Definition (Probabilistically Checkable Proofs (PCP)). A Turing machine with direct access is a Turing machine with:

- a special state, called the reading state,
- a reading oracle,
- two special working tapes, called the direct access tape, and the address tape.

The machine never reads directly the content of the direct access tape (in the sense that the normal transitions of the machine are independent of the content of the direct access tape). This tape is only accessed via the reading oracle in the following way: when the machine goes in the reading state, the content of the address tape is interpreted as the binary representation of a position $i$ of the direct access tape. The reading oracle then provides in one step, the symbol in position $i$ of the direct access tape. (You can assume this symbol is stored in the control state, or in a special output tape of the reading oracle.)

A PCP($R(n), Q(n), T(n)$)-verifier is a probabilistic Turing machine with direct access to a tape called the proof tape over alphabet $\{0, 1\}^*$. On input $x$ of size $n$ and proof tape content $\pi$, the machine uses $R(n)$ random bits and works in the following three phases:

1. It first computes $Q(n)$ positions $p_1, \ldots, p_{Q(n)}$ (in binary) in time polynomial in $n$, and with no calls to the reading oracle (i.e. these positions are only a function of $x$ and the random tape content).

2. Then it makes $Q(n)$ calls to the reading oracle, to retrieve the symbols of the proof tape $\pi$ in positions $p_1, \ldots, p_{Q(n)}$.

3. Finally, it computes a boolean value (either accept or reject) in time $T(n)$ and with no calls to the reading oracle (i.e. the answer computed in this phase is only a function of $x$, the random tape content, and the symbols $\pi[p_1], \ldots, \pi[p_{Q(n)}]$).

The class PCP($R(n), Q(n), T(n)$) is the set of languages $L$ such that there exists a PCP($R(n), Q(n), T(n)$)-verifier $V$ such that:

- if $x \in L$, there exists a proof $\pi \in \{0, 1\}^*$ such that $Pr_r[V(x, \pi, r) \text{ rejects }] = 0$;
- if $x \notin L$, then for all $\pi \in \{0, 1\}^*$ $Pr_r[V(x, \pi, r) \text{ accepts }] \leq 1/2$.

Where the probability is computed over all random tape contents $r$ of size $R(n)$.

Exercise 1 (PCP witnessing). Let PCP($k_1 \cdot \log n, Q(n), T(n)$) be defined as PCP($k_1 \cdot \log n, Q(n), T(n)$) except that only proofs $\pi$ of size $n^{k_1} \cdot Q(n)$ are considered, and addresses computed by the verifier have $\log(n^{k_1} \cdot Q(n))$ bits. Prove that PCP($k_1 \cdot \log n, Q(n), T(n)$) = PCP($k_1 \cdot \log n, Q(n), T(n)$).
Solution 1. Straightforwardly, we have \( \text{PCP}(k \cdot \log n, Q(n), T(n)) \supseteq \text{PCP}'(k \cdot \log n, Q(n), T(n)) \). Consider now a \( \text{PCP}(k \cdot \log n, Q(n), T(n))-\text{verifier} V \). On an input \( x \) of size \( n \), for any random tape content of size \( k \cdot \log n \), at most \( Q(n) \) different positions are queried. If we consider the set \( \text{Pos}(x) = \{ i \mid \exists r \in \{0,1\}^{k \cdot \log n}, \exists j \leq Q(n), p_j = i \text{ on random tape content } r \text{ on input } x \} \) of all positions of the proof tape used by the verifier \( V \) on input \( x \), we have \( |\text{Pos}(x)| \leq n^k \cdot Q(n) \). Hence, one can construct an injective function \( f_x : \text{Pos}(x) \mapsto \{0, \ldots, n^k \cdot Q(n) - 1\} \) in polynomial time (\( Q(n) \) is polynomial since we have to able to compute \( Q(n) \) positions in polynomial time) and a \( \text{PCP}'(k \cdot \log n, Q(n), T(n))-\text{verifier} V' \) that simulates \( V \) and instead of querying and using tape content of position \( i \in \text{Pos}(x) \) on input \( x \), it queries and uses position \( f_x(i) \in \{0, \ldots, n^k \cdot Q(n) - 1\} \). The languages accepted by \( V \) and \( V' \) are the same since the function \( f_x \) is injective. The proof tape content for \( V' \) only needs to have \( \log(n^k \cdot Q(n)) \) bits.

Exercise 2 (PCP and non-deterministic classes). Prove that, with \( R(n) = \Omega(\log n) \), we have \( \text{PCP}(R(n), Q(n), T(n)) \subseteq \text{NTIME}(2^{O(R(n))} \cdot Q(n) \cdot (T(n) + \text{poly}(n))) \).

Solution 2. Consider a \( \text{PCP}(R(n), Q(n), T(n))-\text{verifier} V \). With the same idea than for the previous question, we assume without loss of generality that every position \( p_i \) queried by the verifier \( V \) is at most \( 2^{R(n)} \cdot Q(n) \) (in time \( 2^{R(n)} \cdot Q(n) \), we can construct a table to associate with each initial position the corresponding position lower than \( 2^{R(n)} \cdot Q(n) \). Consider now the non-deterministic algorithm that, on an input \( x \) of size \( n \) guesses \( 2^{R(n)} \cdot Q(n) \) bits (and stores them). Then, it enumerates all possible random tape content of size \( R(n) \) and simulates the execution of \( V \) with the bits guessed as the content of the proof tape needed while maintaining a counter \( c \) that corresponds to the number of accepting random tapes. The algorithm accepts if and only if \( c > 1/2^{R(n) - 1} \). This runs in time \( 2^{O(R(n))} \cdot Q(n) \cdot (T(n) + \text{poly}(n)) \) (the final \( \text{poly}(n) \) comes from the polynomial time taken in the first step, to compute the \( Q(n) \) positions) and accepts if and only if there is a proof tape content leading to acceptance.

Exercise 3 (Known classes). What are these versions of \( \text{PCP} \) equal to (in term of known complexity classes)?

\[
\bigcup_{c \in \mathbb{N}, T(n) \text{ a polynomial}} \text{PCP}(0, c \cdot \log n, T(n))
\]

\[
\bigcup_{Q(n), T(n) \text{ polynomials}} \text{PCP}(0, Q(n), T(n))
\]

\[
\bigcup_{R(n), T(n) \text{ polynomials}} \text{PCP}(R(n), 0, T(n))
\]

Solution 3. 1. This is in fact \( \text{P} \). The direct inclusion comes from the fact that a polynomial time algorithm can simulate all the different possible calls to the proof content tape (there are polynomially many as an exponential of a logarithm) and check that there exists one that leads to acceptance (note that the calls do not depend on any random bit). The reverse inclusion is straightforward: one can simulate a polynomial time algorithm by just ignoring the randomness and the calls to the proof tape.
2. This is in fact $\text{NP}$. The direct inclusion is straightforward. As for the reverse inclusion, the number of non deterministic calls of a non deterministic Turing machine may depend on the result of the queries considered, however in any case there are polynomially many. Hence, it suffice to consider as many non deterministic bits as the worst case execution time of the Turing machine and simulate the execution of the non- deterministic Turing machine.

3. This is in fact $\text{coRP}$ (it comes directly from the definition).

Exercise 4 (Graph non-isomorphism). Show that $\text{ISO} \in \text{PCP}(p(n), 1, c)$ for some polynomial $p$ and constant $c$.

Solution 4. Consider a pair of graphs $(G_0, G_1)$ with $n$ vertices. In the proof tape, the verifier $V$ expects, for each graph $H$ with $n$ vertices, that $\pi[H] = b \in \{0, 1\}$ where $H$ is isomorphic to $G_b$ (if $H$ is isomorphic to neither or both $G_0$ and $G_1$, the value of $\pi[H]$ is not specified) (that is, to an exponential number (in $n$) of natural indexes corresponds an adjacency matrix of an $n$ vertices graph). Then, a verifier $V$ randomly picks $b \in \{0, 1\}$ and a permutation $\nu$ of the vertices, computes $H = \nu(G_b)$ accordingly, queries $\pi[H]$ and checks that $b = \pi[H]$. Then, if $G_0$ and $G_1$ are not isomorphic, an honest proof tape will always lead to acceptance whereas, if they are isomorphic, the content of $\pi[H]$ is not specified and there is at most probability $1/2$ of acceptance.

Exercise 5 (Multi-prover protocol).

Definition. Let $P_1, \ldots, P_k$ be infinitely powerful machines whose output is polynomially bounded. Let $V$ be a probabilistic polynomial-time machine. $V$ is called the verifier, and $P_1, \ldots, P_k$ are called the provers.

A round of a multi-prover interactive protocol on input $x$ consists of an exchange of messages (i.e. words over a given alphabet) between the verifier and the provers, and works as follows:

- The verifier $V$ is executed on an input consisting of $x$, the history of all previous messages exchanged with all provers (both sent and received messages), and a random tape content of size polynomial in $|x|$. The output of the verifier is computed in time polynomial in $|x|$, and consists of messages to some or all of the provers.

- Each message $q_i$ sent from the verifier to prover $P_i$ is followed by an answer $a_i$, of size polynomial in $|x|$, sent from the prover $P_i$ to the verifier. The answer $a_i$ is computed by $P_i$ on input consisting of $x$ and the history of all messages previously exchanged between the verifier and the prover $P_i$ (and only $P_i$).

- Alternatively the verifier may decide not to produce messages, and terminates the protocol by either accepting or rejecting, based on the input $x$ and the history of all previous messages exchanged with all provers.

You can view the protocol as executed by the verifier sharing communication tapes with each $P_i$, where different provers $P_i$ and $P_j$ (for $i \neq j$) have no tapes they can both access, besides the input tape. In a round the verifier stores each message $q_i$ to prover $P_i$ on the $i$-th communication tape, shared between the prover and $P_i$. The answer of $P_i$ is put on tape $i$ as well. The verifier has access to the input and all communication tapes, while each prover $P_i$ has access only to the input and tape $i$.

$P_1, \ldots, P_k$ and $V$ form a multi-prover interactive protocol for a language $L$ if the execution of the protocol between $V$ and $P_1, \ldots, P_k$ terminates after a polynomial number of rounds (in the size of the input $x$) and:
• if \( x \in L \), then \( \Pr[(V,P_1,\ldots,P_k) \text{ accepts } x] > 1 - 2^{-n} \);
• if \( x \notin L \), then for all provers \( P'_1,\ldots,P'_k \), \( \Pr[(V,P'_1,\ldots,P'_k) \text{ accepts } x] < 2^{-n} \);

where the probability is computed over all possible random choices of \( V \).

In this case, we denote \( L \in \text{MIP}_k \). The number of provers \( k \) need not be fixed and may be a polynomial in the size of the input \( x \). We say that \( L \in \text{MIP} \) if \( L \in \text{MIP}_{p(n)} \) for some polynomial \( p \). Clearly \( \text{MIP}_1 = \text{IP} = \text{PSPACE} \) (as you will see in the lecture), but allowing more provers makes the interactive protocol model potentially more powerful.

1. Let \( M \) be a probabilistic polynomial-time Turing machine with access to a function oracle. A language \( L \) is accepted by \( M \) iff:

   • if \( x \in L \), then there exists an oracle \( O \) s.t. \( M^O \) accepts \( x \) with probability greater than \( 1 - 2^{-n} \);
   • if \( x \notin L \), then for any oracle \( O' \), \( M^{O'} \) accepts \( x \) with probability lower than \( 2^{-n} \).

Show that \( L \in \text{MIP} \) if and only if \( L \) is accepted by a probabilistic polynomial time oracle machine.

2. Show that \( \text{MIP} = \text{MIP}_2 \) (assuming we can use error-reduction).

Solution 5. 1. Suppose that \( L \in \text{MIP} \) with a verifier \( V \). We define the probabilistic polynomial time Turing machine \( M \) that simulates \( V \). However, \( M \) can only call an oracle that only gives yes-or-no answers, not a polynomially-bounded size response like a prover. Therefore, we call an oracle to get each bit of the response of the prover. That is, when \( V \) sends a message to a prover, \( M \) asks the query \((x,i,j,l,q_{i,1},\ldots,q_{i,j})\) to the oracle, which, in turn, will be used as the \( l \)-th bit of the \( j \)-th message of \( V \) sent to prover \( i \) with \( q_{i,1},\ldots,q_{i,j} \) the first \( j \) message sent from the verifier to prover \( i \). Then, \( M \) accepts iff the verifier \( V \) does. We get:

   • If \( x \in L \), then the oracle \( O \) that faithfully simulates the calls to the different provers by sending the corresponding bits ensures that the probability of acceptance is greater than \( 1 - 2^{-n} \) (since \( L \in \text{MIP} \)).
   • Suppose now that \( x \notin L \) and that there exists an oracle \( O' \) such that the probability of acceptance by \( M^{O'} \) is at least \( 1 - 2^{-n} \). Then, we can construct the prover \( (P_i) \), by using the oracle \( O' \) (the same way \( M \) does, by calling it bit by bit). It would follow that the probability of acceptance is at least \( 1 - 2^{-n} \), hence the contradiction since \( L \in \text{MIP} \).

Suppose now that \( L \) is accepted by a oracle probabilistic polynomial-time Turing machine \( M \) running in time \( n^c \) (with \( n \) the size of the input). We use \( 2 \cdot n^{c+1} \) provers. We consider a verifier \( V \) that first chooses randomly (and uniformly) an ordering of these provers. Then, the verifier \( V \) simulates \( M \) and, each time \( M \) makes a query to the oracle, \( V \) asks the question to the next \( 2n \) provers. If they are unanimous, \( V \) proceeds with the simulation of \( M \) with the common answers of the provers as answer of the oracle, otherwise it rejects. Then, \( V \) accepts iff \( M \) does (assuming that all queries to the oracle passed successfully). Note that indeed \( 2 \cdot n^{c+1} \) provers suffice.

4
2. Consider a language \( L \) and a verifier \( V \). If \( x \in L \), there exists an oracle \( O \) such that the probability of acceptance is at least \( 1 - 2^{-n} \), hence if the provers faithfully simulate the answer of the oracle \( O \), then the probability of acceptance will be the same.

Suppose now that \( x \notin L \). Consider any provers \( P_1, \ldots, P_{2n^c+1} \) and the oracle \( O' \) that answers like the majority of the \( 2 \cdot n^c+1 \) provers. Now, there are two possibilities: either all oracle queries in the simulation is consistent with \( O' \), or there is at least one difference. In the first case, the probability of acceptance is less than \( 2^{-n} \) (by definition of the acceptance condition of a probabilistic Turing machine). In the other case, the probability of acceptance is bounded by the probability that at least one oracle query is inconsistent with \( O' \) and it did not reject immediately. Therefore, the probability (for a fixed sequence of \( 2 \cdot n \) provers) that the provers are unanimously inconsistent with \( O' \) is lower than \( 2^{-2n} \). By summing over all \( n^c \) queries, it follows that the probability that there is at least one inconsistency not rejected is lower than \( n^c \cdot 2^{-2n} \).

Overall, we have \( \Pr(\text{accept}) < 2^n + n^c \cdot 2^{-2n} < 2^{-n+1} \) for \( n \) large enough.

We need to reduce that probability even further. Hence, we consider \( V' \) that simulates \( V \) three times in a row and answers according to the majority. In that case:

- If \( x \in L \), we have:
  \[
  \Pr(\text{accept}) > (1 - 2^{-n})^3 + 3(1 - 2^{-n})^2 \cdot 2^{-n} > 1 - 3 \cdot 2^{-2n} > 1 - 2^{-n}
  \]

- If \( x \notin L \), we have:
  \[
  \Pr(\text{accept}) < (2^{-n+1})^3 + 3(2^{-n+1})^2 \cdot (1 - 2^{-n+1}) < 2^{-n}
  \]

2. Consider a language \( L \in \text{MIP} \) with an arbitrary number of provers \((P_i)_i \) (but polynomially bounded in the size of the input) with a verifier \( V \). We want to simulate what happens with these provers with only two provers. The idea is to use one prover to simulate the calls to all these provers and then simulating \( V \) with these calls. However, since these queries are made to a single prover, in the simulation, the provers simulated may interact with each other (which is not allowed in a MIP protocol), hence we check with the second prover that the answer for a given prover \( P_i \) can be obtained with a single prover that does not interact with the other. However, we can only do one call to that second prover, otherwise the second call would have the information of the first call (i.e. there would an interaction between provers). Therefore, we randomly choose the prover \( P_i \) to check, and we use error reduction. More formally, assume that the verifier \( V \) uses \( k \) provers in time \( n^c \) on an input \( x \). We consider a verifier \( V' \) generating a random word \( r \) of length \( n^c \) and sending it to the first prover. This prover answers with the complete interaction of all \( k \) provers with the verifier \( V \) over the whole computation of \( V \) on \( x \) with the random word \( r \). Then, \( V' \) simulates \( V \) with the given interaction with the provers. If \( V \) rejects, so does \( V' \). Otherwise, it randomly picks a number \( j \) between 1 and \( k \) and simulates the complete interaction with prover \( P_j \) with queries to the second prover (while indicating the number \( j \) in the exchange). If it differs from what was sent by the first prover, \( V' \) rejects, otherwise it accepts. Then, we have:
• If $x \in L$, then the two provers can just faithfully simulate the other provers and get a probability greater than $1 - 2^{-n}$.

• If $x \notin L$, either the first prover simulates faithfully the other provers (in which case, the probability of accepting is below $2^{-n}$), otherwise at least one prover cannot have this interaction without exchanging with other provers (in which case, the probability to reject is at least $1/k$ if the prover chosen at random is the faulty one). Overall, the probability to accept is lower than:

$$Pr[\text{accept}] < 2^{-n} + (1 - 1/k)$$

We conclude by using error reduction with $k^2$ rounds (where, at each round, one new random word is chosen and therefore the knowledge of the previous rounds is not an issue). Note this is possible since $k$ is bounded polynomially in $n$.

Exercise 6 (PCP, MIP and NEXPTIME). Prove that

$$\bigcup_{R(n),Q(n),T(n) \text{ polynomials}} \text{PCP}(R(n),Q(n),T(n)) \subseteq \text{MIP} \subseteq \text{NEXPTIME}$$

(Hint. It is possible to prove (but you are not required to) that, as with IP, one can equivalently use perfect completeness in the definition of MIP. That is, in the case $x \in L$, we require that the protocol accepts with probability 1, rather than at least $1 - 2^{-q(n)}$. In this exercise use the definition of MIP with perfect completeness, and the corresponding notion of probabilistic oracle machine.)

Remark. Indeed MIP and this version of PCP coincide with NEXPTIME, but you are not required to prove the opposite inclusions.

Solution 6. With the definition of MIP with oracle, the first inclusion is straightforward: every call to the proof content tape can be simulated by calls to an oracle which are (as in this case) independent from one another (which is different from calls to a prover from an interactive protocol).

Consider a language $L$ decided by an oracle probabilistic machine $M$ running in time $n^c$. This machine makes at most $n^c$ calls to the oracle. Note that it is not enough to only guess the result of the $n^c$ queries, and then counting the number of accepting runs over all possible random word of the appropriate size (which would yield a polynomial space algorithm) since the calls to the oracle may depend on the random bits read. Hence, we consider a non-deterministic exponential time machine $M'$ that guesses the oracle $O$, or more precisely that guesses the answer of the oracle $O$ to all possible $2^{n^c} - 1$ queries that the machine $M$ can make (since the size of a query is at most $n^c$). Then, for a word $r$ of size $n^c$, denote $f$ the function such that $f(x, O, r) = 1$ if the simulation of $M$ on the input $x$ with $r$ used as random tape accepts, and 0 otherwise. Then, le machine $M'$ accepts $x$ iff:

$$S = \sum_{r \in \{0,1\}^{n^c}} f(x, O, r) \geq 2^{n^c - 1}$$

Then, if $x \in L$, there exists an oracle $O$ such that $S > (1 - 2^n) \cdot 2^{n^c} \geq 2^{n^c - 1}$ and if $x \notin L$, for all oracle $O'$, we have $S < 2^{-n} \cdot 2^{n^c} \leq 2^{n^c - 1}$. Therefore, $L \in \text{NTIME}(2^{O(n^c)}) \subseteq \text{NEXPTIME}$. 

6