \Pr[\ \mathrm{DC}(G|_\rho) \geq \Delta \mid F|_\rho \equiv 1 \wedge G_1|_\rho \not\equiv 1 \wedge \rho(Y) = * \wedge \rho(T-Y) = 0]\\ &\quad = \sum_{\emptyset\neq Y\subseteq T} a_Y \cdot \Pr[\ \mathrm{DC}(G|_\rho) \geq \Delta \mid F|_\rho \equiv 1 \wedge \rho(Y) = * \wedge \rho(T-Y) = 0]\end{aligned} \tag{11.8}$$

我们将从以上角度考虑个别条款,以此限定这一数额。 首先考虑固定的`Y ' < <。

Fix ρ such that $\mathrm{DC}(G|_\rho) \geq \Delta$ and $F|_\rho \equiv 1 \wedge \rho(Y) = * \wedge \rho(T-Y) = 0$. We claim that there exists $\sigma_Y : Y \to \{0,1\}$, such that $\mathrm{DC}(G|_\rho|_{\sigma_Y}) \geq \Delta - |Y|$.

Suppose not. Then for all $\sigma_Y : Y \to \{0,1\}$, $\mathrm{DC}(G|_\rho|_{\sigma_Y}) < \Delta - |Y|$. So we can construct a decision tree for $G|_\rho$ as follows. Start with a full binary tree of depth $|Y|$ in which each level determines the value of x_i for some $i \in Y$. Now let each leaf of this full binary tree, which represents a restriction σ_Y, be the root of a decision tree for $G|_\rho|_{\sigma_Y}$ of depth $\leq \Delta - |Y|$. The result is a decision tree for $G|_\rho$ of depth $\leq \Delta$. So $\mathrm{DC}(G|_\rho) < \Delta$, which is a contradiction.

We conclude from this contradiction that such a σ_Y exists. We also note that $\sigma_Y \neq 0^Y$. For if $\sigma_Y = 0^Y$, then $G_1|_\rho|_{0^Y} \equiv 0$, so $G|_\rho|_{0^Y} \equiv 0$, so $\mathrm{DC}(G|_\rho|_{0^Y}) = 0$.

Since $\sigma_Y \neq 0^Y$, we have $G_1|_\rho|_{\sigma_Y} \equiv 1$, so $G|_\rho|_{\sigma_Y} \equiv G'|_\rho|_{\sigma_Y}$ and $\mathrm{DC}(G'|_\rho|_{\sigma_Y}) = \mathrm{DC}(G|_\rho|_{\sigma_Y}) \geq \Delta - |Y|$.

现在,再次将 \ 作为随机的p -限制,我们

$$\Pr[\ \mathrm{DC}(G|_\rho) \geq \Delta \mid F|_\rho \equiv 1 \wedge \rho(Y) = * \wedge \rho(T-Y) = 0]$$

$$
\begin{aligned}
&\leq \Pr[\exists \sigma_Y : Y \to \{0,1\},\ \mathrm{DC}(G'|_\rho|_{\sigma_Y}) \geq \Delta - |Y| \mid F|_\rho \equiv 1 \wedge \rho(Y) = * \wedge \rho(T-Y) = 0] \\
&\leq \sum_{\substack{\sigma_Y : Y \to \{0,1\} \\ \sigma_Y \neq 0^Y}} \Pr[\ \mathrm{DC}(G'|_\rho|_{\sigma_Y}) \geq \Delta - |Y| \mid F|_\rho \equiv 1 \wedge \rho(Y) = * \wedge \rho(T-Y) = 0]
\end{aligned}
\tag{11.9}
$$

G' is a formula with $w-1$ clauses. We would like to invoke the induction hypothesis to bound the above sum. But, the condtion "$F|_\rho \equiv 1 \wedge \rho(Y) = * \wedge \rho(T-Y) = 0$" does not fit the induction hypothesis. We can overcome this as follows. Suppose $\rho(Y) = * \wedge \rho(T-Y) = 0$. Then, $F|_\rho \equiv 1$ iff for any (partial) truth assignment $\tau_Y : Y \to \{0,1\}$, we have $F|_\rho|\tau_Y \equiv 1$. To express this formally, set

$$
\begin{array}{lcl}
0^{T-Y} & = & \text{the all 0 assignment on } T-Y, \\
\widetilde{F} & = & \bigwedge_{\tau_Y : Y \to \{0,1\}} F|_{0^{T-Y}}|_{\tau_Y} \ \text{ and} \\
\widetilde{\rho} & = & \rho \text{ restricted to the complement of } T.
\end{array}
$$

Then whenever $\rho(Y) = * \wedge \rho(T-Y) = 0$, as in the condition on the probabilities in (11.9),

$$
\widetilde{F}|_{\widetilde{\rho}} \equiv 1 \iff F|_\rho \equiv 1
$$

Again, suppose $\rho(Y) = *$ and $\rho(T-Y) = 0$. Then, for any $\sigma_Y : Y \to \{0,1\}$, we have $G'|_\rho|\sigma_Y$ is equivalent to $(G'|_{0^{T-Y}}|_{\sigma_Y})|_{\widetilde{\rho}}$. Thus,

$$
\begin{aligned}
&\Pr[\ \mathrm{DC}(G'|_\rho|_{\sigma_Y}) \geq \Delta - |Y| \mid F|_\rho \equiv 1 \wedge \rho(Y) = * \wedge \rho(T-Y) = 0] \\
&\leq \Pr[\ \mathrm{DC}((G'|_{0^{T-Y}}|_{\sigma_Y})|_{\widetilde{\rho}}) \geq \Delta - |Y| \mid \widetilde{F}|_{\widetilde{\rho}} \equiv 1]
\end{aligned}
$$

As $G'|_{0^{T-Y}}|_{\sigma_Y}$ has (at most) $w-1$ clauses, we can apply induction hypothesis and get

$$
\Pr[\ \mathrm{DC}((G'|_{0^{T-Y}}|_{\sigma_Y})|_{\widetilde{\rho}}) \geq \Delta - |Y| \mid \widetilde{F}|_{\widetilde{\rho}} \equiv 1] \leq \alpha^{\Delta - |Y|}
$$

因此(11.9)成为

$$
\Pr[\ \mathrm{DC}(G|_\rho) \geq \Delta \mid F|_\rho \equiv 1 \wedge \rho(Y) = * \wedge \rho(T-Y) = 0] \ \leq \ (2^{|Y|} - 1)\alpha^{\Delta - |Y|} \tag{11.10}
$$

We have proved the above bound for $|Y| < \Delta$. Now consider the case $|Y| \geq \Delta$. Here (11.10) holds trivially, because $|Y| \geq \Delta > 0$ and $\alpha < 1$ imply $(2^{|Y|} - 1)\alpha^{\Delta - |Y|} \geq 1$.

因此(11.8)成为

$$
\Pr[\ \mathrm{DC}(G|_\rho) \geq \Delta \mid F|_\rho \equiv 1, G_1|_\rho \not\equiv 1\ ] \ \leq \sum_{\emptyset \neq Y \subseteq T} a_Y \cdot (2^{|Y|} - 1)\alpha^{\Delta - |Y|} \tag{11.11}
$$

让

$$
b_Y \ = \ \Pr[\ \rho(Y) = * \mid F|_\rho \equiv 1 \wedge G_1|_\rho \not\equiv 1\ ]. \tag{11.12}
$$

Clearly $b_Y \geq a_Y$. Also, we see intuitively that $Z \subseteq T$, $b_Z = \Pr[\ \rho(Z) = * \mid F|_\rho \equiv 1 \wedge G_1|_\rho \not\equiv 1\ ] \leq q^{|Z|}$, where $q = p/(p + \frac{1-p}{2})$ is the probability of ρ assigning any variable to be $*$ given that it is assigned to be either $*$ or 0. For we already saw that $G_1|_\rho \not\equiv 1$ means that each variable in Z is assigned either 0 or $*$. The additional condition that $F|_\rho \equiv 1$ can only decrease the probability that some variable is assigned a $*$. We will give a rigorous proof that $b_Z \leq q^{|Z|}$ at the end of these notes.

由此可见

$$\begin{aligned}
\Pr[\ \mathrm{DC}(G|_\rho) \geq \Delta \mid F|_\rho \equiv 1, G_1|_\rho \not\equiv 1\ ] &\leq \sum_{\emptyset \neq Y \subseteq T} a_Y \cdot (2^{|Y|} - 1)\alpha^{\Delta - |Y|} \\
&= \sum_{Y \subseteq T} a_Y \cdot (2^{|Y|} - 1)\alpha^{\Delta - |Y|} \qquad (11.13) \\
&\leq \sum_{Y \subseteq T} b_Y \cdot (2^{|Y|} - 1)\alpha^{\Delta - |Y|} \\
&\leq \sum_{Y \subseteq T} q^{|Y|} \cdot (2^{|Y|} - 1)\alpha^{\Delta - |Y|} \\
&\leq \alpha^\Delta \left[\left(\sum_{Y \subseteq T} \left(\frac{2q}{\alpha} \right)^{|Y|} \right) - \left(\sum_{Y \subseteq T} \left(\frac{q}{\alpha} \right)^{|Y|} \right) \right] \\
&= \alpha^\Delta \left[\left(1 + \frac{2q}{\alpha} \right)^{|T|} - \left(1 + \frac{q}{\alpha} \right)^{|T|} \right] \\
&\qquad (11.14)
\end{aligned}$$

上述衍生物中的最后一种平等是使用身份获得的,P

$$_{X \subseteq A}\, x^{|X|} = (1 +$$

If we set $c = 1/\log_e \phi \approx 2.078$, where $\phi = \frac{1+\sqrt{5}}{2} \approx 1.618$ is the golden ratio, then we have $e^{2/c} - e^{1/c} = \phi^2 - \phi = 1$. Then, setting $\alpha = cqt < 5pt$, we get

$$\left(1 + \frac{2q}{\alpha} \right)^t - \left(1 + \frac{q}{\alpha} \right)^t = \left(1 + \frac{2}{ct} \right)^t - \left(1 + \frac{1}{ct} \right)^t < e^{2/c} - e^{1/c} = 1. \qquad (11.15)$$

t) t < eλ eλ for all t(参见此,按术语比较LHS和Taylor系列RHS的二元扩展).

我们已经证明,

$$\Pr[\ \mathrm{DC}(G|_\rho) \geq \Delta \mid F|_\rho \equiv 1, G_1|_\rho \not\equiv 1\ ] \ < \ \alpha^\Delta.$$

这证明了

$$\Pr[\ \mathrm{DC}(G|_\rho) \geq \Delta \mid F|_\rho \equiv 1\ ] \ < \ (5pt)^\Delta.$$

♣

现在我们证明莱玛11.14的航线更紧 请注意:

$$b_Y \;=\; \sum_{Y\subseteq Z\subseteq T} a_Z,$$

和M'obius倒置公式,

$$a_Y \;=\; \sum_{Y\subseteq Z\subseteq T}$$

替换P $_{Y\subseteq Z\subseteq T}(-1)^{|Z-Y|}b_Z$ for a_Y in (11.13), we have

$$\begin{aligned}
&\Pr[\ \mathrm{DC}(G|_\rho) \geq \Delta \mid F|_\rho \equiv 1, G_1|_\rho \not\equiv 1\ ] \\
&\quad\leq\ \sum_{Y\subseteq T}\sum_{Y\subseteq Z\subseteq T}(-1)^{|Z-Y|}b_Z\cdot(2^{|Y|}-1)\alpha^{\Delta-|Y|} \\
&\quad=\ \sum_{Z\subseteq T}\ \begin{matrix}\mathsf{bZ(\ -\ 1)\ +\ |}\\ \mathsf{\alpha\Delta X}\end{matrix}\qquad\left[\left(\frac{-2}{\alpha}\right)^{|Y|}-\left(\frac{-1}{\alpha}\right)^{|Y|}\right]
\end{aligned}$$

因此,

$$\begin{aligned}
&\Pr[\ \mathrm{DC}(G|_\rho) \geq \Delta \mid F|_\rho \equiv 1, G_1|_\rho \not\equiv 1\ ] \\
&\quad\leq\ \alpha^\Delta\sum_{Z\subseteq T}\ \begin{matrix}\mathsf{bZ(\ -\ 1)\ +}\\ \mathsf{|}\end{matrix}\ \left[\left(1-\frac{2}{\alpha}\right)^{|Z|}-\left(1-\frac{1}{\alpha}\right)^{|Z|}\right] \\
&\quad=\ \alpha^\Delta\sum_{Z\subseteq T}b_Z\left[\left(\frac{2}{\alpha}-1\right)^{|Z|}-\left(\frac{1}{\alpha}-1\right)^{|Z|}\right] \\
&\quad\leq\ \alpha^\Delta\sum_{Z\subseteq T}q^{|Z|}\left[\left(\frac{2}{\alpha}-1\right)^{|Z|}-\left(\frac{1}{\alpha}-1\right)^{|Z|}\right] \\
&\quad=\ \alpha^\Delta\left\{\left[1+q\left(\frac{2}{\alpha}-1\right)\right]^{|T|}-\left[1+q\left(\frac{1}{\alpha}-1\right)\right]^{|T|}\right\} \\
&\quad\leq\ \alpha^\Delta\left\{\left[1+q\left(\frac{2}{\alpha}-1\right)\right]^{t}-\left[1+q\left(\frac{1}{\alpha}-1\right)\right]^{t}\right\}
\end{aligned}$$

Now, we want to choose an appropriate α so that the quantity in curly braces above is ≤ 1. Set $q=\beta/t$. Then,

$$\left[1+\left(\frac{2}{\alpha}-1\right)\frac{\beta}{t}\right]^t-\left[1+\left(\frac{1}{\alpha}-1\right)\frac{\beta}{t}\right]^t < e^{\left(\frac{2}{\alpha}-1\right)\beta}-e^{\left(\frac{1}{\alpha}-1\right)\beta}$$

因此,我们希望维持e(2/α−1)β-e(1/α−1)β < 1. 解决这个方程式,我们得到

$$\alpha=\frac{\beta}{ln\left[\frac{1+\sqrt{1+4e^\beta}}{2}\right]}. \tag{11.16}$$

这完成了莱玛11.14的证据. ♣

Now we will show that for any $Z \subseteq T$, $b_Z = \Pr[\ \rho(Z) = * \mid F|_\rho \equiv 1 \wedge G_1|_\rho \not\equiv 1\ ] \leq q^{|Z|}$, where $q = p/(p + \frac{1-p}{2})$. Before we proceed, note that for $Z \subseteq T$, we have

$$\Pr[\ \rho(Z) = * \mid G_1|_\rho \not\equiv 1\ ] \ = \ q^{|Z|},$$

因为G1QQQ1意味着 $ $ $ 只在 T 上指定 或 0。

我们现在显示

$$b_Z \leq \Pr[\ \rho(Z) = * \mid G_1|_\rho \not\equiv 1\ ]$$

This is trivial if $Z = \emptyset$. Suppose $Z \neq \emptyset$.

Note that $b_Z \leq \max_{\rho'} \Pr[\ \rho(Z) = * \mid F|_\rho \equiv 1, G_1|_\rho \not\equiv 1, \rho|_{Z^c} = \rho'\ ]$.

Consider any fixed restriction ρ' on the complement of Z, $\rho' : Z^c \to \{0, 1, *\}$. Then, there is a unique extension of ρ' over Z, call it ρ^*, that satisfies $\rho^*(Z) = *$.

我们声称,

$$\Pr[\ \rho(Z) = * \mid F|_\rho \equiv 1, G_1|_\rho \not\equiv 1, \rho|_{Z^c} = \rho'\ ] \ \leq \ q^{|Z|}.$$

The event $\rho(Z) = *$ refers to the unique ρ^*, under the condition $\rho|_{Z^c} = \rho'$. If $F|_{\rho^*} \equiv 1$ then $F|_\rho \equiv 1$ for all extensions ρ of ρ' to Z. Hence $F|_\rho \equiv 1, G_1|_\rho \not\equiv 1, \rho|_{Z^c} = \rho'$ refers to the $2^{|Z|}$ assignments ρ, such that $\rho(i) \in \{0, *\}$ for all $i \in Z$. Out of these $2^{|Z|}$ assignments, $\rho(Z) = *$ occurs with probability $q^{|Z|}$. On the other hand, if $F|_{\rho^*} \not\equiv 1$, then the above conditional probability is 0 and the claim trivially holds. Hence, the claim is true and $b_Z \leq q^{|Z|}$. ♣

11.3.5 切换莱玛:改进的下界桩

我们现在在换Lemma时改进了界限 利用lemma,我们建立电路计算或近似等价大小的下限.

在最后的演讲中,我们证明了以下的lemma.

莱玛 11.15 让G成为t-An-Or公式 G1 QQG2 QQ. . . .. ∧. 对于任何β, 0 < β < t,

let ρ be a random p-restrıctıon, where $p = \frac{\rho}{t-\beta}$, *and let* $\alpha = \beta/ln\left\lfloor \frac{\cdot\ \cdot\ \cdot\ \cdot}{2} \right\rfloor$. . . .. 那为所有人 $\Delta \geq 0$, *we have*

$$\Pr[DC(G|\rho) \geq \Delta] \leq \alpha^\Delta.$$

In the above lemma, α is minimized when $\beta = \beta_0 \approx 0.227537$. In which case, $\alpha = \alpha_0 \approx 0.4164447$. Let $\gamma_0 = \beta_0/2 \approx 0.1137685$.

使用Lemma 1,我们将证明Lemma 2,一个更强大的版本切换lemma. 键是随机限制的下列复合属性. 注意p1限制后加p2限制后加单p1p2限制的效果相同. 此属性持有, 因为布尔变量在每一步骤都独立指定。

LEMMA 11.16 *Let G be a t-And-Or formula $G_1 \wedge G_2 \wedge \ldots \wedge G_w$, and let ρ be a random γ_0/t-restriction. Then for all $\Delta \geq 0$, we claim*

$$\Pr[DC(G|_\rho) \geq \Delta] \leq {\alpha_0}^\Delta$$

Proof. Let $q = \beta_0/t$ and $p = \frac{q}{2-q}$. Then $q = \frac{2p}{1+p} = \frac{p}{p+\frac{1}{2}}$**2是变量的概率**

在随机 p 选择中,以被指派 * 或 0 为条件, 指定。

我们已经证明,

$$\Pr[DC(G|_{\rho'}) \geq \Delta] \leq {\alpha_0}^\Delta,$$

其中QQ是一个随机的p限制。

Since $p = \frac{q}{2-q} > \frac{q}{2} = \frac{\gamma_0}{t}$, we know that $\gamma_0/(pt)$ is still a number less than 1. Because of the composite property of random restriction, a random γ_0/t-restriction ρ can be realized by first

pt -限制. 注意 DC(GQQ) < QQ) 表示存在一个深度小于QQ的决定树,它可以计算G. 因此,我们也能够用更严格的限制在不到QQ的深度内计算G。 这就是说,

$$\Pr[DC(G|_\rho) \geq \Delta] \leq \Pr[DC(G|_{\rho'}) \geq \Delta] \leq {\alpha_0}^\Delta.$$

11.3.6 电路下界

Consider general constant depth circuits. Denote by $C^d(s,t)$ the class of depth d circuits with bfi (the abbreviation of bottom fanin) $\leq t$, and the number of gates above the first level $\leq s$. Denote by $C^d(s)$ the class of depth d circuits without a bfi condition but with total size $\leq s$. It is clear that a circuit in $C^d(s)$ can be considered as a circuit in $C^{d+1}(s,1)$, by adding an extra layer of gates with fan-in 1.

下面的lemma通过切换lemma来证明. 在16号讲座中,我们讨论了如何使用切换lemma来证明电路下限. 我们在这里使用同样的想法,因此,我们只提供证据的草图。

LEMMA 11.17 *For all $C \in C^d(s, \gamma_0 n^{1/d})$, we have*

$$\Pr[DC(C|_\rho) \geq \gamma_0 n^{1/d}] \;\leq\; s \cdot \alpha_0^{\gamma_0 n^{1/d}} \;\approx\; s \cdot 2^{-0.143781 \cdot n^{1/d}},$$

在 $ 中随机 1/n 中 $\frac{d}{}\dot{\mathrm{d}}$ **-限制。**

Proof. Let $t = \gamma_0 n^{1/d}$ and $p = 1/n^{\frac{1}{d}}$. Let $C \in C^d(s,t)$. Denote the number of gates on each level as $s_1, s_2, \ldots, s_d$ from the bottom to the top (output level). Clearly, $s_d = 1$ and $\sum_{i=1}^{d-1} s_i = s$. Apply Lemma 2 repeatedly $d-1$ times, each time with a random p-restriction.

After applying the first random restriction, suppose all the s_1 many t-AND-OR circuits at level 1 have DC $\leq t$. Then, we can switch these into t-OR-AND circuits and merge first and second levels eliminating a level. This process also eliminates s_1 gates from the circuit. Nonetheless, for each gate at the bottom level, it is possible that we cannot switch. By

零点 因此,这一级的失败概率最多为s1 → αt
零点 累积每个级别发生故障的概率, 并结合Lemma 2,我们有

$$\Pr[DC(C|_\rho) \geq \gamma_0 t] \;\leq\; \sum_i \underset{\alpha t}{(-)} \quad = s \cdot \alpha_0^t = s \cdot \alpha_0^{\gamma_0 n^{1/d}} \;\approx\; s \cdot 2^{-0.143781 \cdot n^{1/d}}$$

最后,通过随机限制的复合属性,应用d−1随机p−限制具有与任意pd−1-限制相同的效果. 证已具足.

We can apply Lemma 3 to $C \in C^d(s)$ by first transforming them into $C \in C^{d+1}(s,1)$. But we can actually do slightly better by a more delicate technique. Here we omit the proof of the following better bound.

LEMMA 11.18 *For all $C \in C^d(s)$, we have*

$$\Pr[DC(C|_\rho) \geq \gamma_0 n^{1/d}] \;<\; s \cdot \alpha_0^{\gamma_0 n^{1/d}} \;\approx\; s \cdot 2^{-0.143781 \cdot n^{1/d}},$$

在 $ 中随机 α0/(2n) $^{\frac{d}{}}$d) 限制。

These results can be used to prove circuit lower bounds for the parity function. Consider any circuit C in $C^d(s, \gamma_0 n^{1/(d-1)})$. Apply $d-2$ rounds of random $1/n^{1/(d-1)}$-restrictions. With probability $> 1 - s \cdot 2^{-0.143781 \cdot n^{1/(d-1)}}$, we get a circuit in $C^2(1, \gamma_0 n^{1/(d-1)})$ after switching and merging. The process is equivalent to applying a single random $n^{(d-2)/(d-1)}$-restriction. Let N be the random variable for number of variables left (i.e., variables assigned *). Then, its expectation $E[N] = n^{1/(d-1)}$. By Chernoff bound we have,

$$\Pr[N \leq \gamma_0 n^{\frac{1}{d-1}}] < e^{-\frac{(1-\gamma_0)^2}{2} \cdot n^{\frac{1}{d-1}}} < e^{-0.3927 n^{\frac{1}{d-1}}}.$$

因此,如果S < 20.143781-n1/(d−1),则C在C2(1,γ0n1/(d−1)和N > γ0n1/(d−1)中都降低为电路的概率接近1。 假设我们开始的电路 在n变量上计算等值 然后,在应用随机限制后获得的电路计算其余N变量的等价. 显然,一个C2(1,t)电路不能计算 > t变量的等值(见讲座16)。 我们证明了以下的lemma。

LEMMA 11.19 *For all $C \in C^d(s, \gamma_0 n^{1/(d-1)})$, if C computes the parity function, then its size s must satisfy*

$$s \;\geq\; 2^{0.143781 \cdot n^{1/(d-1)}}.$$

Lemma 11.19可用于为普通电路获得下限(不含bfi). 同样,我们可以简单地将Cd(s)中的电路转换成Cd+1(s,1)中的电路,然后应用Lemma 11.19. 使用更直接更精细的分析,可以证明以下的lemma.

LEMMA 11.20 *For all $C \in C^d(s)$, if C computes the parity function, then its size s must satisfy*

$$s \geq 2^{0.143781 \cdot n^{\frac{1}{d-1}}}$$

11.3.7 近似性类型

现在,我们考虑 近似型的下限。 不可接近性是指具有某些限制的电路(如大小,深度和bfi)无法计算出半数以上可能案例的均价. 具体地说,对于对等函数,我们可以简单地猜测0和1作为函数值. 因此,很容易获得50%的成功。 我们想表明,我们无法大大改进。 决策树深度下限最适于得出不近似类型下限,而决策树视角正是出于这个原因提出的。 我们的目标是显示,当电路的大小低于一些下限时,电路会在所有可能的输入上造成非同步的50%的错误.

让C成为深度d电路. 请注意,在经过某些限制后,如果将C缩小为深度小于所剩变量数的决定树,那么对于ρ 0-1 扩展的精确一半,C同意对等. 这是因为在决定树的每一个叶子上,电路C是完全确定的.

Consider $\Pr[\ C(x_1, \ldots, x_n) = \oplus(x_1, \ldots, x_n)\ ]$, where $\oplus(x_1, \ldots, x_n)$ denotes the parity function, and the probability is over all 2^n assignments. This random restriction technique can be realized by first assigning any random restriction, followed by an unbiased 0-1 assignments for all the remaining variables. Let E_1 denote the event that after the random restriction, we end up with a decision tree of depth not more than t, and let E_2 denote the event that the number of variables N assigned to * is more than t. Then let $E = E_1 \wedge E_2$, and let $[\,C = \oplus\,]$ denote $[\,C(x_1, \ldots, x_n) = \oplus(x_1, \ldots, x_n)\,]$ for convenience. As we already pointed out, $\Pr[\,C = \oplus \,|\, E\,] = 1/2$ due to a property of the parity function.

在有条件的概率方面,我们有

$$\begin{aligned}
\Pr[C = \oplus] &= \Pr[E] \cdot \Pr[C = \oplus | E] + \Pr[\neg E] \cdot \Pr[C = \oplus | \neg E] \\
&= (1 - \Pr[\neg E]) \cdot \Pr[C = \oplus | E] + \Pr[\neg E] \cdot \Pr[C = \oplus | \neg E] \\
&= \Pr[C = \oplus | E] + \Pr[\neg E](Pr[C = \oplus | \neg E] - \Pr[C = \oplus | E]).
\end{aligned}$$

As we noted, $\Pr[\,C = \oplus \,|\, E\,] = 1/2$, and $\Pr[C = \oplus | \neg E] \leq 1$. Then substitute these 2 observations into the above equation,

$$\left|\Pr[C = \oplus] - \frac{1}{2}\right| \;\leq\; \frac{1}{2}\Pr[\neg E].$$

Since $\Pr[C = \oplus] + \Pr[C \neq \oplus] = 1$, we have

$$\Pr[\,C = \oplus\,] - \Pr[\,C \neq \oplus\,] = 2\left(\Pr[\,C = \oplus\,] - \frac{1}{2}\right),$$

并因此

$$|\Pr[\,C = \oplus\,] - \Pr[\,C \neq \oplus\,]| \;\leq\; \Pr[\neg E].$$

Now we specify the parameters of the random restrictions. Let $m = \gamma_0 n^{1/d}$. First consider any $C \in C^d(s, \gamma_0 m)$. Let $t = \gamma_0 m$ and apply Lemma 11.17. With a random $1/n^{(d-1)/d}$-restriction, we have

$$\Pr[\neg E_1] \leq s\alpha_0^t \approx s\dot{2}^{0.143781 \cdot m}.$$

Again by using the Chernoff bound, we estimate $\Pr[\neg E_2] = \Pr[N \leq \gamma_0 m]$ as follows.

$$\Pr[\neg E_2] \leq e^{-\frac{(1-\gamma_0)^2}{2}m} < e^{0.3927m}.$$

因此,Pr[¬E2]由Pr[¬E1]主导. 这一分析给出了以下约束.

LEMMA 11.21 *For all $C \in C^d(2^{0.07189n^{1/d}}, \gamma_0 n^{1/d})$, we have*

$$|\Pr[\,C = \oplus\,] - \Pr[\,C \neq \oplus\,]| \;\leq\; 2^{-0.07189n^{1/d}}.$$

以上lemma的直截了当的应用再次给普通电路(不含bfi)带来近似效果. 更仔细的分析导致以下lemma.

LEMMA 11.22 *For all circuits $C \in C^d(2^{0.07189n^{1/d}})$, we have*

$$|\Pr[C = \oplus] - \Pr[C \neq \oplus]| \leq 2^{0.07189n^{1/d}}.$$

第12章

其他结果

12.1 NP和P的相对分离

We construct an oracle relative to which NP $\neq$ P. And also another oracle relative to which NP = P.

定理12.1(巴克尔语,吉尔语,索洛瓦伊语) 有一些可计算语言A和B,例如:

1. $\mathrm{NP}^A = \mathrm{P}^A$.
2. $\mathrm{NP}^B \neq \mathrm{P}^B$.

证据(第一部分):这很容易显示。 选择 A 成为任何 SPACE- complete 语言。 然后,我们...

$$\mathrm{NP}^A \subseteq \mathrm{PSPACE}^A \subseteq \mathrm{PSPACE} \subseteq \mathrm{P}^A$$

第一个包含是因为,我们可以通过运行NP机的所有可能的猜测来模拟PSPACE中的NP机. 第二项是A在PSPACE,因此我们可以在PSPACE中解决A的询问,而不问oracle. 由于A是PSPACE-Hard,任何PSPACE语言都可以简化为A. 所以,最后的包含维持。

证明(第二部分):对于任何语言B来说,让TB成为语言

$$T_B = \{1^n | \exists x, |x| = n, x \in L\}$$

We first show that for any B, $T_B \in \mathrm{NP}^B$. Given 1^n as input, simply guess a string x of length n, then, use the oracle to verify whether $x \in B$. If so accept, else reject.

We construct a language B such that $T_B \notin P^B$. The basic idea is simple. Intuitively, any machine that decides T_B, given 1^n as input, has to find out whether there is some string of

长度n在B。 但有2N这样的串。 没有多名机器,可以向oracleB询问很多问题.

让N1,N2,。 . . .. 成为所有决定性多名时间Turing machine的列表. 让尼的运行时间与尼为界. 这种列举可通过以下方式实现:首先列举所有决定性TMs,然后在ITh TM中添加一个“点”大小的ni。 我们要把所有这些尼对齐 我们分阶段建造B型 我们可以想到B最初是空的,我们在每个阶段都加入一些弦. 在kth阶段,我们确保,Nk不会决定肺结核.

我们描述Kth阶段。 我们已经模拟了N1,N2 . . .. Nk－1在前(k－1)阶段,本会回答许多oracle询问. 让 n0 成为我们模拟中这些 k - 1 机器所询问的任何查询的最大长度。 到K -1阶段结束时,我们将会建造B,长度可达零。 在kth阶段,我们确保Nk不会决定肺结核. 现在,选择n,这样n > n0和2n > nk,模拟Nk(1n). 捕捉量处于第二个条件. Nk是一种有时间约束nk的机器. 它最多可以询问nk. 如果询问长度小于n0的查询,我们已经知道答案-(我们已经将 B 构建到n0 长度)。 如果它问到长度超过0的询问,我们只是回答“不”。 (即,不要用B线) 现在该是诊断的时候了。 在模拟结束时,假设Nk接受1n. 我们保持所有长度n的线 的B。 假设Nk拒绝1n。 它最多可以问n的nk询问(我们答复了所有“否”的问题)。 作为 2n > nk, 长度 n 至少有一个字符串 y , 这不是Nk 的查询。 我们把这个字符串加到B 无论是哪种情况,Nk都没有作出正确的决定。

A formal algorithm to enumerate B is as follows. Think of B being empty initially and we add strings to it in stages. At any stage k, Q_k represents the set of queries asked by N_k in that stage. And n_0 is set to 0 initially. Run the following procedure for $k = 1, 2 \ldots$.

1. 国家 选择 n, 使用 n > n0 和 2n > nk。 设置 Qk 为空。

2. 国家 模拟 Nk(1n) 的 nk 步骤。 如果q是Nk的查询,

   - 在Qk中添加q。
   - If $|q| \leq n_0$ then answer $B(x)$ else answer "no".

3 (中文(简体)). 模拟结束时:

   ^ 如果nk拒绝 1n : — 选择一个长度为 y 的字符串 y, 不在 Qk 中, 并添加到 B 中。

   - 如果nk接受1n:-不做任何事情。

4 , 4 , 4 , 4 , 4 , 4 , 4 让q成为Qk中最长的字符串. 设置 n0 {max{n0,} {x|}}。

Bibliography

[1] D. Du and K. Ko. *Theory of Computational Complexity*. John Wiley and Sons, 2000.

[2] M. Garey and D. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

[3] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[4] M. Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 1997.