Complexity Theory
WS 2009/10

Prof. Dr. Erich Grädel

Mathematische Grundlagen der Informatik
RWTH Aachen

Contents

6 Complexity Theory for Probabilistic Algorithms

Probabilistic algorithms are algorithms that can, at certain points during their computation, choose one possibility for the next operation *at random* from a number of different possibilities. They can thus be seen as a modification of nondeterministic algorithms. The computation result of such an algorithm therefore is not a definite answer but a random variable: it depends on the decisions made "at random" during the computation. Please note that this has nothing to do with an assumption on the distribution of possible inputs. The probability does not concern the inputs but rather the decisions during the computation.

Probabilistic algorithms play an important role in many different areas. They are often simpler and more efficient than the best known deterministic algorithms for the same problem. Even more, some important areas such as algorithmic number theory or cryptology are inconceivable without probabilistic algorithms. We will look at two examples.

6.1 Examples of probabilistic algorithms

6.1.1 *Perfect matching and symbolic determinants*

We first recall the definition of the marriage problem. Given is a bipartite graph $G = (U, V, E)$ with two disjoint sets of nodes $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_n\}$ of the same size and a set of edges $E \subseteq U \times V$. The problem is to determine whether G permits a *perfect matching*, i.e., a subset $M \subseteq E$ such that for all $u \in U$ there is a $v \in V$ and for all $v \in V$ there is a $u \in U$ such that $(u, v) \in M$. We can rephrase the problem like this: Is there a permutation $\pi \in S_n$ so that $(u_i, v_{\pi(i)}) \in E$ for all $i \in \{1, \dots, n\}$?

The marriage problem can further be described as a problem over matrices and determinants. The graph $G = (U, V, E)$ is then characterised by a matrix A^G whose items are variables x_{ij} or 0.

$$A^G := (z_{ij})_{1 \le i,j \le n} \text{ with } z_{ij} := \begin{cases} x_{ij} & \text{if } (u_i, v_j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

The determinant of A^G is $\det A^G := \sum_{\pi \in S_n} \operatorname{sgn}(\pi) \prod_{i=1}^n z_{i\pi(i)}$ where $\operatorname{sgn}(\pi) = 1$ if π is the product of an even number of transpositions and $\operatorname{sgn}(\pi) = -1$ otherwise. Obviously, $\det A^G$ is a polynomial in $\mathbb{Z}[x_{11}, \dots, x_{nn}]$ (i.e., a polynomial with coefficients in $\mathbb{Z}$) of total degree n that is linear in every variable x_{ij}.

A permutation $\pi \in S_n$ defines a perfect matching if and only if $\prod_{i=1}^n z_{ij} \neq 0$. Since all of these products are pairwise different, we obtain

$$G \text{ allows a perfect matching} \iff \det A^G \neq 0.$$

Hence, if we were able to compute symbolic determinants (i.e., determinants of matrices that can contain variables) efficiently, we could use this to solve the marriage problem.

Determinants using Gauss elimination. We know from linear algebra how to compute determinants from numerical matrices: the given matrix is transformed (e.g., by interchanging lines or by adding linear combinations of lines to other lines) into a triangular matrix that has the same determinant. The products of the diagonal elements are then calculated to obtain the determinant. This requires $O(n^3)$ arithmetical operations. Further, the entries of the transformed matrices remain polynomially-bounded since they are subdeterminants of the given matrix.

Unfortunately, the application of this procedure to symbolic matrices is problematic. The entries of the transformed matrices are rational

functions in the entries of the original matrix and these functions generally have exponentially many terms. Even the problem whether a fixed mononom, e.g., $x_{11}x_{23}x_{31}$, appears in the determinant of A^G is NP-hard. Hence, Gauss elimination does not seem to be useful to calculate symbolic determinants.

However, we do not need to compute the determinant of A^G. It suffices to know whether it is 0 or not. The idea for the probabilistic algorithm solving the perfect matching problem is to substitute a tuple $\bar{a} = (a_{11}, \dots, a_{nn})$ of random numbers into the matrix A^G and then to calculate the determinant of the numerical matrix $A^G(\bar{a})$ using Gauss elimination.

If we obtain that $\det A^G(\bar{a}) \neq 0$ then the symbolic determinant $\det A^G$ is obviously not 0. The inverse does not hold: It might be the case that we incidentally find a root $\bar{a}$ of $\det A^G$ and, hence, obtain $\det A^G \neq 0$.

The following lemma allows us to control the probability to obtain the roots of a non-identically disappearing polynomial $\det A^G$ by finding a suitable set to choose $\bar{a}$ from.

Lemma 6.1. Let $p(x_1, \dots, x_n)$ be a polynomial such that $p \neq 0$ and every x_i is at most of degree d in p. Then, for every $m \in \mathbb{N}$,

$$|\{(a_1, \dots, a_n) \in \{0, \dots, m-1\}^n : p(a_1, \dots, a_n) = 0\}| \le ndm^{n-1}.$$

Proof. We will use induction over n. For $n = 1$, the induction hypothesis is a known fact: no polynomial $p \neq 0$ with one variable of degree d has more than d roots. Further, consider $n > 1$. We write $p(x_1, \dots, x_n)$ as a polynomial in x_n with coefficients from $\mathbb{Z}[x_1, \dots, x_{n-1}]$:

$$\begin{aligned} p(x_1, \dots, x_n) = p_0(x_1, \dots, x_{n-1}) + p_1(x_1, \dots, x_{n-1})x_n \\ + \dots + p_d(x_1, \dots, x_{n-1})x_n^d. \end{aligned}$$

Let now $p(a_1, \dots, a_n) = 0$ for $(a_1, \dots, a_n) \in \{0, \dots, m-1\}^n$. We consider two cases:

(a) $p_d(a_1, \dots, a_{n-1}) = 0$. By induction hypothesis, this is the case for at most $(n-1)dm^{n-2}$ tuples $(a_1, \dots, a_{n-1}) \in \{0, \dots, m-1\}^{n-1}$. Thus, there are at most $(n-1)dm^{n-1}$ roots $(a_1, \ldots, a_n) \in \{0, \ldots, m-1\}^n$ of p with $p_d(a_1, \ldots, a_{n-1}) = 0$.

(b) $p_d(a_1, \ldots, a_{n-1}) \neq 0$. Then, $p(a_1, \ldots, a_{n-1}, x_n)$ is a polynomial of degree d in variables x_n, for which there are at most d roots a_n. In addition to the roots in case (a), there are, hence, at most dm^{n-1} new roots.

Hence, we have at most ndm^{n-1} roots $(a_1, \ldots, a_n) \in \{0, \ldots, m-1\}^n$. Q.E.D.

Consequently, we obtain a probabilistic algorithm for the perfect matching problem.

```
Input: a matrix $A^G$ for a bipartite graph $G = (U, V, E)$, $|U| = |V| = n$
       a security parameter $k \in \mathbb{N}$
Set $m := 2n^2$
for $i = 1, \ldots, k$ do
    Choose at random numbers $a_{11}, \ldots, a_{nn} \in \{0, \ldots, m-1\}$
    Compute det $A^G(\bar{a})$ using Gauss elimination
    if det $A^G(\bar{a}) \neq 0$ then output 'There is a perfect matching'
endfor
output 'There is probably no perfect matching'
```

Since the computation of numerical determinants can be done in polynomial-time using Gauss elimination, this is also a polynomial-time algorithm. If the algorithm finds a tuple $\bar{a}$ such that $\det A^G \neq 0$, it will return 'There is a perfect matching' and this is correct. If it does not find such a $\bar{a}$ after k iterations, it will return 'There is probably no perfect matching'. This, however, is not always correct. The error probability, i.e., the probability that the algorithm does not find a non-root for a non-disappearing polynomial $\det A^G$, can be estimated using the above lemma.

Since $\det A^G$ is linear in each of the n^2 variables, the ratio of tuples $\bar{a} \in \{0, \ldots, m-1\}^{n^2}$ that are roots of $\det A^G$ is at most

$$\frac{n^2 d m^{n^2-1}}{m^{n^2}} = \frac{n^2 d}{m} = \frac{n^2}{2n^2} = \frac{1}{2}.$$

The probability to find only such tuples in k iterations is at most 2^{-k}. Please note this is not a probability statement with respect to bipartite graphs or symbolic determinants. It is indeed a statement on the error probability of a probabilistic algorithm with respect to its random decisions and is valid for all bipartite graphs.

6.1.2 A probabilistic prime number test

Two central problems of algorithmic number theory are the existence of polynomial algorithms for

(1) Primality testing: given an integer $n \in \mathbb{N}$, determine whether it is prime;
(2) Factoring: given an integer $n \in \mathbb{N}$, calculate its factorisation (its prime factors).

Primality testing has a long history going back to ancient Greece. The first systematic approach, the *Sieve of Eratosthenes*, where multiples of primes are successively removed from a list of numbers leaving only the primes, dates back to around 240 BC. While being based on multiplication only, this approach yields an algorithm that is still exponential in the size of the input like the naïve approach.

Obviously, PRIMES $\in$ coNP since each non-trivial factor is a polynomial witness for compositeness. In 1974, Pratt could prove membership in NP with some more effort.

A year later, Miller presented a deterministic polynomial-time algorithm based on Fermat's Little Theorem, but its correctness depends on the assumption of the Extended Riemann Hypothesis. In 1980, Rabin modified this test and obtained an unconditional but randomised polynomial-time algorithm, thus placing the problem in coRP. Later, in 1987, Adleman and Huang proved the quite involved result that PRIMES $\in$ RP, and hence in ZPP.

Only recently, Agrawal, Kayal and Saxena presented a deterministic polynomial-time algorithm based on a generalisation of Fermat's Little Theorem. The first version of their algorithm had a running-time in $O(n^{12})$, which could be improved to $O(n^{7.5})$, and lately to $O(n^6)$. Depending on some number-theoretic hypotheses, the running time might be further improved to $O(n^3)$. For details see [Agrawal, Kayal, Saxena. PRIMES is in P. Annals of Mathematics 160 (2004)].

However, the currently long running-time renders this algorithm practically unusable since it is outperformed by the simple and efficient probabilistic methods which are able to determine with an almost arbitrarily high probability whether a given number is prime.

Unfortunately, neither of these methods can be used to efficiently obtain a factorisation for composite numbers. In fact, it is widely assumed that the factorisation of integers is difficult in practice, and many modern public-key cryptology systems are based on this assumption.

In the following we will present the randomised primality test due to Rabin and Miller which is based on *Fermat's Little Theorem*.

Definition 6.2. For $n \in \mathbb{N}$, let

$$\mathbb{Z}_n^* := \{a \in \{1, \dots, n-1\} : \gcd(a, n) = 1\}.$$

Note that $(\mathbb{Z}_n^*, \cdot \pmod{n})$ is a group.

Theorem 6.3 (Fermat)**.** Let p be prime. Then, for all $a \in \mathbb{Z}_p^*$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof. Let $f(p,a)$ be the number of different non-periodic colourings of cycles of length p with a colours. Since p is prime and the period must be a divisor of p for every *periodic* colouring, only periods of length 1 are possible, that is, only monochrome colourings. The number of colourings of p nodes with a colours is a^p, the number of monochrome colourings is a and, hence, $f(p,a) = (a^p - a)/p = a(a^{p-1} - 1)/p$. We obtain that p is a divisor of $a^{p-1} - 1$ and therefore, $a^{p-1} \equiv p \pmod{p}$. Q.E.D.

One might hope that also the inverse holds, i.e., for every *composed* number n, there is an $a \in \mathbb{Z}_n^*$ such that $a^{n-1} \not\equiv 1 \pmod{n}$. If one could show furthermore that there are "many" $a \in \mathbb{Z}_n^*$ with this property, a prime number test could work as follows: Given some n, it would choose an $a \in \mathbb{Z}_n^*$ at random. Then, it would check whether $a^{n-1} \equiv 1$

$\pmod{n}$. For this approach to work, we need to be able to verify whether $a^{n-1} \not\equiv 1 \pmod{n}$ in polynomial time (with respect to the length of the input, i.e., $\log n$). This can be done by repeating the square operation modulo n: For $k = \lfloor \log n \rfloor$, compute the numbers $b_0, \dots, b_k$ with $b_0 := a$, $b_{i+1} := (b_i)^2 \pmod{n}$, i.e., $b_i = a^{2^i} \pmod{n}$. Let $n - 1 = \sum_{i=1} u_i 2^i$ be the binary representation of $n-1$, with $u_i \in \{0,1\}$. Then,

$$a^{n-1} = a^{\sum_i u_i 2^i} = \prod_i a^{u_i 2^i} \equiv \prod_{u_i=1} b_i \pmod{n}.$$

Unfortunately, the Fermat test in this simple form fails. This is because the inversion of Fermat's Little Theorem is incorrect. There are (even infinitely many) composites $n \in \mathbb{N}$ such that $a^{n-1} \equiv 1 \pmod{n}$ for all $a \in \mathbb{Z}_n^*$. These numbers are called *Carmichael numbers*. The first Carmichael numbers are 561 and 1729.

The idea works, however, for every non-Carmichael number. For $n \in \mathbb{N}$, let

$$F_n := \{a \in \mathbb{Z}_n^* : a^{n-1} \equiv 1 \pmod{n}\}.$$

Lemma 6.4. If n is composite and not a Carmichael number, then $|F_n| \leq |Z_n^*|/2$.

Proof. It is easy to see that $(F_n, \cdot \pmod{n})$ is a subgroup of $(\mathbb{Z}_n^*, \cdot \pmod{n})$. Since n is neither prime nor a Carmichael number, $F_n \subsetneq Z_n^*$. The order of a subgroup is always a divisor of the order of the group, i.e., $|\mathbb{Z}_n^*| = q|F_n|$ for some $q \geq 2$. Q.E.D.

Hence, the fact that our original idea for a prime number test does not work is simply due to the Carmichael numbers. It is, however, possible to refine the Fermat test and treat Carmichael numbers properly. There are two variants of such probabilistic primality tests, the Solovay-Strassen test and the Rabin-Miller test, which will be described in the following. It is based on the following observation.

Lemma 6.5. Let p be prime. Then, for all $a \in \mathbb{Z}_p^*$ if $a^2 \equiv 1 \pmod{p}$, then $a \equiv \pm 1 \pmod{p}$.

Proof. If p is prime, then $(\mathbb{Z}_p^*, + \pmod{n}, \cdot \pmod{n})$ is a field and fields have only the trivial roots 1 and -1. Q.E.D.

Theorem 6.6.

(i) The Rabin-Miller primality test (Algorithm 6.1) can be performed in polynomial-time (with respect to $\log n$).
(ii) If n is prime, the test always returns "n is probably prime".
(iii) If n is composite, the test returns "n is composite" with a probability of $\geq 1 - 2^{-k}$.

Hence, the result "n is composed" is always correct, and the answer "n is probably prime" means that n is indeed prime with a very high probability.

Proof. Proposition (i) is obviously correct. Proposition (ii) results from Theorem 6.3 and Lemma 6.5. If n is prime, then for all a used in the test:

- $a^{n-1} \equiv 1 \pmod{n}$.
- $b_j \not\equiv 1 \pmod{n}$ but $b_{j+1} = (b_j)^2 \equiv 1 \pmod{n}$.
  Hence, $b_j \equiv -1 \pmod{n}$

We obtain that the test returns "n is probably prime".

Algorithm 6.1. The Rabin-Miller primality test

```
Input: an odd number $n \in N$
       a security parameter $k$
Compute $t, w$ such that $n - 1 = 2^t w$ with $w$ odd
for $k$ times do
    Choose $a \in \{1, \ldots, n-1\}$ at random
    Compute $b_i := a^{2^i w} \pmod{n}$ for $i = 0, \ldots, t$
    if $b_t = a^{n-1} \not\equiv 1 \pmod{n}$ then output "$n$ is composite"
    Determine $j := \max\{i : b_i \not\equiv 1 \pmod{n}\}$
    if $b_j \not\equiv -1 \pmod{n}$ then output "$n$ is composite"
endfor
output "$n$ probably prime"
```

As for Proposition (iii), let M_n be the set of all $a \in \{1, \ldots, n-1\}$ such that the choice of a by the Rabin-Miller test with input n does *not* lead to the result "n is composed". It obviously suffices to show that $|M_n| \leq (n-1)/2$ for all composed, odd n. The probability to obtain only elements $a \in M_n$ when choosing k times some random a is smaller than 2^{-k}.

We see that $M_n \subseteq \mathbb{Z}_n^*$. If indeed $a \in M_n$ then $a^{n-1} \equiv 1 \pmod{n}$ and, hence, $a^{n-2}a + rn = 1$ for a suitable $r \in \mathbb{Z}$. If a and n had a common divisor $q > 1$, it would also be a divisor of the sum $a^{n-2}a + rn$ which is impossible since it is equal to 1. Therefore, a and n are co-prime and thus $a \in \mathbb{Z}_n^*$. Hence, it suffices to show the following.

Claim 6.7. There is a proper subgroup $U_n < Z_n^*$ which contains M_n.

From this, we obtain $|M_n| \leq |U_n| \leq |\mathbb{Z}_n^*|/2 \leq (n-1)/2$.

For composed non-Carmichael numbers n, the claim follows directly from Lemma 6.4 since $M_n \subseteq F_n$. For Carmichael numbers, we first show that these are not powers of primes, i.e., every Carmichael number n can be written as the product of two co-prime odd numbers n_1, n_2. Fix such an $n = n_1 \cdot n_2$.

For every $a \in M_n$, the sequence $b_0, \ldots, b_t$ (with $b_i = a^{2^i w} \pmod{n}$) has the form

$$* * * \cdots * \; -1 1 1 \cdots 1 \;\text{ or }\; 1 1 \cdots 1.$$

Set

$$h := \max\{i : 0 \leq i \leq t, \text{there is an } a \in Z_n^* \text{ with } a^{2^i w} \equiv -1 \pmod{n}\}.$$

Such an h exists since, for example, $(-1)^{2^0 w} = -1$. Let now

$$U_n := \{a \in \mathbb{Z}_n^* : a^{2^h w} \equiv \pm 1 \pmod{n}\}.$$

Obviously, U_n is a subgroup of Z_n^* containing M_n. We now show that $U_n \subsetneq Z_n^*$ as follows: Let $b \in Z_n^*$ such that $b^{2^h w} \equiv -1 \pmod{n}$. By the Chinese Remainder Theorem, there is an $a \in \mathbb{Z}_n^*$ such that

(1) $a \equiv b \pmod{n_1}$, and
(2) $a \equiv 1 \pmod{n_2}$.

We show that $a \notin U_n$ by leading the claim $a \in U_n$ to a contradiction.

At first, let us consider $a \in U_n$ since $a^{2^h w} \equiv 1 \pmod{n}$. Then, also $a^{2^h w} \equiv 1 \pmod{n_1}$. However, because of (1) $a^{2^h w} \equiv b^{2^h w} \equiv -1 \pmod{n_1}$ holds, which is impossible since $n_1 > 2$.

The other possibility is that $a \in U_n$ since $a^{2^h w} \equiv -1 \pmod{n}$. Then, $a^{2^h w} \equiv -1 \pmod{n_2}$. However, because of (2) $a^{2^h w} \equiv 1 \pmod{n_2}$, which is impossible since $n_2 > 2$. Q.E.D.

Miller showed that, under the assumption of the *Extended Riemann Hypothesis* (ERH), this test yields a deterministic polynomial-time algorithm witnessing PRIMES $\in$ P.

Theorem 6.8 (Miller)**.** The ERH implies that there is a function $f : \mathbb{N} \to \mathbb{N}$ such that $f(n)$ is bounded by a polynomial in $\log n$ such that, for all odd *non-prime* numbers $n > 2$, one of the following is true:

(i) n is a prime power;
(ii) there is an $a < f(n)$ with $a \notin M_n$, i.e., the use of a in the Rabin-Miller test on input n leads to the result "n is composed".

Corollary 6.9. The ERH implies PRIMES $\in$ P.

6.2 Probabilistic complexity classes and Turing machines

For $m \in \mathbb{N}$, we consider $\{0,1\}^m$ as a *probability space* with *uniform distribution*: For every $u \in \{0,1\}^m$, the probability

$$\Pr_{y\in\{0,1\}^m}[y = u] = \frac{1}{2^m}.$$

Definition 6.10. A *probabilistic Turing machine* (PTM) is a Turing machine whose input consists of a pair $(x,y) \in \Sigma^* \times \{0,1\}^*$. Here, $x \in \Sigma^*$ denotes the actual input and $y \in \{0,1\}^*$ a random word controlling the computation of the machine.

A PTM M is called $p(n)$-time bounded if M stops after at most $p(|x|)$ steps on input (x,y). Without loss of generality, we can assume that $|y| = p(|x|)$.

Let M be $p(n)$-time bounded. If we consider M as an acceptor over $\Sigma^* \times \{0,1\}^*$, we obtain the language $L(M) \subseteq \Sigma^* \times \{0,1\}^*$. Hence, we define M as a *probabilistic acceptor* over Σ^*. For $x \in \Sigma^*$ with $|x| = n$, we set:

$$\begin{aligned}\Pr[M \text{ accepts } x] &:= \Pr_{y\in\{0,1\}^{p(n)}}[(x,y) \in L(M)] \\ &= \frac{|\{y \in \{0,1\}^{p(n)} : (x,y) \in L(M)\}|}{2^{p(n)}}.\end{aligned}$$

Lemma 6.11. A language $A \subseteq \Sigma^*$ is in NP if and only if there is a polynomial PTM M such that $A = \{x \in \Sigma^* : \Pr[M \text{ accepts } x] > 0\}$.

Proof. Consider $A \in$ NP. Then, there is a $B \in$ P and a polynomial $p(n)$ such that $A = \{x \in \Sigma^* : \exists y\ (|y| \le p(|x|) \wedge (x,y) \in B\}$. It is not difficult to modify B and $p(n)$ in a way that $A = \{x \in \Sigma^* : (\exists y \in \{0,1\}^{p(|x|)})\ (x,y) \in B\}$. Let M be a polynomial, deterministic TM over $\Sigma^* \times \{0,1\}^*$ with $L(M) = B$. If we consider M as a probabilistic TM over Σ^*, we obtain:

$$\begin{aligned}A &= \{x \in \Sigma^* : \Pr_{y\in\{0,1\}^{p(n)}}[(x,y) \in L(M)] > 0\} \\ &= \{x \in \Sigma^* : \Pr[M \text{ accepts } x] > 0\}.\end{aligned}$$

Consider now $A = \{x \in \Sigma^* : \Pr[M \text{ accepts } x] > 0\}$ for a polynomial PTM M. Hence, for some suitable polynomial p, $A = \{x \in \Sigma^* : \Pr_{y\in\{0,1\}^{p(n)}}[(x,y) \in L(M)] > 0\}$. Then, $B := \{(x,y) \in \Sigma^* \times \{0,1\}^{p(|x|)} : (x,y) \in L(M)\}$ in P and therefore, $A = \{x \in \Sigma^* : \exists y\ (|y| \le p(|x|) \wedge (x,y) \in B\}$ in NP. Q.E.D.

The probability to find a suitable witness y for an NP problem on input x simply by guessing can be very small. "Good" probabilistic algorithms are successful in guessing, i.e., they guess suitable witnesses with a high probability. We call a probabilistic algorithm for A stable, if $\Pr[M \text{ accepts } x]$ for $x \in A$ is *significantly* larger than $\Pr[M \text{ accepts } x]$ for $x \notin A$.

Definition 6.12. Consider a language $A \subseteq \Sigma^*$.

- $A \in \mathrm{PP}$ (**p**robabilistic **p**olynomial time), if there is a polynomial PTM M such that
$$A = \{x : \Pr[M \text{ accepts } x] > \frac{1}{2}\}.$$
- $A \in \mathrm{BPP}$ (**b**ounded error **p**robabilistic **p**olynomial time), if there is a polynomial PTM M such that
$$x \in A \implies \Pr[M \text{ accepts } x] \geq \frac{2}{3} \text{ and}$$
$$x \notin A \implies \Pr[M \text{ accepts } x] \leq \frac{1}{3}.$$

Probabilistic algorithms are subject to two kinds of *error probabilities*:

(1) *Incorrect positive:* $x \notin A$ but $\Pr[M \text{ accepts } x] > 0$.
(2) *Incorrect negative:* $x \in A$ but $\Pr[M \text{ accepts } x] < 1$, i.e., $\Pr[M \text{ does not accept } x] = 1 - \Pr[M \text{ accepts } x] > 0$.

We obtain the following picture for the complexity classes defined so far:

BPP:	both error probabilities $\leq \frac{1}{3}$,
PP:	only the trivial bound, error probability $\leq \frac{1}{2}$, that can be obtained by tossing a coin,
NP:	no incorrect positive error, but $\Pr[M \text{ accepts } x]$ for $x \in A \subseteq \mathrm{NP}$ can be arbitrarily small.

Definition 6.13. In addition to PP and BPP, the notion of error probability leads us to the following probabilistic complexity classes:

- $A \in \mathrm{RP}$ (**r**andom **p**robabilistic **p**olynomial time), if there is a polynomial PTM M such that
$$x \in A \Longrightarrow \Pr[M \text{ accepts } x] \geq \tfrac{2}{3} \text{ and}$$
$$x \notin A \Longrightarrow \Pr[M \text{ accepts } x] = 0.$$
(no incorrect positive results).

- $A \in \text{Co-RP} :\Longleftrightarrow \overline{A} \in \mathrm{RP}$, i.e., there is a polynomial PTM M such that
$$x \in A \Longrightarrow \Pr[M \text{ accepts } x] = 1 \text{ and}$$
$$x \notin A \Longrightarrow \Pr[M \text{ accepts } x] \leq \tfrac{1}{3}.$$
(no incorrect negative results).
- $A \in \mathrm{ZPP}$ (**z**ero-error **p**robabilistic **p**olynomial time), if $A \in \mathrm{RP}$ and $A \in \text{Co-RP}$.

For the interpretation of ZPP, we consider a language $A \in \mathrm{ZPP}$. Then, there are polynomial PTM M^+ and M^- for $A \in \mathrm{RP}$ and $\overline{A} \in \mathrm{RP}$. Consider now a PTM M that simulates the computations of M^+ and M^- in parallel and accepts the input if M^+ accepts it and rejects if M^- accepts it. In the case that M^+ rejects and M^- accepts, M returns "'don't know"'. Obviously, M is working error-free, i.e., the answers are always correct. It does, however, return an unsatisfying result with a probability of $\varepsilon \leq 1/3$. By repeating with independent random inputs, ε can be made arbitrarily small.

Example 6.14. The Rabin-Miller primality test (RM) shows that PRIMES $\in$ coRP. In 1987, Adleman and Huang have shown (the much more difficult result) that PRIMES is also in RP. Hence, PRIMES $\in$ ZPP.

Obviously, the following inclusions hold:

$$\begin{tikzcd}
 & & \mathrm{RP} \arrow[dr, phantom, "\subseteq" sloped] & & \\
P \arrow[r, phantom, "\subseteq"] & \mathrm{ZPP} \arrow[ur, phantom, "\subseteq" sloped] \arrow[dr, phantom, "\subseteq" sloped] & & \mathrm{BPP} \arrow[r, phantom, "\subseteq"] & \mathrm{PP} \\
 & & \text{Co-RP} \arrow[ur, phantom, "\subseteq" sloped] & &
\end{tikzcd}$$

Furthermore, $\mathrm{RP} \subseteq \mathrm{NP}$, $\text{Co-RP} \subseteq \text{Co-NP}$ and $\mathrm{ZPP} \subseteq \mathrm{NP} \cap \text{Co-NP}$.

Theorem 6.15. $\mathrm{NP} \subseteq \mathrm{PP} \subseteq$ PSPACE.

Proof. Consider $A \in \mathrm{NP}$. By Lemma 6.11, there is a PTM M with $A = \{x \in \Sigma^* : \Pr[M \text{ accepts } x] > 0\}$. Let M' be a PTM accepting $(x, y_0 y_1 y_2 \ldots)$ if, and only if, either $y_0 = 1$ or M accepts $(x, y_1 y_2 \ldots)$.

Then,

$$\Pr[M' \text{ accepts } x] = \frac{1}{2} + \frac{1}{2}\Pr[M \text{ accepts } x],$$

and we obtain $A = \{x : \Pr[M' \text{ accepts } x] > \frac{1}{2}\} \in \mathrm{PP}$.

On the other hand, consider $A \in \mathrm{PP}$. Then,

$$x \in A \iff \Pr[M \text{ accepts } x] = \Pr_{y\in\{0,1\}^{p(n)}}[(x,y) \in L(M)] > \frac{1}{2}.$$

Therefore, for some given input x, all computations of M on input (x,y) with $y \in \{0,1\}^{p(n)}$ can be simulated using polynomial space to determine whether more than $2^{p(n)-1}$ of the pairs (x,y) are accepted.

Q.E.D.

Note that the relation between BPP and NP remains unclear.

In the following, we introduce a method to reduce the error probability of a BPP algorithm. Here, the fundamental idea is to use k iterations and then to decide for the most frequent result obtained.

Let M be a $p(n)$-time bounded PTM with an error probability $\le \varepsilon < \frac{1}{2}$. Let M^k be a PTM accepting $(x, y_1y_2\dots y_k)$ with $y_i \in \{0,1\}^{p(n)}$ if and only if $|\{i : (x,y_i) \in L(M)\}| \ge k/2$. The algorithm M^k is polynomial if $k = k(n)$ is polynomial in n.

To compute the error probability of M^k, we need a result from probability theory.

Let $X_1,\dots,X_k$ be random variables over $\{0,1\}$ with $\Pr[X_i = 1] = p$ and $\Pr[X_i = 0] = 1-p$ for $0 < p < 1$ (Bernoulli random variables). The sum $X = \sum_{i=1}^k X_i$ is a binomially distributed random variable over $\mathbb{N}$. Its expectation is $E(X) = p\cdot k$. The following lemma gives a probability estimate for the case that the value of X differs less than d from the expectation:

Lemma 6.16 (Chernoff)**.** For $d \ge 0$,

$$\Pr[X - pk \ge d] \le e^{-\frac{d^2}{4kp(1-p)}} \le e^{-\frac{-d^2}{k}} \text{ and } \Pr[pk - X \ge d] \le e^{-\frac{-d^2}{k}}.$$

Coming back to our original problem, for $\bar{y} = y_1\dots y_k$ (with $y_i \in \{0,1\}^{p(n)}$), we define the random variables

$$X_i(\bar{y}) := \begin{cases} 1 & \text{if } (x,y_i) \in L(M), \\ 0 & \text{otherwise.} \end{cases}$$

Let A be a language that is decided by a BPP algorithm M with an error probability $\le \varepsilon < \frac{1}{2}$. Then,

(1) For $x \notin A$, $p := \Pr[X_i = 1] \le \varepsilon$. With $X := \sum_{i=1}^k X_i$,

$$\Pr[M^k \text{ accepts } x] = \Pr[X \ge \frac{k}{2}].$$

Applying Chernoff's Lemma, we obtain

$$\Pr[X \ge k/2] = \Pr[X - pk \ge k/2 - pk] \le e^{-(\frac{1}{2}-p)^2k} = 2^{-\Omega(k)}.$$

Let $q(n)$ be a suitable polynomial. For $k \ge c\cdot q(n)$ (c is a suitable constant), we obtain an incorrect positive error probability $\le 2^{-q(n)}$.

An analogous statement holds for incorrect positive error probability:

(2) For $x \in A$, $\Pr[M \text{ accepts } x] = \Pr[X_i = 1] = p \ge 1-\varepsilon$ and

$$\begin{aligned}\Pr[M^k \text{ does not accept } x] &= \Pr\left[X < \frac{k}{2}\right] \\ &= \Pr\left[pk - X \ge (p - \frac{1}{2})k\right] \\ &\le e^{-(p-\frac{1}{2})^2k} = 2^{-\Omega(k)}.\end{aligned}$$

Hence, we have shown:

Theorem 6.17. For every language $A \in \mathrm{BPP}$ and every polynomial $q(n)$, there is a polynomial PTM M accepting A with an error probability $\le 2^{-q(n)}$, i.e.,

$$\begin{aligned} x \in A &\implies \Pr[M^k \text{ accepts } x] \ge 1 - 2^{-q(|x|)}, \\ x \notin A &\implies \Pr[M^k \text{ accepts } x] \le 2^{-q(|x|)}. \end{aligned}$$

An analogous statement also holds for the class RP.

From Theorem 6.17, we obtain an interesting result concerning the relationship between BPP and circuit complexity.

Let M be some BPP algorithm for A with an error probability of $\le 2^{-q(n)}$. For all x,

$$\frac{|\{y \in \{0,1\}^{p(n)} \;:\; (x,y) \in L(M) \iff x \in A\}|}{2^{p(n)}} \ge 1 - 2^{-q(n)}.$$

It follows that for every fixed input length n, there are random values $y \in \{0,1\}^{p(n)}$ returning the correct result for *all* $x \in \Sigma^n$:

$$|\{y \in \{0,1\}^{p(n)} \;:\; y \text{ "bad" for at least one } x \in \Sigma^n\}| = \sum_{|x|=n} |\{y \in \{0,1\}^{p(n)} \;:\; y \text{ "bad" for } x\}| \le |\Sigma^n| \cdot 2^{-q(n)} \cdot 2^{p(n)}.$$

If $q(n)$ is chosen such that $\lim_{n\to\infty} |\Sigma^n| \cdot 2^{-q(n)} = 0$ (e.g., $q(n) = n^2$, or $q(n) = cn$ with $c \ge \log|\Sigma|$), we obtain that for large n at least one $y(n) \in \{0,1\}^{p(n)}$ gives the correct result for all $x \in \Sigma^n$;

Hence, there is a function $f : \mathbb{N} \to \{0,1\}^*$ with the following properties:

- f is polynomially-bounded: $|f(n)| = p(n)$ and
- for all sufficiently long $x \in \Sigma^*$,

$$x \in A \iff \underbrace{(x, f(x)) \in L(M)}_{\text{polynomial}}.$$

Definition 6.18. $A \in \Sigma^*$ is *non-uniform polynomially-decidable* ($A \in$ non-uniform P) if there is a function $f\colon \mathbb{N} \to \{0,1\}^*$ and a set $B \in \mathrm{P}$ such that

- $|f(n)| \le p(n)$ for a polynomial p and
- $A = \{x \in \Sigma^* \;:\; (x, f(|x|)) \in B\}$.

Such a function f is called an *advice* function since it provides additional information $f(n)$ on every input length n that allows to decide A in polynomial time. Note that f itself does not need to be computable. The class non-uniform P is sometimes also denoted by

P/poly, "P with polynomial advice". This additional information $f(n)$ can be understood as the encoding of a polynomial circuit deciding A on input length n. Indeed, it is easy to see that

$$A \in \text{non-uniform P} \iff A \text{ is decided by a sequence of circuits of polynomial size.}$$

Corollary 6.19. BPP $\subseteq$ non-uniform P. Therefore, all problems in BPP are of polynomial circuit complexity.

Theorem 6.20. BPP $\subseteq \Sigma_2^p \cap \Pi_2^p$.

Proof. It is sufficient to show that BPP $\subseteq \Sigma_2^p$. Since coBPP = BPP, it follows directly that BPP $\subseteq \Pi_2^p$.

Consider $A \in$ BPP. By Theorem 6.17, there is a polynomial PTM M deciding A with error probability $< 2^{-n}$:

$$x \in A \implies \Pr_{y\in\{0,1\}^{p(n)}}[M \text{ accepts } x] > 1 - 2^{-n}\text{, and}$$
$$x \notin A \implies \Pr_{y\in\{0,1\}^{p(n)}}[M \text{ accepts } x] < 2^{-n}.$$

In particular,

$$x \in A \iff |\{y \in \{0,1\}^{p(n)} \;:\; (x,y) \in L(M)\}| > 2^{p(n)}(1 - 2^{-n}).$$

Fix some x, $|x| = n$. Let $\Omega = \{0,1\}^{p(n)}$ and $B \subseteq \Omega$. We seek a criterion for the property $|B| > (1 - 2^{-n})|\Omega|$. The idea is to cover all of Ω with "few" images of B under translation modulo 2.

For $y, z \in \Omega$, let $y \oplus z := w_0 \dots w_{p(n)-1} \in \Omega$ with $w_i = y_i \oplus z_i$ (bitwise addition modulo 2). Let $B \oplus z := \{y \oplus z \;:\; y \in B\}$.

Lemma 6.21. For sufficiently large n and $B \in \{0,1\}^{p(n)}$ such that either

(i) $|B| < 2^{-n} \cdot 2^{p(n)}$ or
(ii) $|B| > (1 - 2^{-n}) \cdot 2^{p(n)}$

the following holds:

$$\text{(ii)} \iff \exists \overline{z} = (z_1, \dots, z_{p(n)}) \in \Omega^{p(n)} : \bigcup_i B \oplus z_i = \{0,1\}^{p(n)}.$$

Proof. ($\Rightarrow$): $\bigcup_i B \oplus z_i$ contains at most $p(n) \cdot |B|$ elements. If $\bigcup_i B \oplus z_i$ covers all of Ω, (i) is impossible since $p(n) \cdot 2^{-n}|\Omega| < |\Omega|$ for large n.

($\Leftarrow$): We use a probabilistic argument. Fix some $y \in \Omega$ and choose $z \in B \oplus y$. If we assume that (ii) holds, it follows that

$$\Pr_{z\in\Omega}[y \in B \oplus z] = \Pr_{z\in\Omega}[z \in B \oplus y] = \Pr_{z\in\Omega}[z \in B] > 1 - 2^{-n}.$$

Hence, we obtain:

$$\Pr_{\overline{z}\in\Omega^n}\left[\bigwedge_i y \notin B \oplus z_i\right] \leq \prod_i \Pr_{z_i\in\Omega}[y \notin B \oplus z_i] \leq 2^{-n\cdot p(n)}.$$

Therefore, the probability that some random $\overline{z} \in \Omega^n$ does *not* fulfil the conditions of the lemma can be approximated as follows:

$$\begin{aligned}\Pr_{\overline{z}\in\Omega^n}\left[\bigcup_i B \oplus z_i \neq \Omega\right] &\leq \sum_{y\in\Omega} \Pr_{\overline{z}\in\Omega^n}\left[\bigwedge_i y \notin B \oplus z_i\right] \\ &\leq 2^{p(n)} \cdot 2^{-n\cdot p(n)} < 1 \quad \text{for large } n.\end{aligned}$$

Hence, there must be a "good" $\overline{z}$. Q.E.D.

We can thus express A as follows: Let $B_x = \{y \in \Omega \ : \ (x,y) \in L(M)\}$. Then,

$$\begin{aligned} x \in A &\implies |B_x| > (1-2^{-n}) \cdot 2^{p(n)}, \text{ and} \\ x \notin A &\implies |B_x| < 2^{-n} \cdot 2^{p(n)}. \end{aligned}$$

Hence,

$$\begin{aligned} x \in A &\iff \exists \overline{z} \in \Omega^{p(n)} : \bigcup_{i=1}^{p(n)} B_x \oplus z_i = \Omega \\ &\iff \exists \overline{z} \in \Omega^{p(n)} \forall y \in \Omega \bigvee_{i=1}^{p(n)} \underbrace{y \in B_x \oplus z_i}_{\equiv y \oplus z_i \in B_x} \\ &\iff \exists \overline{z} \in \Omega^{p(n)} \forall y \in \Omega \bigvee_{i=1}^{p(n)} \underbrace{(x, y \oplus z_i) \in L(M)}_{\text{in P}}. \end{aligned}$$

Therefore, $A \in \Sigma_2^p$. Q.E.D.

6.3 Probabilistic proof systems and Arthur-Merlin games

We go back to the year 528 and turn our attention to the Court of King Arthur. A Round Table for 150 knights needs to be prepared. King Arthur is worried about peace at table as many knights are enemies. A seating arrangement needs to be found that makes sure no knights that are enemies are seated next to each other. Hence, King Arthur has the following problem: Given a graph $G = (P, E)$ with $P = \{\text{Arthur}\} \cup \{K_1 \ldots K_{150}\}$, and $E = \{(x,y) \ : \ x \text{ not enemy with } y\}$; find a Hamilton cycle of G.

Arthur is a wise man and assumes that the design of such a seating arrangement might lead to evaluate all 150! possibilities for which the remaining time until the Round Table would not be sufficient. That is why he charges his magician Merlin with this task. Merlin possesses some super-natural power and can therefore find a peaceful arrangement if it does exist.

As most reasonable people, King Arthur does not completely rely on magic. He therefore always double-checks all solutions that Merlin proposes before actually implementing them. That is, once Merlin proposes a seating arrangement $k_0, k_1, \ldots, k_{150}$ (let k_0 be the king), Arthur himself makes sure that for all j, $(k_j, k_{j+1}) \in E$.

However, the day comes when a new Round Table is going to take place. Some knights have reconciled, others have become enemies. Merlin finds out that there is no peaceful arrangement for this situation any more. King Arthur does not want to accept this result without proof, but a verification of all 150! possibilities is impossible.

Hence, Merlin needs to find a proof for the nonexistence of a seating arrangement that can be verified by Arthur. Since he cannot come up with one (as he does not know whether HAM $\in$ coNP), Merlin ends up in prison. After a while, the king regrets his impatience and is willing to accept a proof that he can verify with a probability of $1/2^{1000}$.

6.3.1 *Interactive proof systems*

The notion of a *proof* can—informally speaking—be defined as an interaction between a prover (P) and a verifier (V). After the interaction is completed, the verifier decides whether to accept the proof. Hence, a *proof system* is a protocol defining the interaction of P and V on input x (the theorem to prove). As opposed to proof notion from classical logic, this approach allows interesting observations on complexity.

The class NP is characterised by the following *deterministic proof system*: A language $Q \subseteq \Sigma^*$ is in NP if there are Turing computable functions $P : \Sigma^* \to \Sigma^*$ and $V : \Sigma^* \times \Sigma^* \to \{\text{accept, reject}\}$ with

- V polynomial in the first argument (i.e., $\text{time}_V(x,y) \leq p(|x|)$ for some polynomial p).
- *Completeness:* for all $x \in \Sigma^*$, $x \in Q \Longrightarrow V(x, P(x)) = \text{accept}$.
- *Correctness:* for all $P' : \Sigma^* \to \Sigma^*$ and all x, we have $x \notin Q \Longrightarrow V(x, P'(x)) = \text{reject}$ (i.e., no prover P' can convince V of an *incorrect* proposition "$x \in Q$.")

Example 6.22. The graph isomorphism problem GRAPHISO $= \{(G, H) : G, H \text{ graphs}, G \cong H\} \in \text{NP}$. Without loss of generality, we can assume that $G = (V, E^G)$ and $H = (V, E^H)$. Then, on an input (G, H), there is the following proof system:

- P returns some permutation $\pi : V \to V$.
- V verifies whether $\pi : G \xrightarrow{\sim} H$.

It is unknown whether GRAPHISO $\in$ coNP, that is, whether there is an NP proof system for GRAPHNONISO $= \{(G, H) : G \ncong H\}$.

In the following, we introduce *interactive proof systems* that allow a more sophisticated interaction between prover and verifier: the statements made by the prover are verified probabilistically in polynomial time; the verifier accepts with an error probability $\varepsilon > 0$.

Definition 6.23. Let Σ be an alphabet. An *interactive protocol* on Σ^* is a pair (P, V) of computable functions $P : \Sigma^* \to \Sigma^*$, $V : \Sigma^* \times \Sigma^* \times \{0,1\}^\omega \to \Sigma^* \cup \{\text{accept}, \text{reject}\}$ where V is polynomial in the first argument (i.e., $\text{time}_V(x, y, z) \leq p(|x|)$ for some polynomial p).

The *history* of (P, V) on $x \in \Sigma^*$ with the random word $y \in \{0,1\}^\omega$

is a sequence $U(x, y) = u_0, u_1, \ldots$ with $u_0 = x$ and $u_{i+1} = u_i a_i$ where

$$a_i := \begin{cases} V(x, u_i, y) & \text{if } i \text{ is even,} \\ P(u_i) & \text{if } i \text{ is odd and } a_i \notin \{\text{accept, reject}\}. \end{cases}$$

We say (P, V) accepts x (with the random word y) if $u_k = \text{accept}$ for some k. For every $x \in \Sigma^*$, the history of (P, V) is a random variable $U(x) : y \mapsto U(x, y)$.

Definition 6.24. Consider $Q \subseteq \Sigma^*$. An *interactive proof system* for Q is an interactive protocol (P, V) satisfying the following requirements:

- *Completeness:* for all $x \in \Sigma^*$,
  $x \in Q \Longrightarrow \Pr[(P, V) \text{ accepts } x] > \frac{2}{3}$.
- *Correctness:* for all $P' : \Sigma^* \to \Sigma^*$ and all x,
  $x \notin Q \Longrightarrow \Pr[(P', V) \text{ accepts } x] < \frac{1}{3}$.

A *round* of $U(x, y)$ is a pair (a_{2i}, a_{2i+1}) (hence, a "message" of P followed by an "answer" of V). We say a protocol has $\leq q$ rounds if, for all $x \in \Sigma^*$ and all $y \in \{0,1\}^\omega$, the history $U(x, y)$ has at most $q(|x|)$ rounds.

Definition 6.25. IP (respectively, IP[q]) is the class of all $Q \subseteq \Sigma^*$ for which there is an interactive proof system (respectively, one with $\leq q$ rounds).

Example 6.26. GRAPHNONISO $\in$ IP[2]. The proof system (P, V) works on input (G_0, G_1) with $G_i = (V_i, E_i)$ as follows:

- V chooses some index $i, i' \in \{0, 1\}$ at random and some permutation $\pi, \pi' : V \to V$. Then, he computes $H = \pi G_i$, $H' = \pi' G_i$ and sends both H and H' to the prover. (The chosen indices i, i', and permutations π, π' remain secret!)
- P replies with $j, j' \in \{0, 1\}$ such that $H \cong G_j$, and $H' \cong G'_j$.
- V accepts if $i = j$ and $i' = j'$; otherwise V rejects.

Analysis.

- If $G_0 \ncong G_1$, then H and H' are isomorphic to exactly one input graph. Thus, P can determine i and i'. It follows $G_0 \ncong G_1 \Longrightarrow \Pr[(P, V) \text{ accepts } (G_0, G_1)] = 1$.

- If $G_0 \cong G_1$ then H and H' are isomorphic to both graphs and all $G \cong G_0, G_1$ appear with equal probability. Every prover P' can do no better than guessing j, j'. Hence, for all P': $G_0 \cong G_1 \Longrightarrow \Pr[(P,V) \text{ accepts } (G_0,G_1)] \leq \frac{1}{4}$.

Lemma 6.27. $\text{NP} \subseteq \text{IP}[1] \subseteq \text{IP}[2] \subseteq \cdots \subseteq \text{IP} \subseteq \text{PSPACE}$.

Proof (IP $\subseteq$ PSPACE)*.*
Let (P,V) be an interactive proof system for Q. Since V is polynomial in the first argument, V on input (x, u_i, y) reads only the first $p(|x|)$ symbols of u_i and y. It is possible, using polynomial space, to simulate all possible histories $U(x,y)$ and to determine $\Pr[(P,V) \text{ accepts } x]$. Q.E.D.

It is interesting to know where—between NP and PSPACE—the class of interactively provable languages is located. For this, we return to Arthur and Merlin.

Definition 6.28. An *Arthur-Merlin game* (for a language Q) is an interactive proof system for Q with the additional requirement that the prover (Merlin) can see the random bits used by the verifier (Arthur). Without loss of generality, the messages from Arthur to Merlin are then composed of a sequence $y_1, \ldots, y_r$ $(r \leq p(|x|))$ of random bits.

AM (respectively, AM$[q]$) is the class of all $Q \subseteq \Sigma^*$ that have an Arthur-Merlin game (respectively, one with $\leq q(|x|)$ rounds).

Obviously, $\text{AM} \subseteq \text{IP}$, and

$$\text{AM}[q] \subseteq \text{IP}[q].$$

In the following, we see that one of the most difficult problems from PSPACE is in AM.

6.3.2 An Arthur-Merlin game for the permanent of a matrix

Definition 6.29. Let R be a ring, $R' \subset R$ some subset and $A \in M_n(R')$ an $n \times n$ matrix over R'. The *permanent* of A (over R) is defined as

$$\text{per}_R A = \sum_{\sigma \in S_n} a_{1\sigma(1)} \cdots a_{n\sigma(n)}.$$

The analogy to determinants is striking:

$$\det A = \sum_{\sigma \in S_n} \text{sgn}(\sigma) a_{1\sigma(1)} \cdots a_{n\sigma(n)}.$$

By removing the ith line and the jth column, one obtains the ij minor A_{ij} of A, and it follows

$$\det A = \sum_{i=1}^{n} (-1)^{i+1} a_{i1} \cdot A_{i1} \quad \text{and}$$

$$\text{per} A = \sum_{i=1}^{n} a_{i1} \cdot A_{i1}.$$

However, the determinant can be computed efficiently where as no efficient algorithm is known for the permanent. It is currently assumed that in general it is not efficiently computable. The following result confirms this assumption.

Definition 6.30. #P is the class of all functions $f : \Sigma^* \to \mathbb{N}$ for which there is a polynomial nondeterministic TM M such that the number of accepting computations of M on x is exactly $f(x)$.

Theorem 6.31 (Toda)**.** PH $\subseteq$ #P.

Theorem 6.32 (Valiant)**.** The permanent (over $\mathbb{Z}$) of 0-1-matrices is #P -complete.

A polynomial algorithm to compute the permanent would therefore imply PH = P. An interactive proof system for

$$\text{PER} = \{(A,q) \ : \ A \in M_n(\{0,1\}), \text{per}_{\mathbb{Z}} A = q\}$$

hence implies that every problem $Q \in \text{PH}$ has an interactive proof system. This would in particular be true for

$$\overline{\text{HAM}} = \{G \ : \ G \text{ contains no Hamilton circle }\} \in \text{coNP} \subseteq \text{PH}.$$

Theorem 6.33. There is an Arthur-Merlin game for PER.

Proof. (1) Since $A \in M_n(\{0,1\}) \Longrightarrow 0 \leq \text{per}_{\mathbb{Z}}(A) \leq n!$. Let $p > n!$ be a prime number. Then, $\text{per}_{\mathbb{F}_p}(A) = \text{per}_{\mathbb{Z}}(A)$ over the field $\mathbb{F}_p$.

(2) For a field $\mathbb{F}$, let $\mathbb{F}[X]_d = \{f \in \mathbb{F} : \text{degree} f \leq d\}$. If $A \in M_n(\mathbb{F}[X]_d)$ then per$A \in \mathbb{F}[X]_{nd}$.

(3) *The protocol:*
Arthur and Merlin work with a list $L = \{(A_1, q_1) \dots (A_r, q_r)\}$ where $A_i \in M_k(\mathbb{F}_p)$ and $q_i \in \mathbb{F}_p$ $(k \leq n)$. L is correct if per$A_i = q_i$ for $i = 1, \dots, r$.
Beginning: $L = \{(A, q)\}$, $A \in M_n(\mathbb{F}_p)\}$.
In a sequence of subprotocols 'expand' and 'reduce', L is changed until, at the
End: $L = \{(B, s)\}$, $B \in M_2(\mathbb{F}_p)$.
Arthur accepts if, and only if, per $B = s$.

- if L contains only one pair: $L = \{(A, q)\}$, $A \in M_k(\mathbb{F}_p)\}$, $k > 2$:

  Expansion step:
  $L \mapsto L' = \{(A_1, q_1), \dots, (A_k, q_k)\}$ where $A_i \in M_{k-1}(\mathbb{F})_p$.
  Subprotocol **expand**(L)
  Input: $L = \{(A, q)\}$
  Merlin: computes $q_i = \text{per}(A_{i1})$ for $i = 1, \dots, k$ and sends the results $q_1, \dots, q_k$ to Arthur.
  Arthur: verifies whether $\sum_{i=1}^k a_{i1} q_i = q$.
  If not, he rejects;
  otherwise, he sets $L' = \{(A_{i1}, q_1), \dots, (A_{k1}, q_k)\}$.

  For this step, we have

  $$L \text{ correct} \Longrightarrow L' \text{ correct.}$$
  $$L \text{ incorrect} \Longrightarrow L' \text{ incorrect (no matter how Merlin plays).}$$

- if $|L| > 1$,
  Reduction step: $L \mapsto L'$ with $|L'| = |L| - 1$ such that

  $$L \text{ correct} \Longrightarrow L' \text{ correct.}$$
  $$L \text{ incorrect} \Longrightarrow L' \text{ incorrect with high probability.}$$

  Consider $(A, q_1), (B, q_2) \in L$, $A, B \in M_k(\mathbb{F}_p)$. Set

  $$C(X) = (1 - X)A + XB$$

$$= \begin{pmatrix} & \vdots & \\ \dots & \alpha x + \beta & \dots \\ & \vdots & \end{pmatrix} \in M_k(\mathbb{F}_p[X]).$$

With per $C(X) =: f \in \mathbb{F}_p[X]$, we have

$$C(0) = A, \quad \text{hence} \quad f(0) = \text{per } A;$$
$$C(1) = B, \quad \text{hence} \quad f(1) = \text{per } A.$$

Remark: To verify whether per $A = q_1$ and per $B = q_2$, it therefore suffices to determine the polynomial f and to evaluate it at 0 and 1.

Subprotocol **reduce**(L):
Input: $\{(A, q_1), (B, q_2)\}$, $A, B \in M_k \mathbb{F}_p$.
Merlin: sends Arthur $c_0, \dots, c_k \in \mathbb{F}_p$
(claiming that $f(X) = c_0 + c_1 X + \dots + c_k X^k$).
Arthur: sets $g(X) = c_0 + c_1 X + \dots + c_k X^k$, and
verifies whether $g(0) = q_1$ and $g(1) = q_2$.
If not, he rejects;
otherwise, he chooses a random number $a \in \mathbb{F}_p$ and sets $L' = (L - \{(A, q_1), (B, q_1)\} \cup \{(C(a), g(a))\}$.

For this step, we have

L correct $\Longrightarrow$ Merlin can play in such a way that L' is correct by sending the correct coefficients of f, i.e., such that $g(X) = f(X) = \text{per } C(X)$, and in particular $g(a) = \text{per } C(a)$.
L incorrect $\Longrightarrow$ with high probability L' incorrect.

Reason: Assume that per $A = \text{per } C(0) \neq q_1$. Merlin sends incorrect coefficients since $g(0) = q_1 \neq f(0) = \text{per } A$. Hence, we have

$$\begin{aligned} f \neq g &\Longrightarrow |\{a : f(a) = g(a)\}| \leq k \\ &\Longrightarrow \Pr[\text{per } C(a) = g(a)] \leq \frac{k}{p} < \frac{1}{(n-1)!} \end{aligned}$$

for $p > n!$, and $k \leq n$.

Hence, we obtain the Arthur-Merlin game described in Algorithm 6.2.
Analysis.

(a) The game is indeed an Arthur-Merlin protocol, i.e., all computations by Arthur are in P since
  - $p < 2n!$, $|p| = O(n \log n)$ since $n! = 2^{O(n \log n)}$,
  - the arithmetical operations in $\mathbb{F}_p$ are polynomial in $|p|$ and
  - the protocol uses $n-2$ expansion steps and $\sum_{i=2}^{n-2}(i-1) = \frac{(n-1)(n-2)}{2}$ reduction steps.

(b) $(A, q) \in$ PER $\Longrightarrow \Pr[(M, A) \text{ accepts } (A, q)] = 1$.

Algorithm 6.2. Arthur-Merlin game for PER

```
Input: $(A,q)$, $A \in M_n(\{0,1\}), q \in \mathbb{N}$
Merlin:
    sends Arthur a prime number $p \in [n!, 2n!]$ together with a short
    proof showing that $p$ is prime[a].
Arthur:
    verifies that $p$ is indeed a prime number between $n!$ and $2n!$.
    if $p$ not prime then reject
/* For the remainder of the protocol, all calculations are
    done in $\mathbb{F}_p$.                                          */
$L := \{(A,q)\}$
while $L \neq \{(B,s)\}$ for a $B \in M_2(\mathbb{F}_p)$ do
    if $|L| = 1$ then expand(L) else reduce(L)
endwhile
Arthur:
    verifies whether per $B = s$.
    if yes then accept else reject
```

[a] It is known that for all $a \in \mathbb{N}$ there is a prime number between a and $2a$ (Bertrand's postulat). Since PRIMES $\in$ NP, there are short proofs for the fact that p is prime (i.e., there is an $L \in$ P with PRIMES $= \{p : \exists w |w| \leq |p|^k : (p, w) \in L\}$ (where w is a short proof).

(c) Let $(A, q) \notin$ PER and M' some prover. Then, (M', A) accepts $(A, q) \Longrightarrow M'$ has cheated successfully in at least one reduction step. This can occur in one single reduction step with a probability of at most $\frac{1}{(n-1)!}$. Altogether, we obtain

$$\begin{aligned} \Pr[(M'A) \text{ accepts } (A,q)] &< 1 - \left(1 - \frac{1}{(n-1)!}\right)^{\frac{(n-1)(n-2)}{2}} \\ &< \frac{(n-1)(n-2)}{(n-1)!} = \frac{1}{(n-3)!}. \end{aligned}$$ Q.E.D.

In 1992, this theorem was published by Lund, Fortnow, Karloff and Nisan in JACM 39(4).

6.3.3 IP = PSPACE

Only one month after Lund, Fortnow, Karloff and Nisan, Shamir showed that IP = PSPACE. In order to do so, he constructed an interactive proof system (even an Arthur-Merlin game) for QBF. QBF is the problem to evaluate quantified Boolean formulae, and it is PSPACE-complete. An Arthur-Merlin game for this problem therefore suffices to show that all $Q \in$ PSPACE have an interactive proof system (and even an Arthur-Merlin game). At the same time it shows that Arthur-Merlin games are equally strong as interactive proof systems.

Theorem 6.34 (Shamir)**.** There is an Arthur-Merlin game for QBF.

Proof (Simplified version).

(1) Arithmetisation of a formula from quantified propositional logic. First, let $\varphi(X_1, \ldots, X_n)$ be a propositional formula (without quantifiers) and let $\mathbb{F}$ be some arbitrary field. A map $\varphi \mapsto F_\varphi \in \mathbb{F}[X_1, \ldots, X_n]$ is defined inductively as follows:

$$\begin{aligned} F_{X_i} &= X_i, \\ F_{\alpha \wedge \beta} &= F_\alpha \cdot F_\beta, \\ F_{\neg \alpha} &= 1 - F_\alpha, \\ F_{\alpha \vee \beta} &= F_\alpha \circ F_\beta := F_\alpha + F_\beta - F_\alpha F_\beta. \end{aligned}$$

Let $\mathfrak{I} : \{X_1, \dots, X_n\} \mapsto \{0,1\}$ be an interpretation with $\mathfrak{I}(X_1) = \varepsilon_1, \dots, \varepsilon(X_n) = \varepsilon_n$. We write $\varphi(\varepsilon_1, \dots, \varepsilon_n)$ for $\mathfrak{I}(\varphi)$. For every field $\mathbb{F}$ and all $\varepsilon_1, \dots, \varepsilon_n \in \{0,1\}$, $F_\varphi(\varepsilon_1, \dots, \varepsilon_n) = \varphi(\varepsilon_1, \dots, \varepsilon_n)$.
For $f \in \mathbb{F}[Y, \overline{X}]$, we define the following:

$$\begin{aligned}
(\forall Y f)(\overline{X}) &= f(0,\overline{X}) \cdot f(1,\overline{X}) && \in \mathbb{F}[\overline{X}],\\
(\exists Y f)(\overline{X}) &= f(0,\overline{X}) \circ f(1,\overline{X}) && \in \mathbb{F}[\overline{X}],\\
(RY f)(\overline{X}) &= f \pmod{Y^2 - Y} && \in \mathbb{F}[Y,\overline{X}].
\end{aligned}$$

(all Y^i with $i > 0$ are replaced by Y)

Hence, we obtain the following arithmetisation of quantified propositional formulae with quantifiers:

$$\begin{aligned}
F_{\forall Y \varphi} &= (\forall Y F_\varphi),\\
F_{\exists Y \varphi} &= (\exists Y F_\varphi).
\end{aligned}$$

Obviously, for all quantified propositional formulae $\Psi(X_1, \dots, X_k)$, we have $F_\Psi(\varepsilon_1, \dots, \varepsilon_k) = \Psi(\varepsilon_1, \dots, \varepsilon_k)$. In particular if free$(\Psi) = \emptyset$, $\Psi \in$ QBF $\iff F_\Psi = 1$ holds.
However, we have the following problem: The explicit construction of F_Ψ is just as difficult as the evaluation of the QBF formula Ψ. Length and degree of the polynomial F_Ψ can become arbitrarily large since every application of the quantifiers $\forall$ and $\exists$ double both the length and degree.

(2) Degree reduction using R.
For $u \in \{0,1\}$, we have $(RYf)(u,\overline{X}) = f(u,\overline{X})$. Further, for $f \in \mathbb{F}[X_1, \dots, X_n]$ set

$$(R^* f)(X_1, \dots X_n) = (RX_1 RX_2 \dots RX_n f)(X_1, \dots, X_n).$$

For $\varepsilon_1, \dots \varepsilon_n \in \{0,1\}$,

- $(R^* f)(\varepsilon_1, \dots \varepsilon_n) = f(\varepsilon_1, \dots \varepsilon_n)$, and
- degree$(R^* f) \leq n$.

Let $\Psi = Q_1 X_1 \dots Q_n X_n \varphi(X_1, \dots, X_n)$ be a quantified Boolean formula with free$(\Psi) = \emptyset$. Then, $\Psi \in$ QBF if, and only if,

$(Q_1 X_1 R^* Q_2 X_2 \dots R^* Q_n X_n R^* F_\varphi) = 1$ since the R^* operator leaves the functional values invariant for the arguments 0,1.

(3) Arthur-Merlin game.
Let $f \in \mathbb{F}[X_1, \dots, X_n]$, $\mathbb{F}$ be finite. We assume there is an Arthur-Merlin game with $(M,A)_f$ for $\{(u,v) \in \mathbb{F}^{n+1} : f(\overline{u}) = v\}$ with

(i) $f(\overline{u} = v) \Longrightarrow \Pr[M$ accepts $(\overline{u}, v)] = 1$,
(ii) $f(\overline{u} \neq v) \Longrightarrow \Pr[M'$ accepts $(\overline{u}, v)] < \varepsilon$ for all M'.

Let $g \in \{(\exists X_i f), (\forall X_i f), (RX_i f) : i = 1, \dots, n\}$. We assume Arthur knows a d with degree $g \leq d$.
Based on these assumptions, we construct an Arthur-Merlin game $(M,A)_g$ that calls $(M,A)_f$ exactly once as subprotocol with

(i) $f(\overline{u} = v) \Longrightarrow \Pr[M$ accepts $(\overline{u}, v)] = 1$,
(ii) $f(\overline{u} \neq v) \Longrightarrow \Pr[M'$ accepts $(\overline{u}, v)] < \varepsilon + \frac{d}{|\mathbb{F}|}$ for all M'.

We obtain the following different cases:

(a) $g(\overline{X}) = (\forall Y f)(Y, \overline{X})$.
Merlin wants to show that $g(\overline{u}) = v$. He sends the coefficients of a polynomial $s(Y)$ (requiring that $s(Y) = f(Y, \overline{u})$). If degree $s > d$ or $s(0) \cdot s(1) \neq v$, Arthur rejects. Otherwise, Arthur chooses some random $w \in \mathbb{F}$. Merlin now needs to convince Arthur with the protocol $(M,A)_f$ that $f(w, \overline{u}) = s(w)$.
(b) $g(\overline{X}) = (\exists Y f)(Y, \overline{X})$.
Analogously, with $s(0) \circ s(1)$ instead of $s(0) \cdot s(1)$.
(c) $g(Y, \overline{X}) = (RY f)(Y, \overline{X})$.

We use

Lemma 6.35. $(RYf)(Y,\overline{X}) = f(0,\overline{X}) + [f(1,\overline{X}) - f(0,\overline{X})] \cdot Y$.

Proof. Let $f = \sum_{i=0}^m Y^i \cdot g_i(\overline{X})$. Then,

$$f(0,\overline{X}) = g_0(\overline{X}) \quad \text{and} \quad f(1,\overline{X}) = \sum_{i=0}^m g_i(\overline{X}).$$

Hence, we obtain $(RYf) = g_0(\overline{X}) + \sum_{i=0}^m g_i(\overline{X}) = f(0,\overline{X}) + [f(1,\overline{X}) - f(0,\overline{X})] \cdot Y$. Q.E.D.

Merlin wants to convince Arthur that $g(z, \overline{u}) = v$. He sends Arthur the coefficients of a polynomial $s(Y)$ (requiring that $s(Y) = f(Y, \overline{u})$). If degree $s > d$ or $s(0) + z(s(1) - s(0)) \neq v$, Arthur rejects. Otherwise, he sends Merlin some random $w \in \mathbb{F}$. Merlin now needs to convince Arthur with the protocol $(M, A)_f$ that $f(w, \overline{u}) = s(w)$.

The game described above fulfils the completeness requirement. As for correctness, we note that M' has the following possibilities to cheat successfully:

- In $(M, A)_f$ (with probability ε).
- If $s(Y) \neq f(Y, \overline{u})$ correspond at a randomly selected point w (probability $\leq \frac{d}{\mathbb{F}}$).

(4) Summary.
Given $\Psi = Q_1 X_1 \dots Q_n X_n \varphi$. Merlin convinces Arthur that

$$G_\Psi = Q_1 X_1 R^* \dots R^* Q_n X_n R^* F_\varphi = 1$$

with the help of the above-mentioned reduction steps. At the end, an equation $F_\varphi(u_1, \dots, u_n) = v$ needs to be verified. For a propositional formula φ, this is possible in polynomial time.
The error probability of the complete protocol is at most

$$\frac{\#\exists, \forall, R\text{-operators} \cdot \text{ maximal degree}}{|\mathbb{F}|} = \frac{O(n^2) + O(|\varphi|^2}{|\mathbb{F}|}.$$

Therefore, it suffices to choose a field $\mathbb{F}_p$ with $p \geq c \cdot |\varphi|^4$. Q.E.D.