Last revised 4/29/2010

In this lecture we prove the Karp-Lipton theorem that if all NP problems have polynomial size circuits then the polynomial hierarchy collapses. A nice application is a theorem of Kannan, showing that, for every k, there are languages in Σ_2 requiring circuits of size $\Omega(n^k)$. The next result we wish to prove is that all approximate combinatorial counting problem can be solved within the polynomial hierarchy. Before introducing counting problems and the hashing techniques that will yield this result, we prove the Valiant-Vazirani theorem that solving SAT on instances with exactly one satisfying assignment is as hard as solving SAT in general.

1 The Karp-Lipton Theorem

Theorem 1 (Karp-Lipton) *If* $\mathbf{NP} \subseteq \mathbf{SIZE}(n^{O(1)})$ *then* $\Sigma_2 = \Pi_2$ *and therefore the polynomial hierarchy would collapse to its second level.*

Before proving the above theorem, we first show a result that contains some of the ideas in the proof of the Karp-Lipton theorem.

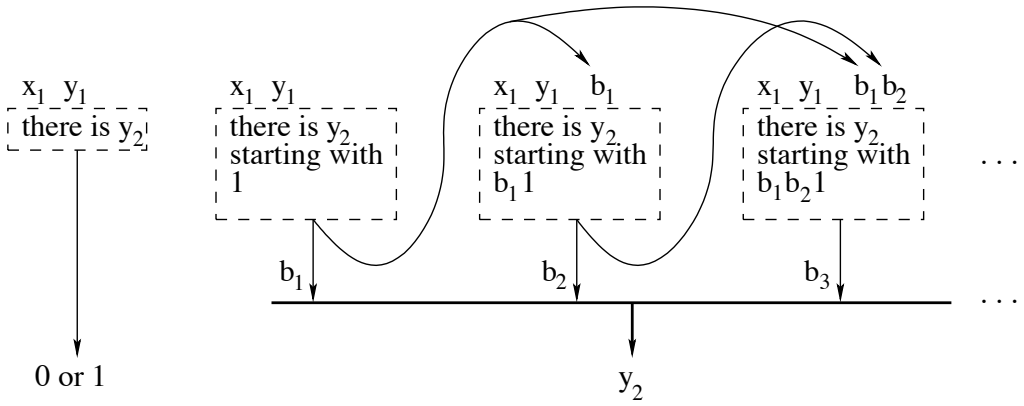
Lemma 2 *If* $\mathbf{NP} \subseteq \mathbf{SIZE}(n^{O(1)})$ *then for every polynomial time computable* $F(\cdot,\cdot)$ *and every polynomial* $p(\cdot)$*, there is a family of polynomial size circuits such that*

$$C_{|x|}(x) = \begin{cases} y : F(x,y) = 1 & \text{if such a } y \text{ exists} \\ \text{a sequence of zeroes} & \text{if otherwise} \end{cases}$$

PROOF: We define the circuits $C_n^1, \ldots, C_n^{p(n)}$ as follows:

C_n^i, on input x and bits $b_1, \ldots, b_{i-1}$, outputs 1 if and only if there is a satisfying assignment for $F(x,y) = 1$ where $y_1 = b_1, \ldots, y_{i-1} = b_{i-1}, y_i = 1$.

Also, each circuit realizes an **NP** computation, and so it can be built of polynomial size. Consider now the sequence $b_1 = C_n^1(x)$, $b_2 = C_n^2(b_1, x)$, ..., $b_{p(n)} = C_n^{p(n)}(b_1, \ldots, b_{p(n)-1}, x)$, as shown in the following picture:



The reader should be able to convince himself that this is a satisfying assignment for $F(x, y) = 1$ if it is satisfiable, and a sequence of zeroes otherwise. □

We now prove the Karp-Lipton theorem.

PROOF: [Of Theorem 1] We will show that if $\mathbf{NP} \subseteq \mathbf{SIZE}(n^{O(1)})$ then $\Pi_2 \subseteq \Sigma_2$. By a result in a previous lecture, this implies that $\forall k \geq 2 \;\; \Sigma_k = \Sigma_2$.

Let $L \in \Pi_2$, then there is a polynomial $p(\cdot)$ and a polynomial-time computable $F(\cdot)$ such that

$$x \in L \leftrightarrow \forall y_1.|y_1| \leq p(|x|)\exists y_2.|y_2| \leq p(|x|).F(x, y_1, y_2) = 1$$

By using Lemma 2, we can show that, for every n, there is a circuit C_n of size polynomial in n such that for every x of length n and every y_1, $|y_1| \leq p(|x|)$,

$$\exists y_2.|y_2| \leq p(|x|) \wedge F(x, y_1, y_2) = 1 \text{ if and only if } F(x, y_1, C_n(x, y_1)) = 1$$

Let $q(n)$ be a polynomial upper bound to the size of C_n.

So now we have that for inputs x of length n,

$$x \in L \leftrightarrow \exists C.|C| \leq q(n).\forall y_1.|y_1| \leq p(n).F(x, y_1, C(x, y_1)) = 1$$

which shows that L is in Σ_2. □

2 Kannan's Theorem

Although it is open to prove that the polynomial hierarchy is not contained in $\mathbf{P}/poly$, it is not hard to prove the following result.

Theorem 3 *For every polynomial $p()$, there is a language $L \in \Sigma_3$ such that $L \notin$ **SIZE**$(p(n))$.*

Note that Theorem 3 is not saying that $\Sigma_3 \not\subseteq \mathbf{P}/poly$, because for that to be true we would have to be able to construct a single language L such that for every polynomial p we have $L \notin$ **SIZE**$(p(n))$, instead of constructing a different language for each polynomial. (This is an important difference: the time hierarchy theorem gives us, for every polynomial $p()$, a language $L \in \mathbf{P}$ such that $L \notin$ **DTIME**$(p(n))$, but this doesn't mean that $\mathbf{P} \neq \mathbf{P}$.)

Kannan observed the following consequence of Theorem 3 and of the Karp-Lipton theorem.

Theorem 4 *For every polynomial $p()$, there is a language $L \in \Sigma_2$ such that $L \notin$ **SIZE**$(p(n))$.*

Proof: We consider two cases:

- if $3SAT \notin$ **SIZE**$(p(n))$; then we are done because $3SAT \in \mathbf{NP} \subseteq \Sigma_2$.
- if $3SAT \in$ **SIZE**$(p(n))$, then $\mathbf{NP} \subseteq \mathbf{P}/poly$, so by the Karp-Lipton theorem we have $\Sigma_3 = \Sigma_2$, and the language $L \in \Sigma_3 -$ **SIZE**$(p(n))$ given by Theorem 3 is in Σ_2.

□

3 The Valiant-Vazirani Theorem

In this section we begin to discuss the proof of the following result, due to Valiant and Vazirani: suppose there is an algorithm for the satisfiability problem that always find a satisfying assignment for formulae that have exactly one satisfiable assignment (and behaves arbitrarily on other instances): then we can get an **RP** algorithm for the general satisfiability problem, and so $\mathbf{NP} = \mathbf{RP}$.

We prove the result by presenting a randomized reduction that given in input a CNF formula ϕ produces in output a polynomial number of formulae $\psi_0, \ldots, \psi_n$. If ϕ is satisfiable, then (with high probability) at least one of the ψ_i is satisfiable and has exactly one satisfying assignment; if ϕ is not satisfiable, then (with probability one) all ψ_i are unsatisfiable.

The idea for the reduction is the following. Suppose ϕ is a satisfiable formula with n variables that has about 2^k satisfying assignments, and let $h : \{0,1\}^n \to \{0,1\}^k$ be a

hash function picked from a family of pairwise independent hash functions: then the average number of assignments x such that $\phi(x)$ is true and $h(x) = (0, \ldots, 0)$ is about one. Indeed, we can prove formally that with constant probability there is exactly one such assignment,[1] and that there is CNF formula ψ (easily constructed from ϕ and h) that is satisfied precisely by that assignment. By doing the above construction for values of k ranging from 0 to n, we obtain the desired reduction. Details follow.

Definition 5 *Let H be a family of functions of the form $h : \{0,1\}^n \to \{0,1\}^m$. We say that H is a family of pair-wise independent hash functions if for every two different inputs $x, y \in \{0,1\}^n$ and for every two possible outputs $a, b \in \{0,1\}^m$ we have*

$$\mathbb{P}_{h \in H}[h(x) = a \wedge h(y) = b] = \frac{1}{2^{2m}}$$

Another way to look at the definition is that for every $x \neq y$, when we pick h at random then the random variables $h(x)$ and $h(y)$ are independent and uniformly distributed. In particular, for every $x \neq y$ and for every a, b we have $\mathbb{P}_h[h(x) = a | h(y) = b] = \mathbb{P}_h[h(x) = a]$.

For m vectors $a_1, \ldots, a_m \in \{0,1\}^m$ and m bits $b_1, \ldots, b_m$, define $h_{a_1,\ldots,a_m,b_1,\ldots,b_m} : \{0,1\}^n \to \{0,1\}^m$ as $h_{\mathbf{a},b}(x) = (a_1 \cdot x + b_1, \ldots, a_m \cdot x + b_m)$ where $a \cdot x := \Sigma_i a_i x_i \bmod 2$, and let H_{AFF} be the family of functions defined this way. Then it is not hard to see that H_{AFF} is a family of pairwise independent hash functions.

[1] For technical reasons, it will be easier to prove that this is the case when picking a hash function $h : \{0,1\}^n \to \{0,1\}^{k+2}$.