Definition 0.1 (Bounded-Error Probabilistic Polynomial Time (BPP)). A language \( L \in \text{BPP} \) if there is a probabilistic Turing Machine \( M \) that takes in random bitstrings \( r \) such that

\[
\forall x \in L, \Pr[M(x, r) = L(x)] \geq \frac{2}{3}
\]

where \( L(x) \) is 1 if \( x \in L \) and is 0 otherwise.

1 Relations between complexity classes

Remark 1.1. \( \text{P} \subseteq \text{BPP} \)

For any language \( L \in \text{P} \) and the deterministic machine \( M \) that decides \( L \), just augment \( M \) such that it takes in an additional random string \( r \), but ignores it during execution. Then \( M(x, r) = L(x) \) always.

Remark 1.2. \( \text{RP} \subseteq \text{NP} \)

Consider any language \( L \in \text{RP} \) and the probabilistic \( \text{RP} \) machine \( M \) that decides \( L \). For any \( x \in L \), there must be some \( r \) of \( \text{poly}(|x|) \) size such that \( M(x, r) = 1 \). Because \( M \) operates deterministically given \( r \) and within \( \text{poly}(|x|) \) time, we can say \( r \) is a certificate for \( x \in L \) for the verifier \( M \). We know \( M \) is a correct verifier since if \( x \notin L \), no such certificate exists.

Theorem 1.3. \( \text{BPP} \subseteq \text{P}/\text{poly} \)

We begin with a failed proof:

Failed proof attempt. Assuming \( L \in \text{BPP} \), there is some poly-time machine \( M \) such that \( \forall x \in L, \Pr[M(x, r) = L(x)] \geq \frac{2}{3} \). Recall \( \text{P}/\text{poly} \) is the set of languages that can be computed by machines that take poly-sized advice strings. Thus we can try to define an advice TM \( \hat{M} \) that simulates \( M \) on a ‘good random string \( r \)’ that is given as advice.

This fails since the advice string directly depends on \( x \) (it is not clear if the same random string works for all inputs of a given length).

However, the above strategy can be easily fixed by defining \( \text{BPP} \) with a \( 1 - 2^{-|x|+1} \) threshold (which is possible via error reduction) instead of a \( \frac{2}{3} \) threshold.

Proof. Define

\[\text{BAD}_x := \{ r : M(x, r) \neq L(x) \}\]

From the threshold, we have for a random string \( r \)

\[\forall x, \Pr[r \in \text{BAD}_x] \leq 2^{-|x|+1}\]
For \( x \in \{0, 1\}^n \), this means that

\[
\Pr[\exists x, r \in \text{BAD}_x] \leq \frac{2^n}{2^{n+1}} < 1
\]

and there is some string \( r' \) such that \( M(x, r') = L(x) \) always. We simply pick \( M \) to be the \( \mathbb{P}/\text{poly} \) machine and \( r' \) to be its advice.

\[ \square \]

**Theorem 1.4.** \( \text{BPP} \subseteq \Sigma_2 \cap \Pi_2 \)

**Proof.** Let \( x \in \{0, 1\}^n \) and define \( \text{BPP} \) using the same threshold as before. Define

\[ 1_x := \{ r : M(x, r) = 1 \} \]

and shifts \( V_i \in \{0, 1\}^n \) such that the set

\[ 1_x + V_i := \{ x \oplus V_i : x \in 1_x \} \]

where \( \oplus \) is an elementwise XOR. If \( x \in L \), most \( r \) will result in \( M(x, r) = 1 \), and few shifts will be required to cover \( \{0, 1\}^n \). If \( x \not\in L \), many shifts will be required to cover \( \{0, 1\}^n \). Specifically, for \( x \not\in L \),

\[ |1_x| \leq 2^n - (n+1) \]

so an exponential number of shifts are required.

We can encode this constraint as

\[ x \in L \iff \exists V_1 \ldots V_t, \forall y \in \{0, 1\}^n, y \in \bigcup_{i=1}^{t} (1_x + V_i) \]

Using the fact that \( a + b = c \iff a = b + c \) in \( \mathbb{F}_2 \), this can be rewritten as

\[ x \in L \iff \exists V_1 \ldots V_t, \forall y \in \{0, 1\}^n, \bigvee_{i=1}^{t} (y + V_i) \in 1_x \]

which can also be rewritten as

\[ x \in L \iff \exists V_1 \ldots V_t, \forall y \in \{0, 1\}^n, \bigvee_{i=1}^{t} M(x, y + V_i) \]

This can be translated into a polynomial sized boolean formula using Cook-Levin assuming \( t \) is bounded by some polynomial of \( n \). The only thing left to show is that we can have such a bound. Consider if \( V_1 \ldots V_t \) are picked independently and at random. We want to bound the probability that

\[ \exists y \in \{0, 1\}^n, \bigwedge_{i=1}^{t} (y + V_i) \not\in 1_x \]

For a fixed \( y \), the probability that \( \bigwedge_{i=1}^{t} (y + V_i) \not\in 1_x \) is \( 2^{-t(n+1)} \), by the \( \text{BPP} \) threshold and independence. Hence, we have

\[ \exists y \in \{0, 1\}^n, \bigwedge_{i=1}^{t} (y + V_i) \not\in 1_x \leq \frac{2^n}{2^{t(n+1)}} \]

and picking \( t = n^c \) pushes this probability to 0 and gives us a \( \text{poly}(n) \) bound on \( t \).

\[ \square \]