T-79.511 Special Course on Cryptology / Zero Knowledge: Rudiments of Interactive Proof Systems

Petteri Kaski

September 26, 2001

1 Introduction

A mathematical proof can be viewed as an interaction between two parties, the *prover* and the *verifier*. The task of the prover is to convince the verifier of the validity of some mathematical assertion known to both the prover and the verifier. For example, we can consider a professor explaining a theorem in front of an audience of students as the prover; the students act as the verifiers of the proof, i.e., the lecture.

A *proof system* is a formalization of the task of communicating a proof and establishing its validity. For example, the classical proof system is one where the prover prepares a written proof of the assertion and sends it to the verifier for review. The verifier then analyzes the proof and either becomes convinced of the validity of the assertion or decides that the proof is not convincing enough. Typically this decision is based on some semantic notion of validity. In particular, a proof system is *complete* if the prover can convince the verifier of the validity of every assertion that is valid in the semantic sense. Conversely, a proof system is *sound* if the verifier cannot be convinced of a semantically invalid assertion.

In a computational complexity setting proof systems are often used as tools for measuring how hard it is to recognize a particular language or a class of languages. Namely, the task of the prover in a proof system for a language L is to convince the verifier that $x \in L$ for a string x; the behaviours of the prover and the verifier on input x are encoded using Turing machines. By varying the resource bounds and capabilities of the Turing machines and the level of interaction allowed between the two machines it becomes possible to characterize different families of languages. For example, a fundamental characterization of **NP**, the class of languages decidable by polynomial time nondeterministic Turing machines, is that it consists of precisely those languages L for which every $x \in L$ is recognizable by a polynomial time deterministic verifier that has as additional input a polynomially bounded string (the *proof* or *witness* of x) computed by an exponential time prover [3, 10]. Thus, **NP** can be considered as the complexity-theoretic analogue of the classical proof system since no further interaction is required between the verifier and the prover apart from the prover providing the proof (e.g. the satisfying truth assignment to a satisfiable Boolean formula).

Interactive proof systems, introduced by Goldwasser, Micali, and Rackoff [7] and independently by Babai [1] (see also [2]), modify the classical setting in three fundamental ways.

1. The interaction between the prover and the verifier becomes genuine. Victor the verifier may query Peggy the prover interactively, and Peggy must provide answers to Victor's queries.

2. Victor has access to a random source, which he can use in formulating his queries to Peggy.

3. The notions of completeness and soundness become probabilistic. It suffices that Victor most of the time declares valid a truly valid assertion, and almost never declares an invalid assertion valid.

In this survey we give a gentle introduction to interactive proof systems as language recognizers, and illustrate the equivalence between interactive proof systems and Arthur–Merlin games [1, 2], which can be viewed as a special type of an interactive proof system.

2 Interactive proof systems

We assume that the reader is familiar with the basics of Turing machines and resource-bounded computation (see e.g. [11]).

First some preliminary definitions. We write Σ^* for the set of all finite strings over a finite alphabet Σ. The length of a string $x \in \Sigma^*$ is denoted by $|x|$. The concatenation of $y, z \in \Sigma^*$ delimited a special symbol $\# \in \Sigma$ is denoted by $y\#z$; the empty string is denoted by ε.

The framework for interactive proof systems presented in this survey is similiar to that in [8]. (The other standard model is one that uses Turing machines that interact by sharing several tapes; see [5, 6, 7]. For the purpose of language recognition, these two models should be identical in all respects.) Suppose that (V, P) is a pair of functions

$$V : \Sigma^* \times \Sigma^* \times \Sigma^* \to \Sigma^* \cup \{\text{accept}, \text{reject}\},$$
$$P : \Sigma^* \times \Sigma^* \to \Sigma^*,$$

and that $L \subseteq \Sigma^*$ is a language. Informally, V models the behaviour of Victor, and P models the behaviour of Peggy during a protocol that terminates in Victor deciding whether $x \in L$ holds for the common input $x \in \Sigma^*$. We encode the messages exchanged between Victor and Peggy so far in the protocol by as a string $s_i = \#y_1\#z_1\# \cdots \#y_i\#z_i \in \Sigma^*$; $s_0 = \epsilon$. The message sent by Victor to Peggy with message history s_i and random string $r \in \Sigma^*$ is $V(x, r, s_i)$. Conversely, the message sent by Peggy to Victor with message history $s_i\#y_{i+1}$ is $P(x, s_i\#y_{i+1})$. Note that the random string r is not known to Peggy.

Definition 2.1 For a $x \in \Sigma^*$ and a random string $r \in \Sigma^*$ we say that (V, P) *accepts* (respectively, *rejects*) x on r in k rounds if there exists a message stream $s_k = \#y_1\#z_1\# \cdots \#y_k\#z_k$ such that $V(x, r, s_k) = \text{accept}$ (respectively, $V(w, r, s_k) = \text{reject}$), and for each $i = 0, \ldots, k-1$, we have $V(x, r, s_i) = y_{i+1}$ and $P(x, s_i\#y_{i+1}) = z_{i+1}$. We call the strings y_i, z_i the *messages* exchanged on round i. The string x is the *input*.

Definition 2.2 We say that (V, P) is *well-formed* if there exist polynomially-bounded functions $l, g, m : \mathbb{N} \to \mathbb{N}$ independent of P such that every $x \in \Sigma^*$ is either accepted or rejected on every $r \in \Sigma^{l(|x|)}$ in at most $g(|x|)$ rounds using messages of length at most $m(|x|)$.

Suppose that (V, P) is well-formed and let $x \in \Sigma^*$. Denote by $\Pr((V, P) \text{ accepts } x)$ the probability that (V, P) accepts x if $r \in \Sigma^{l(|x|)}$ is selected uniformly at random.

Definition 2.3 Let $L \subseteq \Sigma^*$ be a language. A well-formed pair (V, P) is *an interactive proof system for L with error probability* ϵ if V is a polynomial time computable function and the following two conditions hold:

(i) *Completeness.* For every $x \in L$,

$$\Pr((V, P) \text{ accepts } x) \geq 1 - \epsilon.$$

(ii) *Soundness.* For every $P' : \Sigma^* \times \Sigma^* \to \Sigma^*$ such that (V, P') is well-formed and every $x \notin L$,

$$\Pr((V, P') \text{ accepts } x) \leq \epsilon.$$

Definition 2.4 Let **IP** denote the class of languages that have interactive proof systems with error probability $1/3$.

Henceforth if we speak of an interactive proof system without fixing a concrete error probability then the error probability is always assumed to be $1/3$.

Definition 2.5 We write $\mathbf{IP}[2t(n)]$ for the class of languages that have an interactive proof system for which the number of rounds required to decide each input of length n is bounded from above by $t(n)$.

3 Example interactive proof systems

The graph isomorphism problem belongs to the class of problems whose relationship to various other complexity classes has become better understood through interactive proof systems [6, 9].

We shall review the necessary graph-theoretic definitions (see e.g. [4] for further information) before presenting interactive proof systems for both graph isomorphism and nonisomorphism. A *graph* is a pair (V, E), where V is a finite set of *vertices* and E is a set of 2-subsets of V called *edges*. The *order* of a graph is the number of vertices it contains.

Two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ are *isomorphic* if there exists a permutation $\pi : V \to V$ such that

$$E_2 = \pi E_1 \stackrel{\text{def}}{=} \{\{\pi(u), \pi(v)\} : \{u, v\} \in E_1\}.$$

A permutation π as above is called an *isomorphism* of G_1 onto G_2.

Problem 3.1 (GRAPH ISOMORPHISM) Given two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, decide whether G_1 and G_2 are isomorphic.

Problem 3.2 (GRAPH NONISOMORPHISM) Given two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, decide whether G_1 and G_2 are *not* isomorphic.

Clearly both languages induced by these problems are decidable by a Turing machine that exhaustively searches every possible permutation of the vertices to locate an isomorphism. Both languages also have an interactive proof system. (Note that Definition 2.3 is asymmetric in the sense that the existence of an interactive proof system for a langugage does not immediately guarantee that the complementary language has an interactive proof system.) We next present interactive proof systems for GRAPH ISOMORPHISM and GRAPH NONISOMORPHISM.

Interactive protocol for GRAPH ISOMORPHISM.
Input: Graphs $G_1 = (V, E_1)$, $G_2 = (V, E_2)$.

1. Victor sends G_1 and G_2 to Peggy.
2. If G_1 and G_2 are isomorphic, then Peggy computes an isomorphism of G_1 onto G_2 and sends it to Victor; otherwise Peggy declares failure.
3. Victor accepts only if the mapping sent by Peggy is indeed an isomorphism and rejects otherwise.

The protocol between Victor and Peggy can clearly be formulated as an interactive proof system: the length and number of messages sent between Victor and Peggy are at most polynomial in the input length, and the computation of Victor is straightforward to perform in polynomial time. The error probability is 0 since no randomization is employed and it is impossible for Victor to accept if the graphs are nonisomorphic. Thus, GRAPH ISOMORPHISM is in **IP**[2].

Interactive protocol for GRAPH NONISOMORPHISM.
Input: Graphs $G_1 = (V, E_1)$, $G_2 = (V, E_2)$.

1. Victor generates randomly two graphs repeating the following procedure twice. First, Victor selects randomly either $i = 1$ or $i = 2$. Then, Victor generates a random vertex permutation $\pi : V \to V$ and constructs $G' = (V, \pi E_i)$.
2. Victor sends the two constructed graphs to Peggy.
3. For each of the graphs received Peggy determines whether the graph is isomorphic to G_1 or G_2 and sends this information to Victor.
4. If Peggy produced the correct answer for both of the constructed graphs, Victor accepts; otherwise Victor rejects.

Victor can obviously perform the required computation within a polynomial time bound. If G_1 and G_2 are nonisomorphic, then it is clear that Peggy can compute the correct answer every time by exhaustively searching through all possibilities. On the other hand, if G_1 and G_2 are isomorphic, then there is no way for Peggy to discover which of graphs G_1, G_2 was selected by Victor in phase 1. The probability that Peggy guesses correctly on both of the constructed graphs is $1/4$. Thus, the error probability is $1/4$ and GRAPH NONISOMORPHISM is in **IP**[2].

4 Simulation and amplification

This section is devoted to sketching several basic observations that are often required as parts of more complex constructions.

In an interactive proof system Victor is restricted to (probabilistic) polynomial time computation, whereas Peggy is assumed to be omnicapable. However, because of Victor's limitation it suffices for Peggy to devote only the computational resources equivalent to a polynomial-space bounded Turing machine for interacting with Victor. This is because Peggy can in polynomial space simulate the protocol she undergoes with Victor and determine the acceptance probability associated with each reply that she makes.

Lemma 4.1 *Let (V, P) be an interactive proof system for $L \subseteq \Sigma^*$. Then there exists an interactive proof system (V, P') for L, where P' is a polynomial space computable function.*

Proof sketch. It suffices to construct a polynomial space computable P' that satisfies the completeness condition. Because (V, P) is well-formed, both the message length and the number of rounds in the protocol are bounded from above by a polynomial in $|x|$. Furthermore, so is the length of the random string $r \in \Sigma^{l(|x|)}$ that is unknown to the prover. Fix an input $x \in \Sigma^*$. Suppose as an induction hypothesis that it is possible to compute in polynomial space the reply z ($|z| \leq m(|x|)$) that causes V to accept $x \in \Sigma^*$ with the maximum probability if at most $i \geq 1$ messages must still be sent by the prover to V. (The associated acceptance probability is also assumed to be produced by the algorithm.)

The base case $i = 1$ is clear: In polynomial space it is possible to simulate the behaviour of V for every random tape content $r \in \Sigma^{l(|x|)}$ after V has received a candidate reply z. If the simulation keeps track of the ratio of accepting random tape contents to the number of random tape contents that are still possible given the current message history (the acceptance probability), then it is straightforward to determine the desired reply z together with its acceptance probability.

The inductive step is shown to hold by using the level i simulation recursively as follows. The behaviour of V is again simulated for every reply z and every random tape content r. During the simulation any message y (other than accept or reject) produced by V is fed to the level i simulation together with z and the current message history. The resulting acceptance probability is incorporated into the overall acceptance probability for z computed over all r.

Because the simulation considers all replies P would make, it must satisfy the completeness condition. ∎

Based on the previous proof it is immediate that for every language in **IP** there exists a Turing machine that can in polynomial space simulate an interactive proof system and decide according to computed verifier acceptance probability. Consequently:

Corollary 4.2 **IP** $\subseteq$ **PSPACE**.

The error probability of an interactive proof system can be made exponentially small with respect to the input size by performing several of the original protocol runs in parallel and deciding according to the outcome of the majority of the runs.

Lemma 4.3 (Amplification Lemma) *Let $L \subseteq \Sigma^*$ be a language and suppose that (V, P) is an interactive proof system for L with error probability $\epsilon < 1/2$. If $q(n)$ is a polynomially bounded, polynomial time constructible function, then L has an interactive proof system (V', P') that (i) has error probability $\epsilon \leq 2^{-q(n)}$ for inputs of length n; and (ii) uses the same number of rounds to decide every input as (V, P).*

Proof sketch. Let α denote the constant that corresponds to $p = \epsilon$ in the Majority Vote Lemma (see Appendix). Modify (V, P) to run, for every input $x \in \Sigma^*$, $m = \lceil \frac{q(|x|) \ln 2}{\alpha} \rceil$ runs of the protocol in parallel. Let the majority outcome of the parallel protocol runs determine the decision of the verifier. ∎

5 Arthur–Merlin games

Arthur–Merlin games introduced by Babai [1] can be considered as a special type of interactive proof system in which the verifier (King Arthur) is restricted to send only random strings to the prover (Merlin the Wizard).

Definition 5.1 An interactive proof system (A, M) for $L \subseteq \Sigma^*$ is an *Arthur–Merlin proof system* if the concatenated messages sent by Arthur to Merlin in every possible run of the protocol constitute a prefix of the random string $r \in \Sigma^*$ supplied to Arthur.

Definition 5.2 We write $\mathbf{AM}(2t(n))$ for the class of languages decidable by an Arthur–Merlin proof system for which the number of rounds required to decide each input of length n is bounded from above by $t(n)$.

Theorem 5.3 (Collapse Theorem) *Let t be a polynomially bounded function and suppose $t(n) \geq 1$ for all $n \geq 0$. Then,*

$$\mathbf{AM}(2t(n)) = \mathbf{AM}(2t(n)+2).$$

Proof sketch. (This proof is a restricted version of the proof of the speedup theorem in [2].) Let $L \in \mathbf{AM}(2t(n)+2)$. Suppose (A, M) is an Arthur–Merlin protocol for L with error probability $\epsilon \leq 1/3$. For every input $x \in \Sigma^*$ the protocol can be assumed to have at least two rounds: On the first round Arthur sends y to Merlin, and Merlin replies with z. On the second round Arthur sends y', and Merlin replies with z'.

To reduce the number of rounds we shall merge the first two rounds into one so that Arthur initially sends both y and y' to Merlin, and Merlin replies with z and z'. The rest of the protocol is as before. Clearly, Merlin might gain a decisive advantage in the new protocol because he knows one of Arthur's moves beforehand. Merlin's advantage can be removed by parallelization as follows. Instead of sending y and y' to Merlin on the first round, Arthur splits the protocol into q parallel runs by sending to Merlin y and $y'_1, \ldots, y'_q$. Merlin answers by sending z and $z'_1, \ldots, z'_q$ to Arthur. Note that Merlin is forced to play the same move z against Arthur's subsequent choices $y'_1, \ldots, y'_q$. After the initial round of the new protocol (which comprises the two initial rounds of the original protocol), the new protocol continues simulating the q runs of the original protocol in parallel with independent moves allowed in each protocol run. In the end Arthur decides according to the majority of the protocol outcomes.

The new protocol clearly satisfies the completeness condition because Merlin may ignore on the first round that he knows Arthur's subsequent parallel moves and play as in the original protocol. We next argue that Merlin's advantage is neglible if q is sufficiently large.[1] Suppose $x \notin L$. We know that Arthur accepts with probability at most ϵ in the original protocol in this case. Then, for at most $1/t$ of the possible initial choices y, Arthur will accept with probability exceeding $t\epsilon$ in the subsequent protocol. (Otherwise Arthur's acceptance probability would exceed ϵ.) So, for at least a fraction $(t-1)/t$ of the possible initial choices y in the original protocol, Arthur's acceptance probability given y does not exceed $t\epsilon$. Fix such a choice y and consider the subsequent protocol. Let z be any choice made by Merlin. Then, for at least a fraction $(s-1)/s$ of the choices y' possible for Arthur, Arthur's acceptance probability cannot exceed $st\epsilon$. So, the overall acceptance probability on a random y' given y, z cannot exceed $\delta \stackrel{\text{def}}{=} (s-1)/s \cdot st\epsilon + 1/s \cdot 1 = (s-1)t\epsilon + 1/s$. Moreover, if $\delta < 1/2$, then the probability that more than half of randomly chosen $y'_1, \ldots, y'_q$ accept for given y, z is bounded by $2^{-\alpha_\delta q}$, where α_δ is the constant in the majority vote lemma. The acceptance probability of Arthur in the new protocol is thus at most $1/t \cdot 1 + (t-1)/t \cdot \sum_z 2^{-\alpha_\delta q}$.

Since ϵ can be fixed to an arbitrarily small constant value by parallelization in the original protocol, we may fix s, t so that $1/s, 1/t < 1/6$ and $st\epsilon < 1/6$. Consequently, $\delta < 1/2$ and the majority vote lemma can be applied to bound the acceptance probability of the majority of the q parallel runs to $2^{-\alpha_\delta q}$. By choosing, say, $q = \lceil 3/\alpha_\delta \cdot (m(|x|)+1) \log_2 |\Sigma| \rceil$, we have $\sum_z 2^{-\alpha_\delta q} \leq 1/8 \sum_z |\Sigma|^{-m(|x|)-1} < 1/6$ since $m(|x|)$ is the upper bound on the length of messages sent in the protocol. Consequently, the acceptance probability of Arthur in the new protocol is not greater than $1/6 + 1/6 = 1/3$ if $x \notin L$. ∎

Corollary 5.4 $\mathbf{AM} \stackrel{\text{def}}{=} \mathbf{AM}(2) = \mathbf{AM}(2k)$ *for any constant $k \geq 1$.*

Intuitively it is perhaps somewhat surprising that the random coin tosses made by the verifier in composing his query can be made known to the prover before she needs to reply without affecting the language recognition capability of an interactive proof system. (For example, the interactive protocol for Graph Nonisomorphism seems to require that the coin tosses of Victor must remain hidden until Peggy has responded.) By a result of Goldwasser and Sipser [8], any language L that has an interactive proof system using at most $t(n)$ rounds also has an Arthur–Merlin proof system that decides L using at most $t(n)+2$ rounds, that is, $t(n)$ rounds, assuming that the collapse theorem applies.

[1] Consider for example Arthur playing in parallel *every* possible move against Merlin; clearly Merlin cannot obtain an advantage in this case. (Naturally playing every possible move in parallel is impossible if we wish the protocol to be well-formed.)

Theorem 5.5 *Let t be a polynomially bounded function. Then,* $\mathbf{IP}[2t(n)] \subseteq \mathbf{AM}(2(t(n)+4))$.

Proof sketch. (See [8] for the full proof.) Suppose $L \in \mathbf{IP}[2t(n)]$ is decided by an interactive proof system (V, P) with an exponentially small error probability in the input length. The idea of the proof is to construct an Arthur–Merlin proof system that simulates the original (V, P) protocol. The heart of the proof is the following Lemma that is used to commit Merlin to a run of the original protocol of (V, P) on input $x \in \Sigma^*$.

Lemma 5.6 *Let $b, k > 0$, $l > \max(b, 8)$, and suppose $C \subseteq \mathbb{Z}_2^k$. Randomly select l linear functions $H = \{h_1, \ldots, h_l\}$, $h_i : \mathbb{Z}_2^k \to \mathbb{Z}_2^b$, and l^2 strings $Z = \{z_1, \ldots, z_{l^2}\} \subseteq \mathbb{Z}_2^b$. Then,*

1. *If $b = 2 + \lceil \log_2 |C| \rceil$, then*
   a) $\Pr\big(|H(C)| \geq |C|/l\big) \geq 1 - 2^{-l}$.
   b) $\Pr\big(|H(C)| \cap Z \neq \emptyset\big) \geq 1 - 2^{-l/8}$.
2. a) $|H(C)| \leq l|C|$.
   b) *Suppose $|C| \leq 2^b/d$. Then,* $\Pr\big(|H(C)| \cap Z \neq \emptyset\big) \leq l^3/d$.

The Arthur–Merlin protocol that simulates the original (V, P) protocol is as follows. On the initial round Arthur makes an empty move and Merlin supplies the initial value of b for simulation of the (V, P) protocol. On each round i that simulates the original protocol Arthur first generates l linear functions and l^2 strings using the Lemma with $k = m(|x|)$ and a value of b provided by Merlin. These strings and functions (which constitute together one random string) are then sent to Merlin who replies with a round message pair y_i, z_i of the original protocol and a new value b for the next round. Arthur verifies that $h_j(y_i) \in Z$ for some $j = 1, \ldots, l$; otherwise Arthur rejects immediately. On the final round Arthur again generates the random linear functions and strings as before and sends them to Merlin, who replies with the random string r used by V in the original protocol run. Arthur verifies that $h_j(r) \in Z$ for some $j = 1, \ldots, l$ and that the (V, P) run to which Merlin has committed is indeed produced by r and results in acceptance. Finally, Arthur checks that the sum of all the b values sent by Merlin exceeds a certain threshold. If all these checks pass, Arthur accepts; otherwise Arthur rejects.

If $x \in L$, then the exponentially small error probability guarantees that Merlin has sufficiently many choices for the final random string r during every round of the protocol to enable him convince Arthur $2/3$ of the time. On the other hand, if $x \notin L$, then the exponentially small error probability combined with the Lemma guarantees that Merlin can only $1/3$ of the time find the necessary strings to convince Arthur so that the sum of the b values still exceeds the threshold. ■

Denote by $\mathbf{AM}(\text{poly})$ the class of all languages that are decidable by an Arthur–Merlin proof system.

Corollary 5.7 $\mathbf{IP} = \mathbf{AM}(\text{poly})$.

References

[1] L. Babai. Trading group theory for randomness. Proc. 17th ACM STOC, p. 421–429, ACM, New York, 1985.

[2] L. Babai and S. Moran. Arthur–Merlin games: A randomized proof system, and a hierarchy of complexity classes. *J. Comput. System Sci.* **36** (1988) no. 2, 254–276.

[3] S. A. Cook. The complexity of theorem-proving procedures. Proc. 3rd IEEE FOCS, p. 151–158, IEEE Computer Society Press, Los Alamitos, California, 1971.

[4] R. Diestel. *Graph Theory, 2nd Edition.* Graduate Texts in Mathematics no. 173. Springer-Verlag, New York, 2000.

[5] O. Goldreich. *Foundations of Cryptography: Basic Tools.* Cambridge University Press, Cambridge, England, 2001.

[6] O. Goldreich, S. Micali and A. Widgerson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM* **38** (1991) no. 1, 691–729.

[7] S. Goldwasser, S. Micali and C. Rackoff. The knowledge complexity of interactive proof systems. Proc. 17th ACM STOC, p. 291–304, ACM, New York, 1985.

[8] S. Goldwasser, M. Sipser. Private coins versus public coins in interactive proof systems. Proc. 18th ACM STOC, p. 59–68, ACM, New York, 1986.

[9] J. Köbler, U. Schöning and J. Torán. *The Graph Isomorphism Problem: Its Structural Complexity.* Birkhäuser, Boston, Massachusetts, 1993.

[10] L. A. Levin. Universal sequential search problems. *Problems of Information Transmission* **9** (1973) no. 3, 265–266.

[11] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, Reading, Massachusetts, 1994.

A Appendix

Lemma A.1 (The Chernoff Bound) *Suppose that $x_1, \ldots, x_n$ are independent random variables taking the values 0 and 1 with probabilities $1-p$ and p, respectively. Then, for all $\theta \geq 0$,*

$$\Pr\left(\sum_{i=1}^{n} x_i \geq (1+\theta)pn\right) \leq e^{pn(\theta-(1+\theta)\ln 1+\theta)}.$$

Proof. See the proof of [11, Lemma 11.9]. ∎

Lemma A.2 (Majority Vote) *Suppose that $x_1, \ldots, x_n$ are independent random variables taking the values 0 and 1 with probabilities $1-p$ and p, respectively, where $p < 1/2$. Then,*

$$\Pr\left(\sum_{i=1}^{n} x_i \geq n/2\right) \leq e^{-\alpha n},$$

for some constant $\alpha > 0$ that depends on p but is independent of n.

Proof. Note that $g(\theta) = \theta - (1+\theta)\ln 1 + \theta$ is negative if $\theta > 0$. Now apply the Chernoff Bound with $\theta = 1/(2p) - 1 > 0$. ∎