The goal of the next couple of lectures will be to prove Toda’s theorem [3], $\text{PH} \subseteq \text{P}^{\text{#P}}$, which we used to prove the IK theorem.

Define $\oplus \text{P}$ as the complexity class of decision problems solvable by an NP machine, where the acceptance condition is that the number of accepting computation paths is odd. An example of a $\oplus \text{P}$ problem is “given a graph, does it have an odd number of perfect matchings?”. It can be viewed as finding the least significant bit of the answer to the corresponding $\text{#P}$ problem. In this lecture we are going to prove the following lemma, which comprises the first part of Toda’s proof.

**Lemma 1.** $\text{PH} \subseteq \text{BPP}^{\oplus \text{P}}$.

We will follow Fortnow’s proof [1], but we will need some preliminaries first.

## 1 The isolation lemma and UniqueCLIQUE

The *isolation lemma*, due to Mulmuley, Vazirani and Vazirani, gives a randomized algorithm to reduce the number of solutions to one, given such a solution exists.

**Definition 2.** Let $X$ be a set of $n$ elements, and let $\mathcal{F}$ be a family of subsets of $X$. Assign a weight $w(x)$ to each element, and define the weight of a set $E \in \mathcal{F}$ as $w(E) = \sum_{x \in E} w(x)$. If $\min_{E \in \mathcal{F}} w(E)$ is achieved by a unique $E \in \mathcal{F}$, we say $w$ is isolating for $\mathcal{F}$.

**Lemma 3** ([2]). Let $X$ be a set of $n$ elements, and let $\mathcal{F}$ be a family of subsets of $X$. Let $w : X \to [N]$ be a random function, each $w(x)$ is chosen independently and uniformly. Then,

$$\Pr_{w}[w \text{ is isolating for } \mathcal{F}] \geq 1 - \frac{n}{N}.$$ 

**Proof.** Draw $w$ uniformly at random. For an element $x \in X$, set

$$\alpha(x) = \min_{E \in \mathcal{F}, x \notin E} w(E) - \min_{E \in \mathcal{F}, x \in E} w(E \setminus \{x\}).$$

Evaluation of $\alpha(x)$ does not require knowledge of $w(x)$, so we have that

$$\Pr_{w}[w(x) = \alpha(x)] = \frac{1}{N}$$

and

$$\Pr_{w}[\exists x \in X, w(x) = \alpha(x)] \leq \frac{n}{N}.$$ 

But if $w$ induces two minimal sets $A, B \in \mathcal{F}$ and $x \in A \setminus B$ then

$$\min_{E \in \mathcal{F}, x \notin E} w(E) = w(B)$$

and

$$\min_{E \in \mathcal{F}, x \in E} w(E \setminus \{x\}) = w(A) - w(x),$$

so $\alpha(x) = w(B) - w(A) + w(x) = w(x)$. Thus, if $w$ is not isolating for $\mathcal{F}$ then $w(x) = \alpha(x)$ for some $x \in X$, and we have already seen that the last event can happen with probability at most $\frac{n}{N}$. $\square$
The isolation lemma gives a probabilistic reduction from CLIQUE to UniqueCLIQUE which we will now see. As the reduction from CLIQUE to SAT preserves the number of accepting witnesses, a probabilistic reduction from SAT to UniqueSAT follows. A probabilistic reduction to UniqueSAT was first given by Valiant and Vazirani [4] using another technique.

**Theorem 4.** There is a probabilistic polynomial-time procedure that, given a graph \( G \) and an integer \( k \), outputs \( G' \) and \( k' \) such that:

- If \( G \) has no clique of size \( k \) then \( G' \) has no clique of size \( k' \).
- If \( G \) has a clique of size \( k \) then, with a non-negligible probability, \( G' \) has exactly one clique of size \( k' \).

**Proof.** Given an input \( \langle G = (V, E), k \rangle \), let \(|V| = n\). The algorithm choose \( w : V \rightarrow [2^n] \) uniformly at random. By the isolation lemma, with probability at least \( \frac{1}{2} \), the clique of maximal weight will be unique (it is easy to see that the proof also works for the maximal weight).

Let \( G' \) be the following graph: For every vertex \( v \in V \), construct a clique of size \( 2nk + w(v) \). For every edge \((u, v) \in E\), connect the \( u \)-clique to the \( v \)-clique in \( G' \) (every vertex to every vertex).

Next, choose a random integer \( r \in [2nk] \) and return \( \langle G', k' = 2nk^2 + r \rangle \). Now:

- If \( \langle G, k \rangle \notin \text{CLIQUE} \) then the size of the smallest clique in \( G' \) is at most \((k-1) \cdot (2nk + 2n) < 2nk^2 \) so \( \langle G', k' \rangle \notin \text{UniqueCLIQUE} \).
- If \( \langle G, k \rangle \in \text{CLIQUE} \) then with probability at least \( \frac{1}{2} \) there is a unique clique \( C \subseteq V \) of size \( k \) with a maximal \( w(C) \). Assume this is indeed the case.

The size of the clique in \( G' \) corresponding to \( C \) is \( 2nk^2 + w(C) \) and note that \( 2nk^2 + 1 \leq 2nk^2 + w(C) \leq 2nk^2 + 2nk \). For any other \( k \)-clique \( C' \subseteq C \), the corresponding clique in \( G' \) has weight \( 2nk^2 + w(C') < 2nk^2 + w(C) \).

We already saw that a clique of size smaller than \( k \) in \( G \) corresponds to a clique of size smaller than \( 2nk^2 \) in \( G' \). A \((k+1)\)-clique in \( G \) corresponds to a clique of size larger than \( 2nk(k+1) + k + 1 > k' \).

It follows that for the correct \( r = w(C) \) we will have a unique clique of size \( k' \). Hence, the probability that \( \langle G, k \rangle \in \text{UniqueCLIQUE} \) is at least \( \frac{1}{2nk} \).

\[ \square \]

2 Preliminary results

We first show:

**Theorem 5.** \( \oplus \text{P} \oplus \text{P} = \oplus \text{P} \).

**Proof.** Let \( L \in \oplus \text{P} \oplus \text{P} \), equipped with an accepting \( \text{NP} \) machine \( M \) making oracle calls to some \( \oplus \text{P} \)-complete language \( A \) having an accepting \( \text{NP} \) machine \( M_A \). We will show an \( \text{NP} \) machine \( N \) accepting \( L \) with no oracle calls. That is, \( x \in L \) iff the number of accepting path of \( N(x) \) is odd. \( N \) on an input \( x \) behaves as follows:
1. \( N \) guesses a computation path \( w \) of \( M \) on input \( x \), which includes possible oracle answers to the query strings appearing in \( w \).

2. If \( w \) is a rejecting path of \( M \) on \( x \) then \( N \) enters a rejecting step. Otherwise, it goes to the next step.

3. Let \( y_1, \ldots, y_m \) be all the query strings which appear in \( w \) and whose corresponding oracle answers in \( w \) are Yes and likewise let \( z_1, \ldots, z_\ell \) be all the query strings which appear in \( w \) and whose corresponding oracle answers in \( w \) are No. Then, \( N \) simulates \( M_A \) successively for each \( y_i \) and \( z_i \) in the following manner:

   (a) For each \( y_i \), it simply simulates \( M_A \). If \( M_A \) enters a rejecting state then so does \( N \). Otherwise, it proceeds to the next simulation.

   (b) For each \( z_i \), it nondeterministically selects one of the following processes:

   - \( N \) goes to the next simulation.
   - \( N \) simulates \( M_A \) on \( z_i \). If \( M_A \) enters a rejecting state, then so does \( N \). Otherwise, it goes to the next simulation.

4. \( N \) enters an accepting state.

For the correctness, we classify all possible accepting paths of \( M \) on \( x \) into two groups, one of which consists of accepting paths with the correct oracle answers to \( A \) and the remaining ones (that contain at least one inconsistent oracle call).

From the definition of \( N \) we can see that:

- Every accepting path in the first group is followed by an odd number of accepting paths in steps 3 and 4 since on the \( y \)-s we always have an odd number of accepting paths, and on the \( z \)-s we always have an odd number of accepting paths.

- Every accepting path in the second group is followed by an even number of accepting paths in steps 3 and 4. To see this, observe that if we do not err on any of the \( y \)-s (odd number of accepting paths) we must err on at least one \( z \), leading to an even number of accepting paths in the \( z \)-s, for a total of even number of accepting paths. If we do err on one of the \( y \)-s, we have an even number of accepting paths and a total of even number of accepting paths, regardless of how we act on the \( z \)-s.

Having established that, we have that if \( x \in L \) then the number of accepting paths in the first group is odd, so the number of accepting paths of \( N \) is odd as well (odd \( \times \) odd + even = odd), and similarly if \( x \notin L \) then the number of accepting paths in the first group is even (even \( \times \) odd + odd = even), so the number of accepting paths of \( N \) is even – as desired.

\[ \square \]

**Theorem 6.** If \( \text{NP} \subseteq \text{BPP} \) then \( \text{PH} \subseteq \text{BPP} \).

**Proof.** As an exercise.

\[ \square \]

As a corollary, we have:

**Lemma 7.** \( \text{NP} \subseteq \text{BPP}^{\text{NP}} \).
Proof. It is sufficient to show that CLIQUE ∈ BPP⊕P. Given an input ⟨G, k⟩, use the probabilistic algorithm from Theorem 4 to produce G’ and k’ and accept iff the NP machine for CLIQUE on input ⟨G’, k’⟩ has an odd number of accepting paths (using the ⊕P oracle).

If ⟨G, k⟩ ∉ CLIQUE then there will always be zero accepting paths and we will always reject. If ⟨G, k⟩ ∈ CLIQUE then with non-negligible probability there will be exactly one accepting path and we will accept. □

3 A proof of Toda’s first lemma

When we relativize a class like BPP⊕P to an oracle A, both the BPP and the ⊕P machines should have access to the oracle A. The BPP machine can make its queries to A via the ⊕P A oracle so we have (BPP⊕P)A = BPP(⊕P A), which we will write simply as BPP⊕P A.

We are now ready to prove that PH ⊆ BPP⊕P.

Proof. Lemma 7 relativizes, so we have

NP⊕P ⊆ BPP⊕P⊕P.

By Theorem 5,

NP⊕P ⊆ BPP⊕P.

Theorem 6 relativizes as well, so NP⊕P ⊆ BPP⊕P implies

PH⊕P ⊆ BPP⊕P.

However, PH ⊆ PH⊕P so we finally have PH ⊆ BPP⊕P and we are done. □

References


