4. The Isolation Technique

Brother: *And the Lord spake, saying, "First shalt thou take out the Holy Pin. Then, shalt thou count to three, no more, no less. Three shalt be the number thou shalt count, and the number of the counting shalt be three. Four shalt thou not count, nor either count thou two, excepting that thou then proceed to three. Five is right out. Once the number three, being the third number, be reached, then lobbest thou thy Holy Hand Grenade of Antioch towards thy foe, who being naughty in my sight, shall snuff it."*
Maynard: *Amen.*
All: *Amen.*
Arthur: *Right! One... two... five!*

—Monty Python and the Holy Grail

Counting is cumbersome and sometimes painful. Studying NP would indeed be far simpler if all NP languages were recognized by NP machines having at most one accepting computation path, that is, if NP = UP. The question of whether NP = UP is a nagging open issue in complexity theory. There is evidence that standard proof techniques can settle this question neither affirmatively nor negatively. However, surprisingly, with the aid of randomness we will relate NP to the problem of detecting unique solutions. In particular, we can reduce, with high probability, the entire collection of accepting computation paths of an NP machine to a single path, provided that initially there is at least one accepting computation path. We call such a reduction method an isolation technique.

In this chapter we present one such technique. Based on this technique, we prove two surprising results relating NP and NL to counting complexity classes: PP is polynomial-time Turing hard for the polynomial hierarchy, and NL and UL are equal in the presence of polynomially length-bounded advice functions.

The organization of this chapter is as follows. In Sect. 4.1, we present the isolation technique, and show that NP is "randomized reducible" to the problem of detecting unique solutions. More precisely, for each language L in NP, there exist a randomized polynomial-time algorithm $\mathcal{F}$ and an NP-decision problem A, such that for every string x, if x is a member of L, then with high probability the output of $\mathcal{F}$ on input x is an instance of A with a

unique solution, and if x is not a member of L, then with probability $\frac{1}{|x|^{O(1)}}$ the output of $\mathcal{F}$ on x is an instance of A with zero solutions or more than one solution.

In Sect. 4.2, we apply the isolation technique for NP to prove Toda's Theorem, PH $\subseteq$ P^{PP}, and we also establish a well-known extension of Toda's Theorem. In Sect. 4.3, we prove that NL/poly = UL/poly.

4.1 GEM: Isolating a Unique Solution

The isolation technique we use in this chapter is often called the Isolation Lemma.

4.1.1 The Isolation Lemma

A *weight function* over a finite set $\mathcal{U}$ is a mapping from $\mathcal{U}$ to the set of positive integers. We naturally extend any weight function over $\mathcal{U}$ to one on the power set $2^{\mathcal{U}}$ as follows. For each $S \subseteq \mathcal{U}$, the weight of S with respect to a weight function W, denoted by $W(S)$, is $\sum_{x \in S} W(x)$. Let $\mathcal{F}$ be a nonempty family of nonempty subsets of $\mathcal{U}$. Call a weight function W *good for* $\mathcal{F}$ if there is exactly one minimum-weight set in $\mathcal{F}$ with respect to W. Call W *bad for* $\mathcal{F}$ otherwise.

Lemma 4.1 (The Isolation Lemma) *Let $\mathcal{U}$ be a finite set. Let $\mathcal{F}_1, \ldots, \mathcal{F}_m$ be families of nonempty subsets over $\mathcal{U}$, let $D = ||\mathcal{U}||$, let $R > mD$, and let $\mathcal{Z}$ be the set of all weight functions whose weights are at most R. Let α, $0 < \alpha < 1$, be such that $\alpha > \frac{mD}{R}$. Then more than $(1-\alpha)||\mathcal{Z}||$ functions in $\mathcal{Z}$ are good for all of $\mathcal{F}_1, \ldots, \mathcal{F}_m$.*

Proof Let $\mathcal{F}$ be one family. For a weight function $W \in \mathcal{Z}$, let $MinWeight_W$ denote the minimum weight of $\mathcal{F}$ with respect to W, i.e., $MinWeight_W = \min\{W(S) \,|\, S \in \mathcal{F}\}$, and let $MinWeightSet_W$ denote the set of all minimum-weight sets of $\mathcal{F}$ with respect to W, i.e., $MinWeightSet_W = \{S \in \mathcal{F} \,|\, W(S) = MinWeight_W\}$. For $x \in \mathcal{U}$, we say that the minimum-weight sets of $\mathcal{F}$ with respect to W are unambiguous about inclusion of x if there exist some $S, S' \in MinWeightSet_W$ such that $x \in (S \setminus S') \bigcup (S' \setminus S)$.

Recall that a weight function $W \in \mathcal{Z}$ is bad for $\mathcal{F}$ if $||MinWeightSet_W|| \geq 2$. Suppose that W is bad for $\mathcal{F}$. Let S and S' be two distinct members of $MinWeightSet_W$. Since $S \neq S'$ there exists some $x \in \mathcal{U}$ such that x belongs to the symmetric difference of S and S', i.e., $(S \setminus S') \bigcup (S' \setminus S)$. Thus, the minimum-weight sets of $\mathcal{F}$ with respect to W are ambiguous about some $x \in \mathcal{U}$. Conversely, if the minimum-weight sets of $\mathcal{F}$ with respect to W are ambiguous about some $x \in \mathcal{U}$, then there is more than one minimum-weight sets of $\mathcal{F}$ with respect to W, so W is bad. Thus, W is bad if and only if there

is some $x \in \mathcal{U}$ such that the minimum-weight sets of $\mathcal{F}$ with respect to W are ambiguous about inclusion of x.

Let $x \in \mathcal{U}$ be fixed. We count the number of weight functions $W \in \mathcal{Z}$ such that the minimum-weight sets of $\mathcal{F}$ with respect to W are ambiguous about inclusion of x. Let $y_1, \ldots, y_{D-1}$ be an enumeration of $\mathcal{U} - \{x\}$ and $v_1, \ldots, v_{D-1} \in \{1, \ldots, R\}$. Let $\mathcal{A}$ be the set of all weight functions W such that for all i, $1 \leq i \leq D-1$, $W(y_i) = v_i$. Suppose that there is a weight function W in $\mathcal{A}$ such that the minimum-weight sets of $\mathcal{F}$ with respect to W are ambiguous about inclusion of x. Let W' be an arbitrary element in $\mathcal{A} \setminus \{W\}$ and $\delta = W'(x) - W(x)$. We claim that the minimum-weight sets of $\mathcal{F}$ with respect to W' are unambiguous about inclusion of x. To see why, first suppose that $\delta > 0$. Then, for all $S \in \mathcal{F}$, $W'(S) = W(S) + \delta$ if $x \in S$ and $W'(S) = W(S)$ otherwise. In particular, for all $S \in \mathit{MinWeightSet}_W$, $W'(S) = W(S) + \delta$ if $x \in S$ and $W'(S) = W(S)$ otherwise. This implies that $\mathit{MinWeight}_{W'} = \mathit{MinWeight}_W$ and $\mathit{MinWeightSet}_{W'} = \{S \in \mathit{MinWeightSet}_W \mid x \notin S\}$. Next suppose that $\delta < 0$. Then, for all $S \in \mathcal{F}$, $W'(S) = W(S) - |\delta|$ if $x \in S$ and $W'(S) = W(S)$ otherwise. In particular, for all $S \in \mathit{MinWeightSet}_W$, $W'(S) = W(S) - |\delta|$ if $x \in S$ and $W'(S) = W(S)$ otherwise. This implies that $\mathit{MinWeight}_{W'} = \mathit{MinWeight}_W - |\delta|$ and $\mathit{MinWeightSet}_{W'} = \{S \in \mathit{MinWeightSet}_W \mid x \in S\}$. Thus, if $\delta > 0$ then all minimum-weight sets of $\mathcal{F}$ with respect to W' contain s, and if $\delta < 0$ then no minimum-weight sets of $\mathcal{F}$ with respect to W' contain s. Hence, for all $W' \in \mathcal{A} \setminus \{W\}$ are the minimum-weight sets of $\mathcal{F}$ with respect to W' are unambiguous about inclusion of x. This implies that there is at most one weight function $W \in \mathcal{A}$ such that the minimum-weight sets of $\mathcal{F}$ with respect to W are ambiguous about inclusion of x. For each i, $1 \leq i \leq D-1$, there are R choices for v_i. So, there are at most R^{D-1} weight functions $W \in \mathcal{Z}$ such that the minimum-weight sets of $\mathcal{F}$ with respect to W are ambiguous about inclusion of x. There are R^D weight functions in $\mathcal{Z}$, there are m choices for $\mathcal{F}$, and there are D choices for x. Thus, the proportion of $\{W \in \mathcal{Z} \mid$ for some i, $1 \leq i \leq m$, W is bad for $\mathcal{F}_i\}$ is at most $\frac{mDR^{D-1}}{R^D} = \frac{mD}{R} < \alpha$. So, the proportion of $\{W \in \mathcal{Z} \mid$ for all i, $1 \leq i \leq m$, W is good for $\mathcal{F}_i\}$ is more than $1 - \alpha$. ❑

4.1.2 NP Is Randomized Reducible to US

US is the class of languages L for which there exists a nondeterministic polynomial-time Turing machine N such that, for every $x \in \Sigma^*$, $x \in L$ if and only if N on input x has exactly one accepting computation path (see Sect. A.9). USAT is the set of all boolean formulas having exactly one satisfying assignment and that USAT is complete for US under polynomial-time many-one reductions (see Sect. A.9).

We say that a language A is randomized reducible to a language B, denoted by $A \leq_{\text{randomized}} B$, if there exist a probabilistic polynomial-time Turing machine M and a polynomial p such that, for every $x \in \Sigma^*$,

- if $x \in A$, then the probability that M on input x outputs a member of B is at least $\frac{1}{p(|x|)}$, and
- if $x \in A$, then the probability that M on input x outputs a member of B is zero.

Using the Isolation Lemma, one can show that every language in NP is randomized reducible to USAT.

Theorem 4.2 $(\forall L \in \text{NP})[L \leq_{\text{randomized}} \text{USAT}]$.

To prove this theorem and other results in this chapter, we will use a pairing function with a special property regarding the encoding length. For binary strings $x, y, \ldots, z$, we write $x\#y\# \cdots \#z$ to denote the string constructed from this expression by replacing each occurrence of 0, 1, and # by 00, 01, and 11, respectively. More precisely, the encoding is the binary string of the form

$$0x_1 \cdots 0x_{|x|}110y_1 \cdots 0y_{|y|}11 \cdots\cdots 0z_1 \cdots 0z_{|z|}.$$

Note that this pairing function satisfies the following conditions:

- If $x_1, x_2, \ldots, x_k$ and $y_1, y_2, \ldots, y_k$ satisfy $|x_1| + |x_2| + \cdots + |x_k| = |y_1| + |y_2| + \cdots + |y_k|$, then $|x_1\#x_2\# \ldots \#x_k| = |y_1\#y_2\# \ldots \#y_k|$.
- For every $x, y, \ldots, z \in \Sigma^*$, we can recover $x, y, \ldots, z$ from $x\#y\# \cdots \#z$ in time polynomial in $|x\#y\# \cdots \#z|$.

Proof of Theorem 4.2 Let L be a language in NP. Let p be a polynomial and A a language in P such that, for all $x \in \Sigma^*$,

$$x \in L \iff (\exists y \in \Sigma^{p(|x|)})\,[\langle x, y\rangle \in A].$$

We may assume that for all $n \geq 0$, $p(n) \geq 1$, and that for all x, $\langle x, 0^{p(|x|)}\rangle \notin A$. For each $n \geq 1$, let $\mu(n)$ be the smallest power of 2 that is greater than or equal to $p(n)$. Define

$$A' = \{\langle x, y\rangle \,\big|\, |y| = \mu(|x|) \wedge (\exists u, v)[|u| = p(|x|) \wedge uv = y \wedge \langle x, y\rangle \in A]\}.$$

Then $A' \in \text{P}$ and for every $x \in \Sigma^*$,

$$x \in L \iff (\exists y \in \Sigma^{\mu(|x|)})\,[\langle x, y\rangle \in A'].$$

Since for all x, $\langle x, 0^{p(|x|)}\rangle \notin A$, for all x, $\langle x, 0^{\mu(|x|)}\rangle \notin A'$.

For each $n \geq 1$, we can specify each string $y \in \Sigma^{\mu(n)}$ by bit positions at which y has a 1; i.e., y can be specified via the set $\{i \,\big|\, 1 \leq i \leq p(n) \wedge y_i = 1\}$.

For each $n \geq 1$, let $\mathcal{U}(n) = \{1, \ldots, \mu(n)\}$. Then for every $n \geq 1$ the power set of $\mathcal{U}(n)$ represents $\Sigma^{\mu(n)}$; i.e., each element in $\Sigma^{\mu(n)}$ can be viewed as a subset of $\mathcal{U}(n)$. For each $x \in \Sigma^*$, let $\mathcal{F}(x)$ be the set of all $y \subseteq \mathcal{U}(|x|)$ such that $\langle x, y\rangle \in A'$. By our assumption, for every $x \in \Sigma^*$, the empty set (which corresponds to the string $0^{\mu(|x|)}$) is not in $\mathcal{F}(x)$. For each $n \geq 1$, let $\mathcal{Z}(n)$ be the family of all the weight functions that assign to each number i, $1 \leq i \leq p(n)$, a positive weight of at most $4\mu(|x|)$.

Let x be an arbitrary string. Apply Lemma 4.1 with $m = 1$, $\mathcal{U} = \mathcal{U}(|x|)$, $\mathcal{Z} = \mathcal{Z}(|x|)$, $\mathcal{F}_1 = \mathcal{F}(x)$, and $R = 4\mu(|x|)$. Then,

- if $x \in L$, then the fraction of the weight functions in $\mathcal{Z}(|x|)$ with respect to which $\mathcal{F}(x)$ has exactly one minimum-weight element is more than $\frac{3}{4}$, and
- if $x \notin L$, then the fraction of the weight functions in $\mathcal{Z}(|x|)$ with respect to which $\mathcal{F}(x)$ has exactly one minimum-weight element is 0.

For every $x \in \Sigma^*$, every $W \in \mathcal{Z}(|x|)$, and every i, $1 \leq i \leq \mu(|x|)$, $W(i) \leq 4\mu(|x|)$. Thus, for every $x \in \Sigma^*$, every $W \in \mathcal{Z}(|x|)$, and every $y \subseteq \mathcal{U}(|x|)$, $W(y) \leq 4\mu^2(|x|)$. Define

$$B = \{\langle x, W, j\rangle \mid W \in \mathcal{Z}(|x|) \wedge 1 \leq j \leq 4\mu^2(|x|) \wedge \\ ||\{y \in \mathcal{F}(x) \mid W(y) = j \wedge \langle x, y\rangle \in A'\}|| = 1\},$$

where W is encoded as $W(1)\# \cdots \#W(\mu(|x|))$. Then $B \in$ US, which can be witnessed by the nondeterministic Turing machine M that on input u behaves as follows: First, M checks whether u is of the form $\langle x, w_1\#w_2\# \cdots \#w_{\mu(|x|)}, j\rangle$ for some j, $1 \leq j \leq 4\mu^2(|x|)$, and some $w_1, \cdots, w_{\mu(|x|)}$, $1 \leq w_1, \cdots, w_{\mu(|x|)} \leq 4\mu(|x|)$. If the check fails, M immediately rejects u. Otherwise, using precisely $\mu(|x|)$ nondeterministic moves, M selects $y \in \Sigma^{\mu(|x|)}$; then M accepts u if and only if $W(y) = j$ and $\langle x, y\rangle \in A$, where W is the weight function expressed by the string $w_1\#w_2\# \cdots \#w_{\mu(|x|)}$, i.e., for all i, $1 \leq i \leq \mu(|x|)$, $W(i) = w_i$. Since $B \in$ US, there is a polynomial-time many-one reduction g from B to USAT.

By the above probability analysis, for every $x \in \Sigma^*$,

- if $x \in L$, the proportion of $W \in \mathcal{Z}(|x|)$ such that for some j, $1 \leq j \leq 4\mu^2(|x|)$, $\langle x, W, j\rangle \in B$ is at least $\frac{3}{4}$, and
- if $x \notin L$, the proportion of $W \in \mathcal{Z}(|x|)$ such that for some j, $1 \leq j \leq 4\mu^2(|x|)$, $\langle x, W, j\rangle \in B$ is 0.

Let N be a probabilistic Turing machine that, on input $x \in \Sigma^*$, behaves as follows:

Step 1 N picks a weight function W as follows: For each i, $1 \leq i \leq \mu(|x|)$, N uniformly, randomly selects a binary string u_i having length $2+\log \mu(|x|)$, then sets the value of $W(i)$ to the binary integer $1\hat{u}_i$, where $\hat{u}_i$ is the string u_i with its leading 0s omitted.

Step 2 N picks j, $1 \leq j \leq 4\mu^2(|x|)$, as follows: N selects a binary string v having length $2 + 2\log \mu(|x|)$ uniformly at random. Then N sets j to

the integer whose binary encoding is $1\hat{v}$, where $\hat{v}$ is the string v with its leading 0s omitted.

Step 3 N asks its oracle whether $g(\langle x, W, j\rangle) \in$ USAT. If the query is answered positively, then N accepts x. Otherwise, it rejects x.

Let $x \in \Sigma^*$ be an input to N^{USAT}. Suppose $x \in L$. In Step 1, N^{USAT} on input x selects with probability at least $\frac{3}{4}$ a weight function W such that for some j, $1 \leq j \leq 4\mu^2(|x|)$, $\langle x, W, j\rangle \in B$. Furthermore, in Step 2, N^{USAT} on input x selects each j, $1 \leq j \leq 4\mu^2(|x|)$, with probability $\frac{1}{4\mu^2(|x|)}$. So, the probability that N^{USAT} on input x generates a query $\langle x, W, j\rangle$ that belongs to B is at least $\frac{3}{16\mu^2(|x|)}$. Define $q(n) = 22p^2(n)$. Since for all $n \geq 1$, $\mu(n)$ is the smallest power of 2 that is greater than or equal to $p(n)$, for all $n \geq 1$, $2p(n) \geq \mu(n)$. So, $\frac{3}{16\mu^2(|x|)} \geq \frac{3}{64p^2(|x|)} \geq \frac{1}{22p^2(|x|)} = \frac{1}{q(|x|)}$. Thus, the probability that N^{USAT} on input x accepts is at least $\frac{1}{q(|x|)}$. On the other hand, suppose $x \notin L$. Then, N^{USAT} on x rejects with probability 1. Thus, $L \leq_{\mathrm{randomized}} \mathrm{USAT}$.

❑ Theorem 4.2

In the proof above, define

$$B' = \{\langle x, W, j\rangle \mid x \in \Sigma^* \wedge 1 \leq j \leq 4\mu^2(|x|) \wedge W \in \mathcal{Z}(|x|) \wedge \\ ||\{y \in \mathcal{F}(x) \mid W(y) = j \wedge \langle x, y\rangle \in A'\}|| \text{ is an odd number}\}.$$

Then $B' \in \oplus\mathrm{P}$. Also, in Step 3 of the program of machine N, replace the query string by $\langle x, W, j\rangle$. Call this new machine $\hat{N}$. For every $x \in L$, the same probability analysis holds because 1 is an odd number, so $\hat{N}^{B'}$ on input x accepts with probability at least $\frac{1}{q(|x|)}$. For every $x \in \overline{L}$, $\hat{N}^{B'}$ on input x rejects with probability 1 because $\mathcal{F}(x)$ is empty and 0 is an even number. This implies that $L \leq_{\mathrm{randomized}} B'$. Furthermore, define T to be the probabilistic oracle Turing machine that on input x, sequentially execute independent simulation of $\hat{N}$ on x $q(|x|)$ times, and then accepts if $\hat{n}$ in at least one of the $q(|x|)$ simulations and rejects otherwise. For every $x \in L$, the probability that $T^{B'}$ on input x rejects is at most $(1 - \frac{1}{q(|x|)})^{q(|x|)}$. Since $q(n) = 22p^2(n)$ and for all $n \geq 0$, $p(n) \geq 1$, for all $n \geq 0$, $q(n) \geq 22$. So, the probability that $T^{B'}$ on input x rejects is at most $(1 - \frac{1}{22})^{22} < \frac{1}{2}$. On the other hand, for every $x \in \overline{L}$, the probability that $T^{B'}$ on input x accepts is 0. Thus, we have proven the following theorem.

Theorem 4.3 $\mathrm{NP} \subseteq \mathrm{RP}^{\oplus\mathrm{P}} \subseteq \mathrm{BPP}^{\oplus\mathrm{P}}$.

4.2 Toda's Theorem: PH $\subseteq$ P$^{\mathrm{PP}}$

4.2.1 PH and BPP$^{\oplus\mathrm{P}}$

By Theorem 4.3, $\mathrm{NP} \subseteq \mathrm{BPP}^{\oplus\mathrm{P}}$. Since $\mathrm{P}^{\mathrm{BPP}^A} = \mathrm{BPP}^A$ for every oracle A (see Proposition 4.6 below), it holds that $\Delta_2^p \subseteq \mathrm{BPP}^{\oplus\mathrm{P}}$.

Pause to Ponder 4.4 *Can we extend this inclusion in* $BPP^{\oplus P}$ *to even higher levels of the polynomial hierarchy than* Δ_2^p*?*

In this section we show that indeed we can.

Theorem 4.5 *For every* $k \geq 1$, $\Sigma_k^p \subseteq BPP^{\oplus P}$. *Hence,* $PH \subseteq BPP^{\oplus P}$.

The proof of Theorem 4.5 is by induction on k. The base case is, of course, Theorem 4.3. For the induction step, we establish, for each $k \geq 1$, that $\Sigma_{k+1}^p \subseteq BPP^{\oplus P}$ by combining Theorem 4.3 and our inductive hypothesis, $\Sigma_k^p \subseteq BPP^{\oplus P}$, in the following three steps:

1. **(Apply Theorem 4.3 to the base machine)** The proof of Theorem 4.3 is relativizable, so, for every oracle A, $NP^A \subseteq BPP^{\oplus P^A}$. Noting that $\Sigma_{k+1}^p = NP^{\Sigma_k^p}$, we have the following, where the first inclusion is via the inductively true $\Sigma_k^p \subseteq BPP^{\oplus P}$ and the second is via using relativized Theorem 4.3 as the oracle ranges over all $BPP^{\oplus P}$ sets.
$$\Sigma_{k+1}^p \subseteq NP^{BPP^{\oplus P}} \subseteq BPP^{\oplus P^{BPP^{\oplus P}}}.$$
2. **(Swap BPP and $\oplus$P in the middle)** By Lemma 4.9 below, $\oplus P^{BPP^A} \subseteq BPP^{\oplus P^A}$, for every oracle A. So,
$$\Sigma_{k+1}^p \subseteq BPP^{BPP^{\oplus P^{\oplus P}}}.$$
3. **(Collapse BPP^{BPP} to BPP, and $\oplus P^{\oplus P}$ to $\oplus$P)** By part 2 of Proposition 4.6 below, $BPP^{BPP^A} = BPP^A$ for every oracle A. By part 2 of Proposition 4.8 below, $\oplus P^{\oplus P} = \oplus P$. So,
$$\Sigma_{k+1}^p \subseteq BPP^{\oplus P}.$$

We will first prove the two collapse results in Step 3, together with characterizations of BPP and $\oplus$P. The characterizations will be useful when we prove the "swapping" property in Step 2.

The results we prove in the rest of the section hold relative to any oracle. For simplicity, we prove only their nonrelativized versions.

Proposition 4.6

1. **(The error probability of BPP computation can be exponentially reduced without sacrificing much computation time)** *For every* $L \in BPP$ *and every polynomial* r*, there exist a polynomial* p *and a language* $A \in P$ *such that, for every* $x \in \Sigma^*$*,*
   a) *if* $x \in L$*, then the proportion of* $y \in \Sigma^{p(|x|)}$ *such that* $x\#y$ *belongs to* A *is at least* $1 - 2^{-r(|x|)}$*, and*
   b) *if* $x \notin L$*, then the proportion of* $y \in \Sigma^{p(|x|)}$ *such that* $x\#y$ *belongs to* A *is at most* $2^{-r(|x|)}$.
2. *The* BPP *hierarchy collapses; i.e.,* $BPP = P^{BPP} = BPP^{BPP} = P^{BPP^{BPP}} = BPP^{BPP^{BPP}} = \ldots$.

Proof We prove first part 1 of the proposition. Let $L \in \mathrm{BPP}$ via probabilistic Turing machine M. That is, for every $x \in \Sigma^*$, M accepts x with probability at least $\frac{3}{4}$ if $x \in L$ and with probability at most $\frac{1}{4}$ otherwise. Let r be any polynomial and let $q(n) = 6r(n)+1$. Let N be a probabilistic Turing machine that, on input x, simulates M on input x exactly $q(|x|)$ times, and accepts then if and only if M accepts in a majority of the simulations. Let $n \geq 1$ and $x \in \Sigma^n$. Suppose that $x \in L$. Let α be the probability that M on input x accepts and $e = \alpha - \frac{1}{2}$. Note that $e \geq \frac{1}{4}$. The probability that N on input x rejects is at most

$$\begin{aligned}
&\sum_{0 \leq i \leq \lfloor \frac{q(n)}{2} \rfloor} \binom{q(n)}{i} \left(\frac{1}{2}+e\right)^i \left(\frac{1}{2}-e\right)^{q(n)-i} \\
&\leq \sum_{0 \leq i \leq \lfloor \frac{q(n)}{2} \rfloor} \binom{q(n)}{i} \left(\frac{1}{2}+e\right)^{\frac{q(n)}{2}} \left(\frac{1}{2}-e\right)^{\frac{q(n)}{2}} \\
&= \sum_{0 \leq i \leq \lfloor \frac{q(n)}{2} \rfloor} \binom{q(n)}{i} \left(\frac{1}{4}-e^2\right)^{\frac{q(n)}{2}} \\
&\leq \sum_{0 \leq i \leq q(n)} \binom{q(n)}{i} \left(\frac{1}{4}-e^2\right)^{\frac{q(n)}{2}} \\
&= 2^{q(n)} \left(\frac{1}{4}-e^2\right)^{\frac{q(n)}{2}} \\
&= (1-4e^2)^{\frac{q(n)}{2}}.
\end{aligned}$$

This is at most $\left(\frac{3}{4}\right)^{\frac{q(n)}{2}} \leq \left(\frac{3}{4}\right)^{3r(n)} < 2^{-r(n)}$. Similarly, if $x \notin L$, then the probability that N on x accepts is at most $2^{-r(|x|)}$.

We view the randomized moves of N as being directed by tossing of coins. More precisely, at each randomized step of N, there are two possible choices and N selects one of the two by tossing a fair coin, the "head" for on move and the "tail" for the other. Then the coin tosses of N can be "normalized" in the sense that there is a polynomial p such that, for every $x \in \Sigma^*$, N on x tosses exactly $p(|x|)$ coins. Namely, the machine keeps track of the number of coin tosses it makes and, at the end of computation, if the number is less than $p(|x|)$, the machine makes dummy coin tosses to make the total number of coin tosses equal to $p(|x|)$. Pick such a p and let A be the set of all $x\#y$ with $y \in \Sigma^{p(|x|)}$ and such that N on x with coin tosses y accepts. Clearly, $A \in \mathrm{P}$ and, for every $x \in \Sigma^*$, the proportion of $y \in \Sigma^{p(|x|)}$ such that $x\#y \in A$ is equal to the probability that N on x accepts. So, conditions 1a and 1b both hold.

We now prove part 2 of the proposition. Let $L \in \mathrm{BPP}^A$ with $A \in \mathrm{BPP}$. The language L is defined in terms of two probabilistic polynomial-time Tur-

ing machines, one for the base computation and the other for the oracle. Intuitively, we will show below that the two machines can be combined into a single probabilistic polynomial-time machine without creating much error.

Note that Part 1 in the above holds relative to any oracle. Let $r(n)$ be an arbitrary strictly increasing polynomial. Then there is a polynomial-time probabilistic oracle Turing machine D such that, for every $x \in \Sigma^*$, D^A on input x decides the membership of x in L correctly with probability at least $1 - 2^{-r(|x|)}$. We may assume that there exists a polynomial p such that, for every $x \in \Sigma^*$ and every oracle Z, D^Z on input x makes at most $p(|x|)$ queries. Also, we may assume that each query of D on x is at least as long as the input. We replace the oracle A by $\hat{A} = \{x\#y \,|\, y \in A\}$ and replace D by a new machine $\hat{D}$ that, on input $x \in \Sigma^*$, simulates D on input x by substituting for each query y to A the a query $x\#y$ to $\hat{A}$. By part 1, there is a polynomial-time probabilistic Turing machine N such that, for every $u \in \Sigma^*$, N on u correctly decides the membership of u in $\hat{A}$ with probability $1 - 2^{-r(|u|)}$. Let M be a probabilistic Turing machine that, on input $x \in \Sigma^*$, simulates $\hat{D}$ on input x and when $\hat{D}$ makes a query, say u, to the oracle, simulates N on input u to decide the oracle answer. For every $x \in \Sigma^*$, $\hat{D}$ on input x makes at most $p(|x|)$ queries, and for every query u of $\hat{D}$ on input x, N on input u makes an error with probability at most $2^{-r(|u|)} \leq 2^{-r(|x|)}$ since $|u| \geq |x|$ and r is an increasing polynomial. So, for every $x \in \Sigma^*$, the probability of the paths on which the computation of M differs from that of $\hat{D}^{\hat{A}}$ is at most $p(|x|)2^{-r(|x|)}$. Since D^A makes an error with probability at most $2^{-r(|x|)}$, the probability that M makes an error is at most $(p(|x|) + 1)2^{-r(|x|)}$. Since r is an increasing polynomial, for x sufficiently large, the error probability of M on x is smaller than $\frac{1}{4}$. Hence $L \in \mathrm{BPP}$. Since $\mathrm{BPP}^{\mathrm{BPP}} = \mathrm{BPP}$, clearly $\mathrm{BPP}^{\mathrm{BPP}^{\mathrm{BPP}}} = \mathrm{BPP}^{\mathrm{BPP}} = \mathrm{BPP}$ via the application of this fact and, more generally, the BPP hierarchy collapses to BPP by induction. ❑

For a class $\mathcal{C}$, $\mathcal{C}/\mathrm{poly}$ is the class of all languages L for which there exist an $A \in \mathcal{C}$ and a polynomially length-bounded function $h : \Sigma^* \to \Sigma^*$ such that, for every $x \in \Sigma^*$, $x \in L$ if and only if $\langle x, h(0^{|x|})\rangle \in A$ (see Sect. A.6). We have the following corollary.

Corollary 4.7 $\mathrm{BPP} \subseteq \mathrm{P}/\mathrm{poly}$.

Proof Let $L \in \mathrm{BPP}$. Let $r(n) = n + 1$. By part 1 of Proposition 4.6, there exist a polynomial p and $A \in \mathrm{P}$ such that, for every $x \in \Sigma^*$, the proportion of $y \in \Sigma^{p(|x|)}$ for which the equivalence,

$$x \in L \iff \langle x, y\rangle \in A,$$

does not hold is at most $2^{-(|x|+1)}$. Let $n \geq 1$. The proportion of $y \in \Sigma^{p(n)}$ such that $||\{x \in \Sigma^n \,|\, x \in L \iff \langle x, y\rangle \in A \text{ does not hold }\}|| \geq 1$ is at most $||\Sigma^n||2^{-(n+1)} = 2^n 2^{-(n+1)} < 1$. So, there is some $y \in \Sigma^{p(n)}$ such that, for every $x \in \Sigma^n$, $x \in L \iff \langle x, y\rangle \in A$. Let $h(0^n)$ be the smallest such y. Then,

for every $x \in \Sigma^n$, $x \in L$ if and only if $\langle x, h(0^n)\rangle \in A$. Since $|h(0^n)| = p(n)$, this implies that $L \in \mathrm{P/poly}$. ❑

Proposition 4.8

1. *For every $L \in \oplus\mathrm{P}$, there exist a polynomial p and a language $A \in \mathrm{P}$ such that, for every $x \in \Sigma^*$, $x \in L$ if and only if $||\{y \in \Sigma^{p(|x|)} \mid x\#y \in A\}||$ is odd.*
2. $\oplus\mathrm{P}^{\oplus\mathrm{P}} = \mathrm{P}^{\oplus\mathrm{P}} = \oplus\mathrm{P}$.

Proof To prove part 1, let L be in $\oplus\mathrm{P}$ via a nondeterministic polynomial-time Turing machine M. That is, for every $x \in \Sigma^*$,

$$x \in L \text{ if and only if } \#\mathrm{acc}_M(x) \text{ is odd.}$$

Let p be a polynomial bounding the runtime of M. Define A to be the set of all $x\#y$, $|y| = p(|x|)$, such that y is an accepting computation path of M on x (that is, a sequence z of bits representing nondeterministic moves for $M(x)$ leading to acceptance, on the path it specifies, on or after the move specified by the $|z|$th bit of z but before any further nondeterministic move is attempted) followed by an appropriate number of zeros. Clearly, $A \in \mathrm{P}$ and, for every $x \in \Sigma^*$, the number of $y, |y| = p(|x|)$, such that $x\#y \in A$ is $\#\mathrm{acc}_M(x)$.

To prove part 2, let L be an arbitrary language in $\oplus\mathrm{P}^{\oplus\mathrm{P}}$. There exist a nondeterministic polynomial-time oracle Turing machine M and a language $B \in \oplus\mathrm{P}$ such that, for all x, $x \in L$ if and only if the number of accepting computation paths of M^B on input x is an odd number. We will construct a nondeterministic polynomial-time Turing machine M' witnessing that $L \in \oplus\mathrm{P}$. Let N_1 be a nondeterministic Turing machine witnessing that $B \in \oplus\mathrm{P}$. As we will see in Proposition 9.3, #P is closed under addition. So, the function $1 + \#\mathrm{acc}_{N_0}$ thus belongs to #P. Let N_1 be such that $\#\mathrm{acc}_{N_1} = 1 + \#\mathrm{acc}_{N_0}$. The function $\#\mathrm{acc}_{N_1}$ flips the parity of $\#\mathrm{acc}_{N_0}$, in the sense that for all x, $\#\mathrm{acc}_{N_1}(x)$ is an odd number if and only if $\#\mathrm{acc}_{N_0}(x)$ is an even number. Thus, N_1 witnesses that $\overline{B} \in \oplus\mathrm{P}$. Let M' be the nondeterministic Turing machine that, on input x, simulates M on x but each time M makes a query to the oracle, instead of making a query M' does the following two steps. (1) M' guesses a bit, $b \in \{0, 1\}$, about the oracle answer (where $b = 0$ is interpreted as 'Yes' and $b = 1$ as 'No') and a path of simulation of N_b (i.e., N_0 or N_1, depending on the choice of b) on input w, where w is the query string of M. (2) Then M' returns to its simulation of M on input x with the guessed oracle answer. The machine M' accepts along a given path if and only if all the simulations of the machines N_0, N_1, and M along that path accepted. We claim that M' witnesses that $L \in \oplus\mathrm{P}$.

For each accepting computation path π of M' on x, let $\tau(\pi)$ be the part of π corresponding to the computation of M on x and the guesses about the queries. That is, $\tau(\pi)$ is π with all simulations of N_0 and N_1 removed. Only the guessed values of b remain encoded in π. Let $t = \tau(\pi)$ for some

π. How many accepting computation paths have t as their τ-value? Let $y_1, \ldots, y_m$ be the query strings along π and $b_1, \ldots, b_m \in \{0,1\}$ be the guessed values encoded in π. Then the number of such paths is the product of $\#\mathrm{acc}_{N_{b_1}}(y_1), \ldots, \#\mathrm{acc}_{N_{b_m}}(y_m)$. This number is odd if and only if all the guesses about the queries are correct. Thus, the parity of the number of accepting computation paths of M' on x equals that of the number of accepting computation paths of M on x relative to B. Hence, $L \in \oplus\mathrm{P}$. ❑

Lemma 4.9 $\oplus\mathrm{P}^{\mathrm{BPP}} \subseteq \mathrm{BPP}^{\oplus\mathrm{P}}$.

Proof Let $L \in \oplus\mathrm{P}^{\mathrm{BPP}}$. We show that $L \in \mathrm{BPP}^{\oplus\mathrm{P}}$. By part 1 of Proposition 4.8, there exist a polynomial p and $A \in \mathrm{P}^{\mathrm{BPP}}$ such that, for every $x \in \Sigma^*$,

$$x \in L \Longleftrightarrow ||\{y \in \Sigma^{p(|x|)} \,\big|\, x\#y \in A\}|| \text{ is odd.}$$

Furthermore, by part 2 of Proposition 4.6, $\mathrm{P}^{\mathrm{BPP}} = \mathrm{BPP}$, so $A \in \mathrm{BPP}$. Then, by part 1 of Proposition 4.6, for every polynomial r, there exist a polynomial p and $B \in \mathrm{P}$ such that, for every $u \in \Sigma^*$,

$$\begin{aligned}&\text{the proportion of } v \in \Sigma^{p(|u|)} \text{ such that } u\#v \in B \text{ is at least}\\ &1 - 2^{-r(|u|)} \text{ if } u \in A \text{ and at most } 2^{-r(|u|)} \text{ otherwise.}\end{aligned} \tag{4.1}$$

Let $r(n) = p(n) + 2$. Let s be the polynomial such that, for every $x \in \Sigma^*$ and $y \in \Sigma^{p(|x|)}$, $s(|x|) = q(|x\#y|)$. Define (recall that $\#$ is the specific function defined in Sect. 4.1.2)

$$\begin{aligned}C = \{x\#v \,\big|\, &v \in \Sigma^{s(|x|)} \wedge\\ &||\{y \in \Sigma^{p(|x|)} \mid (x\#y)\#v \in B\}|| \text{ is an odd number}\}.\end{aligned}$$

Clearly, $C \in \oplus\mathrm{P}$. For each $x \in \Sigma^*$, let

$$a(x) = ||\{y \in \Sigma^{p(|x|)} \,\big|\, x\#y \in A\}||$$

and, for each $x \in \Sigma^*$ and $v \in \Sigma^{s(|x|)}$, let

$$c(x\#v) = ||\{y \in \Sigma^{p(|x|)} \,\big|\, (x\#y)\#v \in C\}||.$$

By equation 4.1, for every $x \in \Sigma^*$, the proportion of $v \in \Sigma^{s(|x|)}$ satisfying the condition

$$(\forall y \in \Sigma^{p(|x|)})\,[x\#y \in A \iff (x\#y)\#v \in B]$$

is at least $1 - 2^{p(|x|)}2^{-r(s(|x|))} \geq 1 - 2^{p(|x|)-p(s(|x|))-2} \geq 1 - 2^{-2} = \frac{3}{4}$, and thus the proportion of $v \in \Sigma^{s(|x|)}$ such that $a(x) = c(x\#v)$ is at least $\frac{3}{4}$. Thus, for every $x \in \Sigma^*$, for at least $\frac{3}{4}$ of $v \in \Sigma^{s(|x|)}$, $a(x)$ is odd if and only if $c(x\#v)$ is odd. Note that $a(x)$ is odd if and only if $x \in L$ and that $c(x\#v)$ is odd if and only if $x\#v \in C$. So, for every $x \in \Sigma^*$, for at least $\frac{3}{4}$ of $v \in \Sigma^{s(|x|)}$, $x \in L$ if and only if $x\#v \in C$. Thus, $L \in \mathrm{BPP}^{\oplus\mathrm{P}}$.

Intuitively, the above argument can be explained as follows: We are looking at a table whose rows are y's and whose columns are v's, where the y's

correspond to the nondeterministic guesses for the "parity" computation and the v's correspond to the random guesses used in the "BPP" computation (for testing whether a given $x\#y$ belongs to A). For each y and z, we place a letter "X" in the (y, v) entry of the table exactly if the randomized guesses v for the BPP computation on input $x\#y$ lead to an error, i.e., either ($x\#y \in A$ and $(x\#y)\#v \notin B$) or ($x\#y \notin A$ and $(x\#y)\#v \in B$). For each column v with no "X" the number of y such that $(x\#y)\#v \in B$ is equal to the number of y such that $x\#y \in A$. So, for such column v, $x \in L$ if and only if the number of y such that $(x\#y)\#v \in B$ is odd. In each row, the fraction of the entries having an "X" is at most $2^{-r(s(|x|))}$. There are only $2^{p(|x|)}$ rows. Thus the fraction of the columns with an "X" is at most $2^{-r(s(|x|))+p(|x|)}$. As equation 4.1 holds for any polynomial r, we can select r so that this amount is less than $\frac{1}{4}$. So, transpose the table: We'll pick v first, then pick y. Then for more than $\frac{3}{4}$ of v it holds that $x \in L$ if and only if the number of y such that $(x\#y)\#v \in B$ is odd. Hence we can switch the "BPP" part and the "parity" part. ❑

This concludes the proof of Theorem 4.5. We now show below some immediate corollaries to Theorem 4.5. Since BPP $\subseteq$ P/poly by Corollary 4.7 and $\mathrm{P}^{\oplus\mathrm{P}} = \oplus\mathrm{P}$ by part 2 of Proposition 4.8, $\mathrm{BPP}^{\oplus\mathrm{P}} \subseteq \oplus\mathrm{P/poly}$. Thus, PH $\subseteq \oplus$P/poly.

Corollary 4.10 PH $\subseteq \oplus$P/poly.

By Lemma 4.9, $\oplus\mathrm{P}^{\mathrm{BPP}} \subseteq \mathrm{BPP}^{\oplus\mathrm{P}}$. By relativizing $\oplus$P by PH and then applying Theorem 4.5, we obtain the following result.

Corollary 4.11 $\oplus\mathrm{P}^{\mathrm{PH}} \subseteq \mathrm{BPP}^{\oplus\mathrm{P}} \subseteq \oplus\mathrm{P/poly}$.

4.2.2 PP Is Hard for the Polynomial Hierarchy

We now prove Toda's Theorem.

Theorem 4.12 (Toda's Theorem) PH $\subseteq \mathrm{P}^{\#\mathrm{P}[1]}$.

Corollary 4.13 PH $\subseteq \mathrm{P}^{\#\mathrm{P}} = \mathrm{P}^{\mathrm{PP}}$.

Theorem 4.12, Toda's Theorem, follows immediately from Theorem 4.5 in light of the following lemma.

Lemma 4.14 $\mathrm{PP}^{\oplus\mathrm{P}} \subseteq \mathrm{P}^{\#\mathrm{P}[1]}$. *In particular,* $\mathrm{BPP}^{\oplus\mathrm{P}} \subseteq \mathrm{P}^{\#\mathrm{P}[1]}$.

Proof of Lemma 4.14 Let $L \in \mathrm{PP}^{\oplus\mathrm{P}}$. There exist a polynomial p, a function $f \in$ FP, and a language $A \in \mathrm{P}^{\oplus\mathrm{P}} = \oplus\mathrm{P}$ (by Proposition 4.8) such that, for every $x \in \Sigma^*$,

$$x \in L \Longleftrightarrow ||\{y \in \Sigma^{p(|x|)} \mid x\#y \in A\}|| \geq f(x). \tag{4.2}$$

Let M be a nondeterministic polynomial-time Turing machine witnessing that $A \in \oplus\mathrm{P}$. So, for every $x \in \Sigma^*$, $x \in A$ if and only if $\#\mathrm{acc}_M(x)$ is odd. Define $s_0(z) = z$ and, for each $i \geq 1$, define polynomial $s_i(z)$ with coefficients in $\mathbb{N}$ by

$$s_i(z) = 3(s_{i-1}(z))^4 + 4(s_{i-1}(z))^3. \tag{4.3}$$

Claim 4.15 *For every $i \geq 0$ and every $z \in \mathbb{N}$, if z is even, then $s_i(z)$ is a multiple of 2^{2^i}, and if z is odd, then $s_i(z) + 1$ is a multiple of 2^{2^i}.*

Proof of Claim 4.15 The proof is by induction on i. The claim trivially holds for the base case $i = 0$. For the induction step, let $i = i_0$ for some $i_0 \geq 1$ and suppose that the claim holds for values of i that are less than i_0 and greater than or equal to 0. Suppose that z is even. By the inductive hypothesis, $s_{i-1}(z)$ is divisible by $2^{2^{i-1}}$. Since $s_i(z)$ is divisible by $(s_{i-1}(z))^2$ and $2^{2^i} = (2^{2^{i-1}})^2$, $s_i(z)$ is divisible by 2^{2^i}. Thus, the claim holds for even z. Suppose that z is odd. By the inductive hypothesis, $s_{i-1}(z) = m2^{2^{i-1}} - 1$ for some $m \in \mathbb{N}$. So,

$$\begin{aligned}
s_i(z) &= 3(m2^{2^{i-1}} - 1)^4 + 4(m2^{2^{i-1}} - 1)^3 \\
&= 3(m^4 2^{4(2^{i-1})} - 4m^3 2^{3(2^{i-1})} + 6m^2 2^{2(2^{i-1})} - 4m2^{2^{i-1}} + 1) \\
&\qquad + 4(m^3 2^{3(2^{i-1})} - 3m^2 2^{2(2^{i-1})} + 3m2^{2^{i-1}} - 1) \\
&= 3m^4 2^{4(2^{i-1})} - 8m^3 2^{3(2^{i-1})} + 6m^2 2^{2(2^{i-1})} - 1 \\
&= 2^{2^i}(3m^4 2^{2^i} - 8m^3 2^{2^{i-1}} + 6m^2) - 1.
\end{aligned}$$

Thus, the claim holds for odd z also. ❑ Claim 4.15

For each $x \in \Sigma^*$, let $\ell_x = \lceil \log p(|x|) + 1 \rceil$ and define $r_x(z) = (s_{\ell_x}(z))^2$ and $g(x) = r_x(\#\mathrm{acc}_M(x))$. For every $x \in \Sigma^*$, $r_x(z)$ is a polynomial in z of degree $2^{2\ell_x}$. The coefficients of the polynomial r_x are all nonnegative and polynomial-time computable. We claim that the function g is in #P. This can be seen as follows. Let G be a nondeterministic Turing machine that, on input x, operates as follows:

Step 1 G computes $r_x(z) = a_0 z^0 + a_1 z^1 + \cdots + a_m z^m$, where $m = 2^{2\ell_x}$.
Step 2 G computes the list $I = \{i \mid 0 \leq i \leq 2^{2\ell_x} \wedge a_i \neq 0\}$.
Step 3 G nondeterministically selects $i \in I$.
Step 4 G nondeterministically selects d, $1 \leq d \leq a_i$.
Step 5 G simulates M on input x i times.
Step 6 G accepts if and only if M accepts during each of the i simulations.

Then G satisfies $g = \#\mathrm{acc}_G$. By Claim 4.15, the following conditions hold for every $x \in \Sigma^*$:

($\star$) If $x \in A$, then $\#\mathrm{acc}_M(x)$ is odd, so $g(x) - 1$ is a multiple of $2^{2^{\ell_x}}$, and thus, $g(x)$ is of the form $m2^{p(|x|)+1} + 1$ for some m.
($\star\star$) If $x \notin A$, then $\#\mathrm{acc}_M(x)$ is even, so $g(x)$ is a multiple of $2^{2^{\ell_x}}$, and thus, $g(x)$ is of the form $m2^{p(|x|)+1}$ for some m.

Define

$$h(x) = \sum_{|y|=p(|x|)} g(x\#y).$$

There is a nondeterministic Turing machine H such that $h = \#\mathrm{acc}_H$, so $h \in \#\mathrm{P}$. In particular, H guesses $y \in \Sigma^{p(|x|)}$ and simulates G on $x\#y$. By equation 4.2, $(\star)$, and $(\star\star)$, the lowest $p(|x|)+1$ bits of the binary representation of $h(x)$ represent the number of $y \in \Sigma^{p(|x|)}$ such that $x\#y \in A$. So, for every $x \in \Sigma^*$, $x \in L \iff$ the leftmost $p(|x|)+1$ bits of $h(x)$ is lexicographically at least $010^{p(|x|)-1}$. This implies that L is decidable by a polynomial-time Turing machine that makes one query to h. Since L was an arbitrary $\mathrm{PP}^{\oplus\mathrm{P}}$ set and $h = h_L \in \#\mathrm{P}$, it follows that $\mathrm{PP}^{\oplus\mathrm{P}} \subseteq \mathrm{P}^{\#\mathrm{P}[1]}$, the class of languages decided by a polynomial-time algorithm with one question to a #P oracle. ❑ Lemma 4.14

So, Theorem 4.12 is established. Corollary 4.13 follows immediately from Theorem 4.12 in light of Proposition 4.16.

Proposition 4.16 $\mathrm{P}^{\mathrm{PP}} = \mathrm{P}^{\#\mathrm{P}}$.

Proof First we show $\mathrm{P}^{\mathrm{PP}} \subseteq \mathrm{P}^{\#\mathrm{P}}$. Let $L \in \mathrm{PP}$. There exist a polynomial p, a language $A \in \mathrm{P}$, and $f \in \mathrm{FP}$ such that, for every $x \in \Sigma^*$,

$$x \in L \iff ||\{y \,\big|\, |y| = p(|x|) \wedge \langle x, y\rangle \in A\}|| \geq f(x).$$

Let N be a nondeterministic Turing machine that, on input x, guesses $y \in \Sigma^{p(|x|)}$, and accepts x if and only if $\langle x, y\rangle \in A$. Clearly, N can be polynomial time-bounded. For every $x \in \Sigma^*$, $\#\mathrm{acc}_N(x) = ||\{y \in \Sigma^{p(|x|)} \,\big|\, \langle x, y\rangle \in A\}||$. Since $f \in \mathrm{FP}$ the membership in L can be tested in $\mathrm{P}^{\#\mathrm{P}[1]}$. Thus, $\mathrm{P}^{\mathrm{PP}} \subseteq \mathrm{P}^{\#\mathrm{P}}$.

Next we show $\mathrm{P}^{\mathrm{PP}} \supseteq \mathrm{P}^{\#\mathrm{P}}$. Let f be an arbitrary #P function. Let $f = \#\mathrm{acc}_N$ for some polynomial-time nondeterministic Turing machine N and let p be a polynomial that strictly bounds the runtime of N. Then for all x $\#\mathrm{acc}_N(x) < 2^{p(|x|)}$. Define $L = \{\langle x, y\rangle \,\big|\, 0 \leq y \leq 2^{p(|x|)} - 1 \wedge \#\mathrm{acc}_N(x) \geq y\}$. Define N' to be the nondeterministic Turing machine that, on input $x \in \Sigma^*$, operates as follows: N' simulates N on input x while counting in a variable C the number of nondeterministic moves that N makes along the simulated path. When N halts, N' guesses a binary string z of length $p(|x|) - C$ using exactly length $p(|x|) - C$ bits. Then N' accepts if and only if the simulated the path of N on x is accepting and $z \in 0^*$. Then for all x $\#\mathrm{acc}_N(x) = \#\mathrm{acc}_{N'}(x)$. Also, for all $x \in \Sigma^*$ and all computation paths π of N' on input x, N' along path π makes exactly $p(|x|)$ nondeterministic moves. Define D to the probabilistic Turing machine that, on input $\langle x, y\rangle$, $0 \leq y \leq 2^{p(|x|)} - 1$, operates as follows: D uniformly, randomly selects $b \in \{0, 1\}$, and then does the following:

- If $b = 0$, then D uniformly, randomly selects $z \in \{0, 1\}^{p(|x|)}$, and then accepts if the rank of z is at most $2^{p(|x|)} - y$ and rejects otherwise.

- If $b = 1$, then D simulates N' on input x by replacing each nondeterministic move of N' by a probabilistic move. More precisely, each time N' makes a nondeterministic move, deciding between two possible actions α and β, D selects uniformly, randomly $c \in \{0, 1\}$, and then D selects α if $c = 0$ and β otherwise. D accepts if N' on x along the simulated path accepts and rejects otherwise.

Clearly, D can be polynomial time-bounded. For every $x \in \Sigma^*$, the probability that D on input x accepts is

$$\frac{2^{p(|x|)} - y}{2^{p(|x|)+1}} + \frac{\#\mathrm{acc}_{N'}(x)}{2^{p(|x|)+1}}.$$

This is greater than or equal to $\frac{1}{2}$ if and only if $\#\mathrm{acc}_{N'}(x) \geq y$. Thus, $L \in \mathrm{PP}$. Hence, $\mathrm{P}^{\mathrm{PP}} \supseteq \mathrm{P}^{\#\mathrm{P}}$. ❑

We can strengthen Corollary 4.13 to show that not only the polynomial hierarchy but also $\mathrm{PP}^{\mathrm{PH}}$ is included in P^{PP}.

Corollary 4.17 $\mathrm{PP}^{\mathrm{PH}} \subseteq \mathrm{P}^{\mathrm{PP}}$.

Proof By Theorem 4.5, $\mathrm{PH} \subseteq \mathrm{BPP}^{\oplus \mathrm{P}}$ and by Lemma 4.14, $\mathrm{PP}^{\oplus \mathrm{P}} \subseteq \mathrm{P}^{\mathrm{PP}}$. So, it suffices to prove that, for every oracle X, $\mathrm{PP}^{\mathrm{BPP}^X} \subseteq \mathrm{PP}^X$. It follows that

$$\mathrm{PP}^{\mathrm{PH}} \subseteq \mathrm{PP}^{\mathrm{BPP}^{\oplus \mathrm{P}}} \subseteq \mathrm{PP}^{\oplus \mathrm{P}} \subseteq \mathrm{P}^{\mathrm{PP}}.$$

Again, we prove only the nonrelativized version. Let $L \in \mathrm{PP}^{\mathrm{BPP}}$. There exist a polynomial p, a function $f \in \mathrm{FP}$, and $A \in \mathrm{BPP}$ such that, for every $x \in \Sigma^*$,

$$\text{if } x \in L, \text{ then } ||\{y \in \Sigma^{p(|x|)} \mid x\#y \in A\}|| \geq f(x), \text{ and} \tag{4.4}$$

$$\text{if } x \notin L, \text{ then } ||\{y \in \Sigma^{p(|x|)} \mid x\#y \in A\}|| \leq f(x) - 1. \tag{4.5}$$

Let $r(n) = p(n) + 2$. By part 1 of Proposition 4.6, there exist a polynomial q and a language $B \in \mathrm{P}$ such that, for every $u \in \Sigma^*$,

$$\text{if } u \in A, \text{ then } ||\{v \in \Sigma^{q(|u|)} \mid u\#v \in B\}|| \geq 2^{q(|u|)}(1 - 2^{-r(|u|)}), \text{ and} \tag{4.6}$$

$$\text{if } u \notin A, \text{ then } ||\{v \in \Sigma^{q(|u|)} \mid u\#v \in B\}|| \leq 2^{q(|u|)}2^{-r(|u|)}. \tag{4.7}$$

Define $s(n) = 2(n + 1 + p(n))$. The length of $x\#y$ with $y \in \Sigma^{p(|x|)}$ is $s(|x|)$. Define D to be the set of all strings $x\#yv$, with $y \in \Sigma^{p(|x|)}$ and $v \in \Sigma^{q(s(|x|))}$, such that $x'\#v \in B$, where $x' = x\#y$. Then D is in P. For each $x \in \Sigma^*$, let $d(x) = ||\{yv \mid x\#yv \in D\}||$ and define g by $g(x) = f(x)2^{q(s(|x|))}(1 - 2^{-r(s(|x|))})$. Then $g \in \mathrm{FP}$. For every $x \in \Sigma^*$, if $x \in L$, then by equations 4.4 and 4.6, $d(x) \geq f(x)2^Q(1 - 2^{-R}) = g(x)$, where P, Q, and R respectively denote $p(|x|)$, $q(s(|x|))$, and $r(s(|x|))$. On the other hand, if $x \notin L$, then by equations 4.5 and 4.7

$$\begin{aligned} d(x) &\leq (f(x)-1)2^Q + (2^P - f(x) + 1)2^Q 2^{-R} \\ &= f(x)2^Q - 2^Q + 2^{P+Q-R} - f(x)2^{Q-R} + 2^{Q-R} \\ &= f(x)2^Q(1-2^{-R}) - 2^Q(1-2^{P-R}-2^{-R}) \\ &= g(x) - 2^Q(1-2^{P-R}-2^{-R}). \end{aligned}$$

Since $r(n) = p(n)+2$ and $s(n) > n$, $1-2^{P-R}-2^{-R}$ is at least $1-\frac{1}{4}-\frac{1}{4} > 0$. So, $d(x) < g(x)$. Hence, for every $x \in \Sigma^*$, $x \in L$ if and only if $d(x) \geq g(x)$. Thus, $L \in \mathrm{PP}^A$. ❑

4.3 NL/poly = UL/poly

In the previous sections we saw a number of results that connect the polynomial hierarchy to polynomial-time counting complexity classes. Do the analogs of those results hold for logspace classes? In particular, do the logspace analogs of PH $\subseteq \oplus$P/poly (Corollary 4.10) and PH $\subseteq \mathrm{P}^{\mathrm{PP}}$ (Corollary 4.13) hold? Since the NL hierarchy (under the Ruzzo–Simon–Tompa relativization, see Chap. 9) collapses to NL since NL = coNL (see Sect. A.7), we can simplify the question of whether the logspace analog of the former inclusion holds to the question of whether NL $\subseteq \oplus$L/poly and the question of whether the logspace analog of the latter inclusion holds to the question of whether NL $\subseteq \mathrm{L}^{\mathrm{PL}}$. Here the latter inclusion, NL $\subseteq \mathrm{L}^{\mathrm{PL}}$, trivially holds because NL $\subseteq$ PL. Can we use the isolation technique to prove NL $\subseteq \oplus$L/poly?

Pause to Ponder 4.18 *Does* NL $\subseteq \oplus$L/poly *hold?*

The answer to this question is in the affirmative. In fact, we can prove something stronger: NL/poly = UL/poly. In other words, if all nondeterministic logspace machines are given access to advice functions having polynomial length, then NL and UL are equivalent.

4.3.1 An NL-Complete Set

The Graph Accessibility Problem is the problem of deciding, for a given directed graph G and two nodes s and t of G, whether t is reachable from s in G. We consider a restricted version of the problem in which G has no self-loops, and s is the first node and t is the last node, where we order the nodes in G according to the adjacency matrix representation of G. More precisely, we consider the set, $\widehat{\mathrm{GAP}}$, of all $a_{11}a_{12}\cdots a_{1n}a_{21}a_{22}\cdots a_{2n}\cdots\cdots, a_{n1}a_{n2}\cdots a_{nn}$, $n \geq 2$, such that the diagonal elements $a_{11}, \ldots, a_{nn}$ are each 0, and n is reachable from 1 in G, where G is the directed graph whose adjacency matrix's (i,j)th element is a_{ij}. Since GAP, the Graph Accessibility Problem (without constraints on the numbering of the start and finish nodes, and without prohibiting self-loops), is well-known to be NL-complete, and since a logspace machine, given G, s, and t, can swap the names of the source node

s and 1, can swap the names of the sink node t and n, and can eliminate all self-loops, $\mathrm{GAP} \leq_m^L \widehat{\mathrm{GAP}}$. $\widehat{\mathrm{GAP}}$ is clearly in NL. Hence, our problem $\widehat{\mathrm{GAP}}$ is NL-complete too.

4.3.2 NL/poly = UL/poly

We show how to apply the Isolation Lemma (Lemma 4.1) to prove NL $\subseteq$ UL/poly. Suppose we wish to decide the membership in $\widehat{\mathrm{GAP}}$ of an arbitrary n-node directed graph without self-loops. Let our universe $\mathcal{U}(n)$ be the set of all potential edges in such a graph. Then $||\mathcal{U}(n)|| = n(n-1)$. Let our weight functions map each edge in $\mathcal{U}(n)$ to an integer between 1 and $2n^3$. For a given n-node directed graph G without self-loops, and for each i, $1 \leq i \leq n$, define $\mathcal{F}(n,G)_i$ to be the set of all simple paths in G from 1 to i. We view each element of $\mathcal{F}(n,G)_i$ as a subset of $\mathcal{U}(n)$. Since $\mathcal{F}(n,G)_i$ is a collection of simple paths from 1 to i, no two elements in $\mathcal{F}(n,G)_i$ specify identical paths. Then a weight function W is good for $\mathcal{F}(n,G)_i$ if and only if the minimum-weight path in G from 1 to i with respect to W is unique. Now apply Lemma 4.1 with $m = n$, $\mathcal{U} = \mathcal{U}(n)$, $\mathcal{Z} = \mathcal{Z}(n)$, $\mathcal{F}_1 = \mathcal{F}(n,G)_1, \ldots, \mathcal{F}_n = \mathcal{F}(n,G)_n$, $D = n(n-1)$, $R = 2n^3 > 2mD$, and $\alpha = \frac{1}{2}$. Let $\mathcal{Z}(n)$ be the set of all weight functions whose values are at most $2n^3$. Then we have the following lemma.

Lemma 4.19 *Let $n \geq 2$ and let G be an n-node directed graph. Let $\mathcal{U}(n)$, $\mathcal{Z}(n)$, and $\mathcal{F}(n,G)_1, \ldots, \mathcal{F}(n,G)_n$ be as stated above. Then more than half of the edge-weight functions in $\mathcal{Z}$ are good for $\mathcal{F}(n,G)_1, \ldots, \mathcal{F}(n,G)_n$.*

Suppose we wish to select, for each $n \geq 2$, a sequence of some $m(n)$ weight functions, $W_1, \ldots, W_{m(n)} \in \mathcal{Z}(n)$, such that for all n-node directed graphs G, there is some i, $1 \leq i \leq m(n)$, such that W_i is good for $\mathcal{F}(n,G)_1, \ldots, \mathcal{F}(n,G)_n$. How large $m(n)$ should be? The following lemma states that $m(n)$ can be as small as n^2.

Lemma 4.20 *Let $n \geq 2$. Let $\mathcal{U}(n)$, $\mathcal{Z}(n)$, and $\mathcal{F}(n,G)_1, \ldots, \mathcal{F}(n,G)_n$ be as stated above. There is a collection of edge-weight functions $W_1, \ldots, W_{n^2}$ in $\mathcal{Z}(n)$ such that, for every n-node directed graph without self-loops, G, there is some k, $1 \leq k \leq n^2$, such that W_k is good for $\mathcal{F}(n,G)_1, \ldots, \mathcal{F}(n,G)_n$.*

Proof of Lemma 4.20 Let $n \geq 2$. By Lemma 4.19, for every n-node directed graph without self-loops, G, the proportion of edge-weight functions in $\mathcal{Z}(n)$ that are good for $\mathcal{F}(n,G)_1, \ldots, \mathcal{F}(n,G)_n$ is more than a half. So, for all n-node directed graphs without self-loops, the proportion of $(W_1, \ldots, W_{n^2})$ such that for all k, $1 \leq k \leq n^2$, W_k is bad for $\mathcal{F}(n,G)_1, \ldots, \mathcal{F}(n,G)_n$ is less than 2^{-n^2}. There are $2^{n(n-1)}$ directed n-node directed graphs without self-loops. So, the proportion of $(W_1, \ldots, W_{n^2})$ such that, for some n-node directed graph without self-loop G, for all i, $1 \leq i \leq n^2$, W_i is bad for $\mathcal{F}(n,G)_1, \ldots, \mathcal{F}(n,G)_n$ is less than $2^{n(n-1)}2^{-n^2} < 1$. This implies that

there is some $\langle W_1, \ldots, W_{n^2} \rangle$ such that for all directed n-node directed graph without self-loop G, there is some i, $1 \leq i \leq n^2$, such that W_i is good for $\mathcal{F}(n, G)_1, \ldots, \mathcal{F}(n, G)_n$. ❑ Lemma 4.20

We define our advice function h as follows. For every $n \geq 2$, h maps each string of length n to a fixed collection $\langle W_1, \ldots, W_{n^2} \rangle$ of n^2 legitimate weight functions possessing the property in Lemma 4.20. For $n = 1$, h maps each string of length n to the empty string. The domain size D is $n(n-1)$ and the largest weight R is $2n^3$. So, by encoding each weight in binary, the encoding length of $h(n)$ will be $\mathcal{O}(n^4 \log n)$.

Now we prove the following theorem.

Theorem 4.21 NL $\subseteq$ UL/poly*, and thus,* NL/poly = UL/poly*.*

In fact, we will prove the following result, from which Theorem 4.21 immediately follows.

Theorem 4.22 *There is a* UL *machine that solves* $\widehat{\text{GAP}}$ *using a polynomially length-bounded advice function.*

Proof For simplicity, in the following, let $n \geq 2$ be fixed and let $W_1 \cdots W_{n^2}$ be the advice for length n (i.e., what the advice function gives, call it h). Also, let G be an n-node graph G whose membership in $\widehat{\text{GAP}}$ we are testing.

We need to define some notions and notation. For each i, $1 \leq i \leq n^2$, and j, $1 \leq j \leq n$, define $\mathit{MinWeight}(i, j)$ to be the weight of the minimum-weight paths from 1 to j with respect to the weight function W_i; if j is not reachable from 1 in G, then $\mathit{MinWeight}(i, j) = \infty$. For each i, $1 \leq i \leq n^2$, and $d \geq 0$, define $\mathit{Reach}(i, d)$ to be the set of all nodes j, $1 \leq j \leq n$, that are reachable from 1 via paths of weight at most d with respect to the weight function W_i, define $\mathit{Count}(i, d) = ||\mathit{Reach}(i, d)||$, and define $\mathit{WeightSum}(i, d) = \sum_j \mathit{MinWeight}(i, j)$, where j ranges over all elements in $\mathit{Reach}(i, d)$; also, we say that W_i is *d-nice* if, for every $j \in \mathit{Reach}(i, d)$, there is a unique minimum-weight path from 1 to j in G with respect to W_i.

Due to our construction of h, every minimum-weight path has weight at most $n(2n^3) = 2n^4$. So, for every i, $1 \leq i \leq n^2$, and for every d, $d \geq 2n^4$, it holds that $\mathit{Reach}(i, d) = \mathit{Reach}(i, d+1)$, $\mathit{Count}(i, d) = \mathit{Count}(i, d+1)$, and $\mathit{WeightSum}(i, d) = \mathit{WeightSum}(i, d+1)$. Note that 1 is the only node that can be reached from 1 without traversing edges. So, for all i, $1 \leq i \leq n^2$, it holds that $\mathit{MinWeight}(i, 1) = 0$, $\mathit{Reach}(i, 0) = \{1\}$, $\mathit{Count}(i, 0) = 1$, $\mathit{WeightSum}(i, 0) = 0$, and W_i is 0-nice.

We prove that if W_i is d-nice and if we know $\mathit{Count}(i, d)$ and $\mathit{WeightSum}(i, d)$, then for any j, $1 \leq j \leq n$, we can test, via unambiguous logspace computation, whether j belongs to $\mathit{Reach}(i, d)$. Recall that in the previous section we presented a nondeterministic logspace procedure for guessing a path, π_j, from 1 to a given node j. Let us modify this procedure as follows:

- For each node j, $1 \leq j \leq n$, attempt to guess a path from 1 to j having weight at most d (with respect to W_i) and having length at most $n - 1$.

Count the number of j for which the guess is successful (call this number C) and compute the sum of $W_i(\pi_j)$ for all successful j (call this number S).
- Output "successful" if $C = \mathit{Count}(i,d)$ and $S = \mathit{WeightSum}(i,d)$. Output "failure" otherwise.

Note that if W_i is d-nice and both $\mathit{Count}(i,d)$ and $\mathit{WeightSum}(i,d)$ are correctly computed, then there is only one computation path in the above along which it holds that $C = \mathit{Count}(i,d)$ and $S = \mathit{WeightSum}(i,d)$. Furthermore, the space requirement for this procedure is $\mathcal{O}(\log n)$, since the guessing part can be sequential, $C \leq n$, and $S \leq n(2n^4) = 2n^5$.

Now modify this procedure further, so that (i) it takes a number j, $1 \leq j \leq n$, as an additional input, (ii) it memorizes whether the guess is successful for j, and if so, it memorizes the weight of the path it guesses, and (iii) if the computation is successful (namely, when $C = \mathit{Count}(i,d)$ and $S = \mathit{WeightSum}(i,d)$) it outputs the information that it has memorized in (ii). We call this modified version `ReachTest`. For a d-nice W_i, given $\mathit{Count}(i,d)$ and $\mathit{WeightSum}(i,d)$, `ReachTest`(j) behaves as an unambiguous logspace procedure. Since the modification does not change the space requirement, if W_i is $2n^4$-nice, then `ReachTest`(n) will discover, via unambiguous logspace computation, whether $G \in \widehat{\mathrm{GAP}}$.

Now we have only to develop a UL procedure for finding an i, $1 \leq i \leq n^2$, such that W_i is $2n^4$-nice, and for computing $\mathit{Count}(i, 2n^4)$ and $\mathit{WeightSum}(i, 2n^4)$ for that i. We design an inductive method for accomplishing this task. We vary i from 1 to n^2 and, for each i, we vary d from 0 to $2n^4$. For each combination of i and d, we test whether W_i is d-nice, and if the test is passed, we compute $\mathit{Count}(i,d)$ and $\mathit{WeightSum}(i,d)$. Note for every i, $1 \leq i \leq n^2$, and for every d, $0 \leq d < 2n^4$, that if W_i is not d-nice, then W_i is not $(d+1)$-nice. Thus, if we discover that W_i is not d-nice for some d, then we will skip to the next value of i without investigating larger values of d. Recall that for every i, $1 \leq i \leq n^2$, W_i is 0-nice, $\mathit{Reach}(i,0) = \{1\}$, $\mathit{Count}(i,0) = 1$, and $\mathit{WeightSum}(i,0) = 0$. Iterating the variables i and d requires only $\mathcal{O}(\log n)$ space. So, it suffices to prove that there is a UL procedure that given i, $1 \leq i \leq n^2$, and d, $0 \leq d \leq 2n^4 - 1$, such that W_i is d-nice, $\mathit{Count}(i,d)$, and $\mathit{WeightSum}(i,d)$, tests whether W_i is $(d+1)$-nice, and if so computes $\mathit{Count}(i,d+1)$ and $\mathit{WeightSum}(i,d+1)$. To obtain such an algorithm, the following fact is useful.

Fact 4.23 *Let $1 \leq i \leq n^2$ and $0 \leq d \leq 2n^4 - 1$. Suppose that W_i is d-nice. Then the following conditions hold:*

1. *For every u, $1 \leq u \leq n$, the condition $u \in \mathit{Reach}(i,d+1) - \mathit{Reach}(i,d)$ is equivalent to: $u \notin \mathit{Reach}(i,d)$ and there is some $v \in \mathit{Reach}(i,d)$ such that (v,u) is an edge of G and $\mathit{MinWeight}(i,v) + W_i(v,u) = d+1$.*
2. *W_i is $(d+1)$-nice if and only if for every $u \in \mathit{Reach}(i,d+1) - \mathit{Reach}(i,d)$ there is a unique node v such that $\mathit{MinWeight}(i,v) + W_i(v,u) = d+1$.*

Proof of Fact 4.23 Part 1 as well as the left to right direction of part 2 is straightforward. To prove the right to left direction of part 2, suppose W_i is not $(d+1)$-nice but is d-nice. Then there exists some $u \in \mathit{Reach}(i, d+1) - \mathit{Reach}(i, d)$ such that there are two distinct paths from 1 to u, with respect to W_i. Since $u \in \mathit{Reach}(i, d+1) - \mathit{Reach}(i, d)$, the weight of the two paths should be $d+1$. So, they are minimum-weight paths. ❑ Fact 4.23

Now we build a UL algorithm for the incremental steps. Let $1 \leq i \leq n^2$ and $1 \leq d \leq 2n^4$. Suppose that W_i is $(d-1)$-nice and that $\mathit{Count}(i, d-1)$ and $\mathit{WeightSum}(i, d-1)$ are known.

Step 1 Set counters c and s to 0.

Step 2 For each node u, $1 \leq u \leq n$, do the following:

(a) Call `ReachTest` to test whether $u \in \mathit{Reach}(i, d-1)$. Then
- if the `ReachTest` outputs "failure," then output "failure" and halt, else
- if `ReachTest` asserts that $u \in \mathit{Reach}(i, d-1)$, then skip to the next u, else
- if `ReachTest` asserts that $u \notin \mathit{Reach}(i, d-1)$, then proceed to (b).

(b) Set the counter t to 0, then for each node v such that (v, u) is an edge in G call `ReachTest` to test whether $v \in \mathit{Reach}(i, d-1)$ and
- if `ReachTest` returns "failure," then output "failure" and halt, else
- if `ReachTest` asserts that $v \in \mathit{Reach}(i, d-1)$ and $\mathit{MinWeight}(i, v) + W_i(v, u) = d$, then increment t.

Next,
- if $t = 0$, then move on to the next u without touching c or s, else
- if $t = 1$, then increment c and add d to s, else
- if $t > 1$, then assert that W_i is not d-nice and halt.

Step 3 Set $\mathit{Count}(i, d)$ to $\mathit{Count}(i, d-1) + c$, set $\mathit{WeightSum}(i, d)$ to $\mathit{WeightSum}(i, d-1) + s$, and halt.

The correctness of the algorithm follows from Fact 4.23. It is clear that the space requirement is $\mathcal{O}(\log n)$. Note that W_i being d-nice guarantees that `ReachTest`(i, d) produces exactly one successful computation path. So, there is a unique successful computation path of this algorithm, along which exactly one of the following two events occurs:

- We observe that W_i is $(d+1)$-nice and obtain $\mathit{Count}(i, d+1)$ and $\mathit{WeightSum}(i, d+1)$.
- We observe that W_i is not $(d+1)$-nice.

Thus, we can execute the induction step by a UL algorithm. Putting all together, we have a UL machine that decides $\widetilde{\mathrm{GAP}}$ with h (i.e., $W_1 W_2 \cdots W_{n^2}$ on inputs of length n) as the advice. This completes the proof of Theorem 4.22. ❑ Theorem 4.22

4.4 OPEN ISSUE: Do Ambiguous and Unambiguous Nondeterminism Coincide?

In Sect. 4.3 we showed that the classes NL and UL are equal under polynomial-size advice. Can we get rid of the polynomial-size advice, i.e., is NL equal to UL? One tempting approach would be to derandomize the Isolation Lemma, however no one has yet succeeded along that path. It is now known that the equivalence holds if there is a set in DSPACE[n] that requires circuits of size at least 2^{cn} for some $c > 0$. In particular, NL = UL if SAT requires circuits of size at least 2^{cn} for some $c > 0$.

Also, what can we say about the question of whether NP = UP? There is an oracle relative to which NP $\neq$ UP. Since there is also an oracle relative to which NP = UP (e.g., any PSPACE-complete set, as $\mathrm{NP}^{\mathrm{PSPACE}} = \mathrm{UP}^{\mathrm{PSPACE}} = \mathrm{PSPACE}$), relativizable proof techniques cannot settle the NP = UP question. In fact, it even remains an open question whether the assumption NP = UP implies a collapse of the polynomial hierarchy.

4.5 Bibliographic Notes

Part 2 of Proposition 4.6 is due to Ko [Ko82]. Lemma 4.9 is due to Schöning [Sch89]. Proposition 4.8 is due to Papadimitriou and Zachos [PZ83]. Regan [Reg85] first noted the closure of #P under addition. The observation applies to #L as well. Proposition 4.16 is due to Balcázar, Book, and Schöning ([BBS86], see also [Ang80,GJ79]). Buntrock et al. [BDHM92] proved that $\oplus\mathrm{L}^{\oplus\mathrm{L}} = \oplus\mathrm{L}$. The oracle, mentioned in Sect. 4.4, relative to which NP $\neq$ UP is due to Rackoff [Rac82].

The Isolation Lemma was established by Mulmuley, Vazirani, and Vazirani [MVV87]. Their version deals only with a single collection of sets. Our version, which deals with multiple collections of sets, is taken from the work of Gál and Wigderson [GW96].

Toda's Theorem is due to Toda [Tod91c], and Lemma 4.14, Theorem 4.5, Theorem 4.12, Corollary 4.10, and Corollary 4.17 are all from his seminal paper. Part 1 of Proposition 4.6 is often written as BPP = BP $\cdot$ P, where BP$\cdot$ is what is known as the BP quantifier or the BP operator. Heller and Zachos [ZH86] introduced this quantifier and first proved part 1 of Proposition 4.6. Corollary 4.7 generalizes Adleman's [Adl78] early result RP $\subseteq$ P/poly, and can be found in Bennett and Gill [BG81] and Schöning [Sch86b]. One other form of Toda's Theorem is $\mathrm{PP}^{\mathrm{PH}} \subseteq \mathrm{BP} \cdot \mathrm{PP}$.

Toda and Ogiwara [TO92] showed that $\mathrm{C_{=}P}^{\mathrm{PH}} \subseteq \mathrm{BP} \cdot \mathrm{C_{=}P}$ and, for every $k \geq 2$, that $\mathrm{Mod}_k\mathrm{P}^{\mathrm{PH}} \subseteq \mathrm{BP} \cdot \mathrm{Mod}_k\mathrm{P}$. Tarui [Tar93] showed that R$\cdot$, the one-sided error version of BP$\cdot$, can replace the BP$\cdot$ on the right-hand side regarding PP and $\mathrm{C_{=}P}$, i.e., $\mathrm{PP}^{\mathrm{PH}} \subseteq \mathrm{R} \cdot \mathrm{PP}$ and $\mathrm{C_{=}P}^{\mathrm{PH}} \subseteq \mathrm{R} \cdot \mathrm{C_{=}P}$.

Gupta [Gup93] showed that $\mathrm{R} \cdot \mathrm{C}_{=}\mathrm{P} = \mathrm{BP} \cdot \mathrm{C}_{=}\mathrm{P}$. Green et al. [GKR^{+}95] observed that PH is included in the class of decision problems whose memberships are determined by the "middle bit" of #P functions.

The isolation technique used in the celebrated paper by Toda is the one by Valiant and Vazirani [VV86]. Roughly speaking, in order to reduce the cardinality of an unknown nonempty set S of nonzero vectors in $\{0,1\}^n$, one applies a sequence of filters. Each filter can be written as $b \cdot x = 0$, where $\cdot$ is the inner product modulo 2 of n dimensional vectors and only vectors satisfying $b \cdot x = 0$ are passed through the filter. Valiant and Vazirani show that, for any nonempty $S \subseteq \{0,1\}^n - \{0^n\}$, if a sequence of n random filters $b_1, \cdots, b_n \in \{0,1\}^n$ is chosen, the probability that at some point i, $0 \leq i \leq n$, there is exactly one vector in S that pass through all the filters up to b_i is at least $\frac{1}{4}$. Thus with this technique, one needs quadratically many bits to achieve a constant success probability in order to reduce the cardinality to one. We may ask whether it is possible to use fewer bits to achieve a constant success probability. Naik, Regan, and Sivakumar [NRS95] developed a reduction scheme that uses a quasilinear number of bits to achieve constant success probability.

Toda's Theorem has applications in circuit theory. A translation of Theorem 4.5 into circuit classes is: AC^0 (the class of languages recognized by families of polynomial-size, constant-depth, unbounded fan-in AND-OR circuits) is included in the class of languages recognized by families of size $2^{\log^{\mathcal{O}(1)} n}$, depth-2 probabilistic circuits with a PARITY gate at the top and polylogarithmic fan-in AND gates at the bottom. This inclusion was first proven by Allender [All89a] using a different technique. Later, Allender and Hertrampf [AH94] showed that, for every prime number p, the class $\mathrm{ACC}_{\mathrm{p}}$—the class of languages recognized by families of polynomial size, constant-depth, circuits consisting of unbounded fan-in ANDs, unbounded fan-in ORs and unbounded fan-in MODULO p (computing whether the number of 1s in the inputs is not divisible by p) gates—is included in the following three classes of languages: (a) the class of languages recognized by depth-4, polylogarithmic bottom-fan-in, size $2^{\log^{\mathcal{O}(1)} n}$ circuits consisting of unbounded fan-in ANDs, unbounded fan-in ORs, and unbounded fan-in MODULO p gates; (b) the class of languages recognized by depth-3, polylogarithmic bottom-fan-in, size $2^{\log^{\mathcal{O}(1)} n}$ circuits consisting solely of MAJORITY gates (computing whether the majority of the inputs are 1s); and (c) the class of languages recognized by depth-2, polylogarithmic bottom-fan-in, size $2^{\log^{\mathcal{O}(1)} n}$ probabilistic circuits with a MODULO p gate at the top and AND gates at the bottom. They also showed that if the $\mathrm{ACC}_{\mathrm{p}}$ class is uniform, then the classes (a), (b), and (c) can be all uniform. Kannan et al. [KVVY93] independently proved the uniformity result regarding the inclusion involving class (c). Yao [Yao90] showed that the inclusion involving class (b) holds for nonprime modulo p as well. Tarui [Tar93] showed that AC^0 is included in the class of languages recognized by depth-2, polylogarithmic bottom-fan-in, size $2^{\log^{\mathcal{O}(1)} n}$ proba-

bilistic circuits with a MAJORITY gate at the top and AND gates at the bottom. Beigel and Tarui [BT94] showed that in the inclusion involving class (c), deterministic circuits suffice. Beigel, Reingold, and Spielman [BRS91] showed that the class of languages accepted by a constant depth, polynomial-size circuits with any symmetric gate (i.e., the output depending only on the number of 1s in the inputs) and unbounded fan-in ANDs and ORs elsewhere is included in the class of languages recognized by a depth-2, size $2^{\log^{\mathcal{O}(1)} n}$ circuits with essentially the same symmetric gate at the top and polylogarithmic fan-in AND gates at the bottom.

Another application of the isolation technique is the probability one "inclusion" of NPMV in $\mathrm{FP}_{tt}^{\mathrm{NP}}$. Watanabe and Toda [WT93] proved that with probability one relative to a random oracle NPMV (the class of multivalued nondeterministic polynomial-time functions) has a refinement (see Chap. 3) in $\mathrm{FP}_{tt}^{\mathrm{NP}}$, the class of functions computable in polynomial time with parallel access to an NP oracle.

Wigderson [Wig94] showed how to apply the Isolation Lemma to prove $\mathrm{NL} \subseteq \oplus\mathrm{L/poly}$. In an expanded version of that paper, Gál and Wigderson [GW96] asked whether there was any use of the multiple collection version of the Isolation Lemma. Allender and Reinhardt [RA99] gave an affirmative answer to that question by proving Theorem 4.22. Chari, Rohatgi, and Srinivasan [CRS95] developed an isolation method that uses fewer random bits than that of Mulmuley, Vazirani, and Vazirani. The results that relate the $\mathrm{NL} = \mathrm{UL}$ question to the circuit complexity of SAT and that of $\mathrm{DSPACE}(n)$, mentioned in Sect. 4.4 are by Allender, Reinhardt, and Zhou [ARZ99]. For the fact that "standard" graph accessibility problem is NL-complete, see [Sav70].