Quantifier Characterization of Complexity Classes

Antonis Antonopoulos
National Technical University of Athens
aanton@corelab.ntua.gr

1 Complexity Classes

1.1 Introduction

We present an alternative characterization of complexity classes using *quantifiers*, and especially those needed for the quantification implied by the definition of each class. This notation provides a uniform description of complexity classes defined in various contexts (deterministic, probabilistic, interactive), and we'll be able to obtain immediate relations and inclusions among them.

For complexity classes like **P**, **NP** and their generalizations, the classical existential and universal quantifiers suffice, but in order to describe classes using Probabilistic Turing Machines, we will need a new one, which assures that a computation has "probabilistic" advantage:

Definition 1.1 (Majority Quantifier)**.** *Let* $R : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ *be a predicate, and* ε *a rational number in* $\left(0, \frac{1}{2}\right)$*. We denote by* $(\exists^+ y, |y| = k)R(x,y)$ *the following predicate:*

> *"There exist at least* $\left(\frac{1}{2} + \varepsilon\right) \cdot 2^k$ *strings* y *of length* k *for which* $R(x,y)$ *holds."*

We call $\exists^+$ *the* overwhelming majority *quantifier.*

The overwhelming quantifier provides a "threshold" for the number of certificates, assuring that the fraction of 2^k possible strings in $\{0,1\}^k$ (that is, of length k) which accepts the computation (or satisfies the predicate R) is bounded away from 50% by a fixed amount ε.

We can generalize this quantifier by attaching the fraction of accepting computations as a parameter. That is, $\exists_r^+$ means that the fraction r of the possible certificates of a certain length satisfy the predicate for the certain input. It is easy to see that: $\exists^+ = \exists_{1/2+\varepsilon}^+ = \exists_{2/3}^+ = \exists_{3/4}^+ = \exists_{0.99}^+ = \exists_{1-2^{p(|x|)}}^+$,

where $|x|$ denotes the length of the input x. Intuitively, this means that we can "increase" the fraction of the accepting branches (the acceptance probability) by indepedent repetitions of the computation.

We also introduce a new notation for an arbitrary complexity class, which utilizes the quantifiers' role in the classical definition:

Definition 1.2. *We denote as* $\mathcal{C} = (Q_1/Q_2)$*, where* $Q_1, Q_2 \in \{\exists, \forall, \exists^+\}$*, the class* $\mathcal{C}$ *of languages* L *satisfying:*

- $x \in L \Rightarrow Q_1 y \ R(x, y)$
- $x \notin L \Rightarrow Q_2 y \ \neg R(x, y)$

In the above definition, we easily notice that:

$$co\mathcal{C} = co(Q_1/Q_2) = (Q_2/Q_1)$$

So, using the classical existential and universal quantifiers we can define the basic complexity classes, by implying their definitional properties. For example, for languages in class **P** the computation path either accepts, either rejects. So, it is easy to see that $\mathbf{P} = (\forall/\forall)$.

On the other hand, for languages in class **NP** there is a computation tree for each input, and we accept it if *there is* an accepting branch, or we reject it if *all* the branches reject. Hence, we have that: $\mathbf{NP} = (\exists/\forall)$. The complementary class *co***NP** can be also defined as $co\mathbf{NP} = (\forall/\exists)$.

A family of complexity classes that are naturally defined by alternating quantifiers is the Polynomial Hierarchy. These classes can be considered as a natural generalization of **NP**. Recall that:

Definition 1.3 (Polynomial-Time Hierarchy)**.** *A language* $L \in \Sigma_k^p$*,* $k \in \mathbb{N}$*, iff there exists a polynomial-time computable predicate* $R(x, y_1, y_2, \ldots, y_k)$*, such that, for* $|y_i| \leq p(n)$*,* $i \in \{1, \ldots, k\}$*,* $p \in poly(n)$*:*

$$x \in L \Leftrightarrow \exists y_1 \forall y_2 \exists y_3 \cdots Q_k y_k \ R(x, y_1, y_2, \ldots, y_k)$$

where Q_k *is* $\exists$ *if* k *is odd, and* $\forall$ *if* k *is even.*

Also, a language $L \in \Pi_k^p$ *iff there exists a polynomial-time computable predicate* $R(x, y_1, y_2, \ldots, y_k)$*, such that, for* $|y_i| \leq p(n)$*,* $i \in \{1, \ldots, k\}, p \in poly(n)$*:*

$$x \in L \Leftrightarrow \forall y_1 \exists y_2 \forall y_3 \cdots Q_k y_k \ R(x, y_1, y_2, \ldots, y_k)$$

where Q_k *is* $\forall$ *if* k *is odd, and* $\exists$ *if* k *is even.*

An equivalent definition can be given recursively using oracles: $\Sigma_k^p = \mathbf{NP}^{\Sigma_{k-1}^p}$ and $\Pi_k^p = co\mathbf{NP}^{\Sigma_{k-1}^p}$, while $\Sigma_0^p = \Pi_0^p = \mathbf{P}$. So, we have that $\Sigma_1^p = \mathbf{NP}$, $\Pi_1^p = co\mathbf{NP}$, $\Sigma_2^p = \mathbf{NP}^{\mathbf{NP}}$ and so on.
Using quantifier notation, we can re-define these complexity classes as:

- $\Sigma_2^p = (\exists\forall/\forall\exists)$, $\Pi_2^p = (\forall\exists/\exists\forall)$, and in general:
- $\Sigma_k^p = (\exists\forall\cdots Q_m)/\forall\exists\cdots Q_n)$, where:
  - Q_m represents $\exists$, if k is odd, or $\forall$, if k is even, and
  - Q_n represents $\forall$, if k is odd, or $\exists$, if k is even.
- $\Pi_k^p = (\forall\exists\cdots Q_m/\exists\forall\cdots Q_n)$, where:
  - Q_m represents $\forall$, if k is odd, or $\exists$, if k is even.
  - Q_n represents $\exists$, if k is odd, or $\forall$, if k is even.

1.2 Randomized Classes

Using the overwhelming majority quantifier, the following characterizations are immediate from the definition of each class:

- **BPP** (Bounded two-sided error, *"Monte-Carlo"*):

  By **BPP** 's definition we have:

  $$\begin{cases} x \in L \Rightarrow \mathbf{Pr}\left[accept\right] \geq 2/3 \\ x \notin L \Rightarrow \mathbf{Pr}\left[reject\right] \geq 2/3 \end{cases} \Rightarrow$$

  $$\begin{cases} x \in L \Rightarrow \mathbf{Pr}\left[R(x)\right] \geq 2/3 \\ x \notin L \Rightarrow \mathbf{Pr}\left[\neg R(x)\right] \geq 2/3 \end{cases} \text{, for a predicate } R \in \mathbf{P} \Rightarrow$$

  $$\begin{cases} x \in L \Rightarrow \exists^+ y\ R(x,y) \\ x \notin L \Rightarrow \exists^+ y\ \neg R(x,y) \end{cases} \Rightarrow \mathbf{BPP} = (\exists^+/\exists^+)$$

- **RP** (Bounded one-sided error, *"Atlantic City"*):
  Similarly:

  $$\begin{cases} x \in L \Rightarrow \mathbf{Pr}\left[accept\right] \geq 2/3 \\ x \notin L \Rightarrow \mathbf{Pr}\left[reject\right] = 1 \end{cases} \Rightarrow$$

  $$\begin{cases} x \in L \Rightarrow \mathbf{Pr}\left[R(x)\right] \geq 2/3 \\ x \notin L \Rightarrow \mathbf{Pr}\left[\neg R(x)\right] = 1 \end{cases} \text{, for a predicate } R \in \mathbf{P} \Rightarrow$$

  $$\begin{cases} x \in L \Rightarrow \exists^+ y\ R(x,y) \\ x \notin L \Rightarrow \forall y\ \neg R(x,y) \end{cases} \Rightarrow \mathbf{RP} = (\exists^+/\forall)$$

- Obviously, $co\mathbf{RP} = (\forall/\exists^+)$

So, we have created alterative definitions for the most usual complexity classes. Now, we can explore what kind of "operations" we can perform with these quantifiers. Firstly, we determine when we can swap $\forall$ and $\exists^+$:

Lemma 1.1 (Swapping Lemma)**.** *Let $R(x,y,z)$ be a predicate that holds only if $|y| = |z| = p(n)$ for some polynomial p, where $n = |x|$, and let C be a set of strings such that $\forall v \in C\ |v| = p(n)$ and $|C| \leq p(n)$. Then, for $|y| = |z| = p(n)$:*

i. $\forall y \exists^{+} z\ R(x,y,z) \Rightarrow \exists^{+} C \forall y\ \bigvee_{z \in C} R(x,y,z)$

ii. $\forall z \exists^{+}_{1-2^{-n}} y\ R(x,y,z) \Rightarrow \forall C \exists^{+} y\ \bigwedge_{z \in C} R(x,y,z)$

Proof: (i) Assume that $\forall y \exists^{+} z\ R(x,y,z)$ holds. Let $p \in poly(n)$ such that for all y with $|y| \leq p(n)$ and considering only z with $|z| \leq p(n)$: $\mathbf{Pr}\left[\{z|\ R(x,y,z)\}\right] > \frac{1}{2} + \varepsilon$. Also, let $q(n) = p(n) + 3$. We will estimate the probability of the event $\neg \forall y\ \bigvee_{z \in C} R(x,y,z)$:

$$\mathbf{Pr}\left[\left\{C \mid\ \exists y : \bigwedge_{z \in C} \neg R(x,y,z)\right\}\right] = \mathbf{Pr}\left[\bigcup_{|y| \leq p(n)} \left\{C \mid \bigwedge_{z \in C} \neg R(x,y,z)\right\}\right]$$

$$\leq \sum_{|y| \leq p(n)} \mathbf{Pr}\left[\left\{C \mid \bigwedge_{z \in C} \neg R(x,y,z)\right\}\right] \leq \sum_{|y| \leq p(n)} \prod_{i=1}^{q(n)} \frac{1}{2} \leq 2^{p(n)+1} \cdot \left(\frac{1}{2}\right)^{q(n)} \leq \frac{1}{4}$$

Note that the predicate $R'(x,y,z) = \bigvee_{z \in C} R(x,y,z)$ is polynomial-time computable, therefore for most of the C: $\bigvee_{z \in C} R(x,y,z)$, that is $\exists^{+} C \forall y\ \bigvee_{z \in C} R(x,y,z)$.

(ii) Without loss of generality, we can assume that $\forall x \forall z\ \mathbf{Pr}\left[\{z|\ R(x,y,z)\}\right] \geq 1 - 1/2^{p(n)}$ for some $p \in poly(n)$. So, for any z, $|z| = p(n)$, we have that $\mathbf{Pr}\left[\neg R(x,y,z)\right] \leq 2^{p(n)}$. For a given C, $|C| \leq q(n)$:

$$\mathbf{Pr}\left[\left\{y \mid \bigvee_{z \in C} \neg R(x,y,z)\right\}\right] \leq \sum_{z \in C} \mathbf{Pr}\left[\{y|\neg R(x,y,z)\}\right] \leq \frac{q(n)}{2^{p(n)}} < \frac{1}{4}$$

for sufficiently large n. Therefore, we have that $\forall C \exists^{+} y\ \bigwedge_{z \in C} R(x,y,z)$. $\square$

The above lemma, can be viewed in terms of a binary matrix A of size $2^{p(n)} \times 2^{p(n)}$, with $A(y,z) \Leftrightarrow R(x,y,z)$. The (i) part states that if every row of A has more than $(2/3)p(n)$ many 1's, then for the majority of the choices of $p(n)$ many columns, every row of A contains at least one 1 in these columns. Similarly for part (ii).

We can prove, using the Swapping Lemma, an alternative, "decisive" characterization of **BPP**, stated in the following theorem:

Theorem 1.2 (BPP Theorem)**.** *The following are equivalent:*

i. $L \in \mathbf{BPP}$.

ii. There exists a polynomial-time computable predicate R and a polynomial p, such that for all x, with $|x| = n$, and $|y| = |z| = p(n)$:

$$x \in L \Rightarrow \exists^{+} y \forall z \ R(x, y, z)$$

$$x \notin L \Rightarrow \forall y \exists^{+} z \ \neg R(x, y, z)$$

iii. There exists a polynomial-time computable predicate R and a polynomial p, such that for all x, with $|x| = n$, and $|y| = |z| = p(n)$:

$$x \in L \Rightarrow \forall y \exists^{+} z \ R(x, y, z)$$

$$x \notin L \Rightarrow \exists^{+} y \forall z \ \neg R(x, y, z)$$

Proof: $(i \Rightarrow ii)$ Let $L \in \mathbf{BPP}$. Then, by definition, there exists a polynomial-time computable predicate Q and a polynomial q such that for all x's of length n:

$$x \in L \Rightarrow \exists^{+} y \ Q(x, y)$$

$$x \notin L \Rightarrow \exists^{+} y \ \neg Q(x, y)$$

Using Lemma 1.1(i) we have[1], for all x's of length n and for some $y, z, |y| = |z| = q(n)$:
$x \in L \Rightarrow \exists^{+} z \ Q(x, z) \Rightarrow \forall y \exists^{+} z \ Q(x, y \oplus z) \Rightarrow \exists^{+} C \forall y \, [\exists (z \in C) \ Q(x, y \oplus z)]$, where C denotes (as in the Swapping Lemma's formulation) a set of $q(n)$ strings, each of length $q(n)$.
On the other hand, by using Lemma 1.1(ii) we similarly have:
$x \notin L \Rightarrow \exists^{+} y \ \neg Q(x, z) \Rightarrow \forall z \exists^{+} y \ \neg Q(x, y \oplus z) \Rightarrow \forall C \exists^{+} y \, [\forall (z \in C) \ \neg Q(x, y \oplus z)]$.
Now, we only have to assure that the appeared predicates $\exists z \in C \ Q(x, y \oplus z)$ and $\forall z \in C \ \neg Q(x, y \oplus z)$ are computable in polynomial time (Note that the above expressions are equivalent to $\bigvee_{z \in C} \neg R(x, y, z)$ and $\bigwedge_{z \in C} \neg R(x, y, z)$ we met in Swapping Lemma.): Recall that in Swapping Lemma's formulation we demanded $|C| \leq p(n)$ and that for each $v \in C : \ |v| = p(n)$. This means that we seek if a string of polynomial length *exists*, or if the predicate holds *for all* such strings in a set with polynomial cardinality, procedure which can be surely done in polynomial time.

$(ii \Rightarrow i)$ Conversely, assume that *there exists* a predicate R and a polynomial p, as stated is (ii). Then, for each string w of length $2p(n)$, we "divide" it in two halfs w_1, w_2, such that $w = w_1 \circ w_2$ and $|w_1| = |w_2| = p(n)$. Then, for each x with $|x| = n$, and $|y| = |z| = p(n)$:

[1] We define the XOR (eXclusive OR) operator $\oplus$ of two strings of the equal length as the bit-by-bit *mod*2 addition. That is: $0 \oplus 0 = 1 \oplus 1 = 0$, and $0 \oplus 1 = 1 \oplus 0 = 1$.

$x \in L \Rightarrow \exists^+ y \forall z\ R(x,y,z) \Rightarrow \exists^+ w(|w| = 2p(n))\ R(x,w_1,w_2)$
$x \notin L \Rightarrow \forall y \exists^+ z\ R(x,y,z) \Rightarrow \exists^+ w(|w| = 2p(n))\ \neg R(x,w_1,w_2)$

($i \Rightarrow iii$) It follows immediately from the fact that **BPP** is closed under complementation (co**BPP** = **BPP**). □

In other words, Theorem 1.2 states that:

$$\mathbf{BPP} = (\exists^+\forall/\forall\exists^+) = (\forall\exists^+/\exists^+\forall) \tag{1}$$

The above characterization of **BPP** is *decisive* in the sense that if we replace the $\exists^+$ quantifier with $\exists$ (if "+" is dropped), then we can decide whether $x \in L$ or $x \notin L$. That is, the two predicates are still complementary[2] to each other, so exactly one holds for x. Note that this doesn't hold for the $(\exists^+/\exists^+)$ characterization of **BPP**, because if we replace the $\exists^+$ quantifier with $\exists$, the two resulting predicates are not complementary, and they do not define a complexity class.

By replacing in (1) the quantifier $\exists^+$ with $\exists$ (*why is this possible?*) we can obtain immediately the following result, known as the Sipser-Gács Theorem:

Corollary 1.3. $\mathbf{BPP} \subseteq \Sigma_2^p \cap \Pi_2^p$

Theorem 1.2 can be generalized for sequences of quantifiers (denoted as $\mathbf{Q}_i$):

Corollary 1.4.

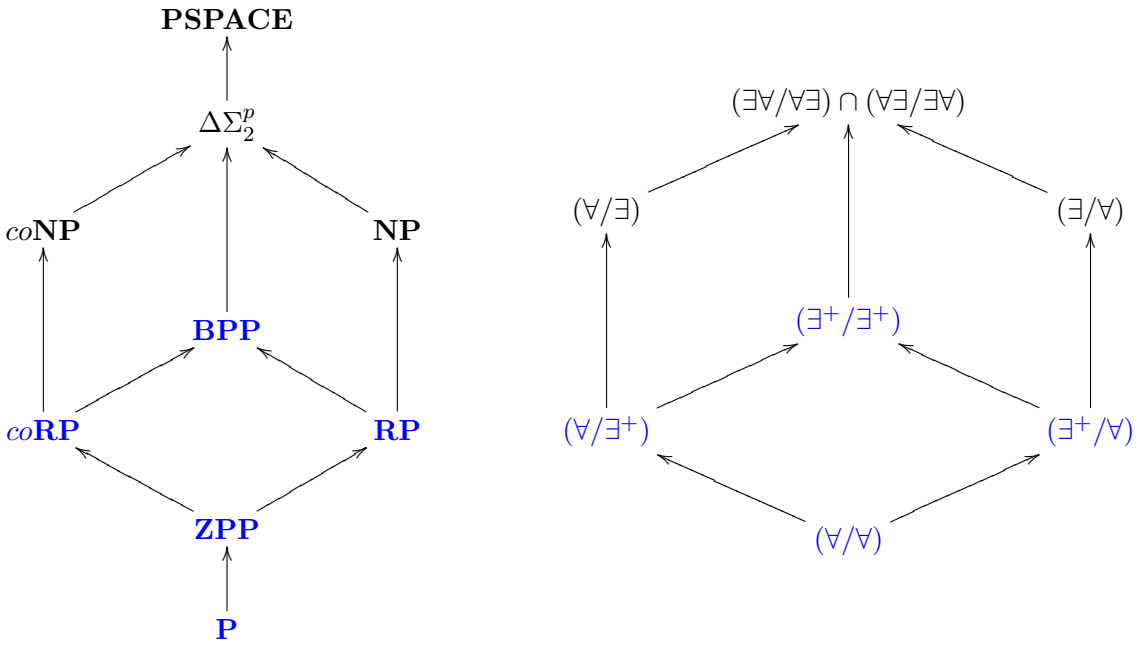
$$(\mathbf{Q}_1\exists^+\mathbf{Q}_2/\mathbf{Q}_3\exists^+\mathbf{Q}_4) = (\mathbf{Q}_1\exists^+\forall\mathbf{Q}_2/\mathbf{Q}_3\forall\exists^+\mathbf{Q}_4) = (\mathbf{Q}_1\forall\exists^+\mathbf{Q}_2/\mathbf{Q}_3\exists^+\forall\mathbf{Q}_4)$$

Using quantifier characterizations, we also have trivially many inclusions between complexity classes:

- **P** ⊆ **RP**, since $(\forall/\forall) \subseteq (\exists^+/\forall)$ (*for all* implies *for most*).
- **RP** ⊆ **BPP**, since $(\exists^+/\forall) \subseteq (\exists^+/\exists^+)$ (same reason).
- **RP** ⊆ **NP**, since $(\exists^+/\forall) \subseteq (\exists/\forall)$ (*for most* implies *for at least one*).

The main inclusions are depicted in the following Hasse diagrams ("→" denotes "⊆"):

[2] Two predicates R and P are called complementary if $R \Rightarrow \neg P$.



2 Arthur-Merlin Games

2.1 Introduction

In this section, we consider the *interaction* model between two Turing Machines as a "game". This setting is very useful to Complexity Theory, for placing upper bounds in problems' complexity, and on the other hand in Cryptography, for proving the security of cryptographic protocols against (efficient) computational attacks. The terminology used in this games is mainly anthropomorphic, known as "Arthur-Merlin" Games.

> "King Arthur recognizes the supernatural intellectual abilities of Merlin, but doesnt trust him. How should Merlin convince the intelligent but impatient King that a string x belongs to a given language L? If $L \in \mathbf{NP}$, Merlin will be able to present a witness which Arthur can check in polynomial time." From [Bab85]

In the above, Arthur is an ordinary player with the ability of making coin tosses (i.e. randomization), and Merlin is a powerful player capable of optimizing his winning chances at every move. The two players alternate moves, the history of the game is known to both, and after k moves there is a deterministic polynomial-time Turing Machine that reads the history and decides who wins. We state the formal definition:

Definition 2.1 (Arthur-Merlin Games)**.** *An Arthur-Merlin Game is a pair of interactive Turing Machines* $\boldsymbol{A}$ *and* $\boldsymbol{M}$*, and a predicate* ρ *such that:*

- *On an input x, with length $|x| = n$, exactly $q(n)$ messages of length $m(n)$ each are exchanged, where $q, m \in poly(n)$.*
- Arthur *plays first, and at iteration $1 \leq i \leq q(n)$ chooses* uniformly at random *a string r_i, where $|r_i| = m(n)$.*
- Merlin*'s reply in the i^{th} iteration, denoted y_i, is a function of all previous choices of Arthur and x. That is: $y_i = M(x, r_1, r_2, \ldots, r_i)$. In other words, M is the strategy of Merlin.*
- *For every Turing Machine $\mathbf{M}'$, a conversation between $\boldsymbol{A}$ and $\mathbf{M}'$ on input x is a string:*

  $$r_1 y_1 r_2 y_2 \cdots r_{q(n)} y_{q(n)}$$

  where for every $1 \leq i \leq q(n)$: $y_i = \mathbf{M}'(x, r_1 r_2 \cdots r_i)$
- *The predicate ρ maps x and a conversation $r_1 y_1 r_2 y_2 \cdots r_{q(n)} y_{q(n)}$ to $\{accept, reject\}$ in polynomial time, and it is called* value-of-the game *predicate.*

Now we need to determine how to test the membership for a language L using an Arthur-Merlin game: Firstly, we define the set of all conversations between Arthur and Merlin as $CONV_x^M$. Obviously, we have that $|CONV_x^M| = 2^{q(n)m(n)}$. We also define the set of accepting conversations $ACC_x^{\rho,M}$ as:

$$\left\{r_1 \cdots r_{q(n)} | \exists (y_1 \cdots y_{q(n)}) : (r_1 y_1 \cdots r_{q(n)} y_{q(n)}) \in CONV_x^M \wedge \rho(r_1 y_1 \cdots r_{q(n)} y_{q(n)}) = accept\right\}$$

Intuitively, $ACC_x^{\rho,M}$ is the set of all random choices leading Arthur to accept the input x when interacting with Merlin, and it depends only on Merlin and the pridecate ρ, given that Arthur follows the protocol. The probability that Arthur accepts x is:

$$\mathbf{Pr}[Arthur\ accepts\ x] = \frac{|ACC_x^{\rho,M}|}{|CONV_x^M|}$$

Definition 2.2. *A language L is in $\mathbf{AM}[k]$ if there exists a k-move Arthur-Merlin protocol such that for every $x \in \Sigma^*$:*

- *If $x \in L$, there exists a strategy for Merlin such that :*

  $$\mathbf{Pr}[Arthur\ accepts\ x] \geq \frac{2}{3}$$
- *If $x \notin L$, for every strategy for Merlin we have:*

  $$\mathbf{Pr}[Arthur\ accepts\ x] \leq \frac{1}{3}$$

  The first is known as completeness *condition, and the second as* soundness *condition.*

The class $\mathbf{MA}[k]$ is defined by similar way, but Merlin plays first.

2.2 Quantifier Characterizations

We denote by $\mathbf{AM} = \mathbf{AM}[2]$, and by $\mathbf{MA} = \mathbf{MA}[2]$. Following [Bab85], we consider as Merlin an **NP** machine, and as Athur a **BPP** machine. So, we can interpret Arthur-Merlin games in terms of quantifiers:

$$\mathbf{AM} = (\exists^+\exists/\exists^+\forall) = \mathcal{BP}\cdot\mathbf{NP}$$

$$\mathbf{MA} = (\exists\exists^+/\forall\exists^+) = \mathcal{N}\cdot\mathbf{BPP}$$

where $\mathcal{BP}\cdot$ and $\mathcal{N}\cdot$ is the bounded-probabilistic and the nondeterministic quantifiers respectively (see Appendix A for definitions). It is well known that we can obtain *perfect completeness* for interactive proof systems, by simulating the given protocol by another. This cannot be obtained in the soundness condition, because this would be equal to a deterministic verifier, so by definition that class collapses to **NP**. We prove perfect completeness for Arthur-Merlin games in the following theorem:

Theorem 2.1. *i.* $\mathbf{AM} = (\exists^+\exists/\exists^+\forall) = (\forall\exists/\exists^+\forall)$

ii. $\mathbf{MA} = (\exists\exists^+/\forall\exists^+) = (\exists\forall/\forall\exists^+)$

iii. In general, for even k and $\mathbf{AM}[k] = (\mathbf{Q}_1/\mathbf{Q}_2)$*:*

- $\mathbf{AM}[k+1] = (\mathbf{Q_1}\exists^+/\mathbf{Q_2}\exists^+) = (\mathbf{Q_1}\forall/\mathbf{Q_2}\exists^+)$
- $\mathbf{AM}[k+2] = (\mathbf{Q_1}\exists^+\exists/\mathbf{Q_2}\exists^+\forall) = (\mathbf{Q_1}\forall\exists/\mathbf{Q_2}\exists^+\forall)$

Proof: (i) $\mathbf{AM} = (\exists^+\exists/\exists^+\forall) = (\forall\exists^+\exists/\exists^+\forall\forall)$ (by Corollary 1.4)
$\subseteq (\forall\exists\exists/\exists^+\forall\forall) = (\forall\exists/\exists^+\forall)$ (by quantifier contraction).
The other direction is trivial: $(\forall\exists/\exists^+\forall) \subseteq (\exists^+\exists/\exists^+\forall) = \mathbf{AM}$.

(ii) $\mathbf{MA} = (\exists\exists^+/\forall\exists^+) = (\exists\exists^+\forall/\forall\forall\exists^+)$ (by Corollary 1.4)
$\subseteq (\exists\exists\forall/\forall\forall\exists^+) = (\exists\forall/\forall\exists^+)$ (by quantifier contraction).
The other direction is trivial: $(\exists\forall/\forall\exists^+) \subseteq (\exists\exists^+/\forall\exists^+) = \mathbf{MA}$.

(iii) $\mathbf{AMA} = (\exists^+\exists\exists^+/\exists^+\forall\exists^+) = (\forall\exists\exists^+/\exists^+\forall\exists^+)$ (by (ii))
$= (\forall\exists\exists^+\forall/\exists^+\forall\forall\exists^+)$ (by Corollary 1.4)
$= (\forall\exists\forall/\exists^+\forall\exists^+)$ (by quantifier contraction)
and so on for $\mathbf{AM}[k]$. □

We also prove the following useful lemma:

Lemma 2.2. $(\exists\forall/\forall\exists^+) \subseteq (\forall\exists/\exists^+\forall)$

Proof: Let $L \in (\exists\forall/\forall\exists^+)$. Then,
$x \notin L \Rightarrow \forall y\ \exists^+ z\ \neg P(x,y,z)$
$\Rightarrow \exists^+ C\ \forall y\ \exists z \in C\ \neg P(x,y,z)$ (by the Swapping Lemma 1.1i)
$\Rightarrow \exists C\ \forall y\ \exists z \in C\ \neg P(x,y,z)$

$\Rightarrow \forall y\ \exists z\ \neg P(x,y,z)$
$\Rightarrow x \notin L$
which means that all logical implications are indeed equivalences, and the second and third lines emply that $L \in (\forall\exists/\exists^{+}\forall)$. □

From the above theorem and lemma, we have the following immediate inclusions:

Corollary 2.3. $\mathbf{MA} \subseteq \mathbf{AM}$

Corollary 2.4. $\mathbf{AM} \subseteq \Pi_2^p$ *and* $\mathbf{MA} \subseteq \Sigma_2^p \cap \Pi_2^p$

Lemma 2.2 can be generalized as follows:

Corollary 2.5.

$$(\mathbf{Q}_1\exists\forall\mathbf{Q}_2/\mathbf{Q}_3\forall\exists^{+}\mathbf{Q}_4) \subseteq (\mathbf{Q}_1\forall\exists\mathbf{Q}_2/\mathbf{Q}_3\exists^{+}\forall\mathbf{Q}_4)$$

If we consider the complexity classes $\mathbf{AM}[k]$ (the languages that have Arthur-Merlin proof systems of a bounded number of rounds), they form an *hierarchy*:

$$\mathbf{AM}[0] \subseteq \mathbf{AM}[1] \subseteq \cdots \subseteq \mathbf{AM}[k] \subseteq \mathbf{AM}[k+1] \subseteq \cdots$$

Unlike the Polynomial Hierarchy, in which we believe the inclusions are *proper*, Arthur-Merlin Hierarchy collapses to the second level (which is why we usually denote as **AM** the class $\mathbf{AM}[2]$):

Theorem 2.6. *For constants* $k \geq 2$, $\mathbf{AM}[k] = \mathbf{AM}[2]$.

Proof. We show as special case the inclusion $\mathbf{MAM} \subseteq \mathbf{AM}$:
$\mathbf{MAM} = (\exists\exists^{+}\exists/\forall\exists^{+}\forall) \subseteq (\exists\exists^{+}\forall\exists/\forall\forall\exists^{+}\forall)$ (by the **BPP** Theorem 1.2)
$\subseteq (\exists\forall\exists/\forall\exists^{+}\forall)$ (by quantifier contraction)
$\subseteq (\forall\exists\exists/\exists^{+}\forall\forall)$ (by Lemma 2.2)
$\subseteq (\forall\exists/\exists^{+}\forall) = \mathbf{AM}$ (by quantifier contraction) □

We give an alternative proof of a result which provides us with strong evidence that *co***NP** $\not\subseteq$ **AM**, originally proved in [BHZ87]:

Theorem 2.7. *If co***NP** $\subseteq$ **AM**, *then:*

i. **PH** *collapses at the second level, and*

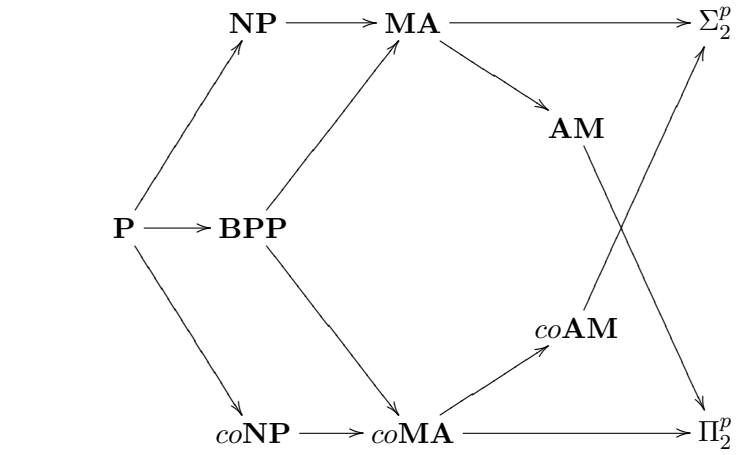
ii. $\mathbf{PH} = \mathbf{AM}$.

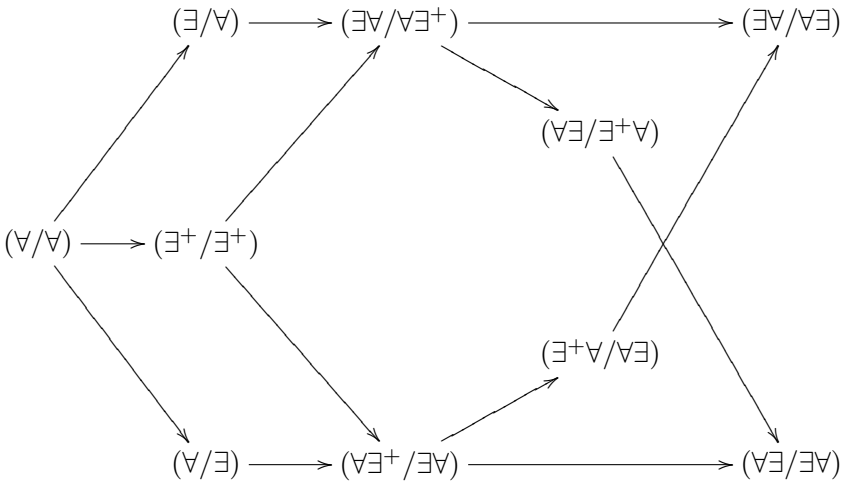
Proof: Since *co***NP** $\subseteq$ **AM**, we have that $(\forall/\exists) \subseteq (\forall\exists/\exists^{+}\forall)$ as assumption. Then:

$$\Sigma_2^p = (\exists\forall/\forall\exists) \subseteq (\exists\forall\exists/\forall\exists^{+}\forall) \subseteq (\forall\exists\exists/\exists^{+}\forall\forall) = (\forall\exists/\exists^{+}\forall) = \mathbf{AM} \subseteq (\forall\exists/\exists\forall) = \Pi_2^p$$

The first inclusion holds from our hypothesis, the second by Lemma 2.2. □

The following Hasse diagrams captures the inclusions between the most important complexity classes we've seen so far, the former in classic and the latter in quantifier notation:





Class	Definition		Notation
P	$x \in L \Rightarrow R(x)$	$x \notin L \Rightarrow \neg R(x)$	$(\forall/\forall)$
NP	$x \in L \Rightarrow \exists y\ R(x,y)$	$x \notin L \Rightarrow \forall y\ \neg R(x,y)$	$(\exists/\forall)$
co**NP**	$x \in L \Rightarrow \forall y\ R(x,y)$	$x \notin L \Rightarrow \exists y\ \neg R(x,y)$	$(\forall/\exists)$
Σ_2^p	$x \in L \Rightarrow \exists y \forall z\ R(x,y,z)$	$x \notin L \Rightarrow \forall y \exists z\ \neg R(x,y,z)$	$(\exists\forall/\forall\exists)$
Π_2^p	$x \in L \Rightarrow \forall y \exists z\ R(x,y,z)$	$x \notin L \Rightarrow \exists y \forall z\ \neg R(x,y,z)$	$(\forall\exists/\exists\forall)$
RP	$x \in L \Rightarrow \exists^+ y\ R(x,y)$	$x \notin L \Rightarrow \forall y\ \neg R(x,y)$	$(\exists^+/\forall)$
co**RP**	$x \in L \Rightarrow \forall y\ R(x,y)$	$x \notin L \Rightarrow \exists^+ y\ \neg R(x,y)$	$(\forall/\exists^+)$
BPP	$x \in L \Rightarrow \exists^+ y\ R(x,y)$	$x \notin L \Rightarrow \exists^+ y\ \neg R(x,y)$	$(\exists^+/\exists^+)$
	Alternative characterization [ZH86]:		$(\exists^+\forall/\forall\exists^+)$
	Alternative characterization [ZH86]:		$(\forall\exists^+/\exists^+\forall)$
PP	$x \in L \Rightarrow \exists_{1/2} y\ R(x,y)$	$x \notin L \Rightarrow \exists_{1/2} y\ \neg R(x,y)$	$(\exists_{1/2}/\exists_{1/2})$
AM	$x \in L \Rightarrow \exists^+ y\ R(x,y)$	$x \notin L \Rightarrow \exists^+ y\ \neg R(x,y)$	$(\exists^+\exists/\exists^+\forall)$
	Alternative characterization [ZF87]:		$(\forall\exists/\exists^+\forall)$
MA	$x \in L \Rightarrow \exists^+ y\ R(x,y)$	$x \notin L \Rightarrow \exists^+ y\ \neg R(x,y)$	$(\exists\exists^+/\forall\exists^+)$
	Alternative characterization [ZF87]:		$(\exists\forall/\forall\exists^+)$

Table 1: Quantifier Notation of the usual Complexity Classes

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 1 edition, April 2009.

[Bab85] L Babai. Trading group theory for randomness. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, STOC '85, pages 421–429, New York, NY, USA, 1985. ACM.

[BHZ87] R. B. Boppana, J. Håstad, and S. Zachos. Does co-NP have short interactive proofs? *Inf. Process. Lett.*, 25:127–132, May 1987.

[DK00] Ding-Zhu Du and Ker-I Ko. *Theory of Computational Complexity*. Wiley-Interscience, January 2000.

[FGM+89] Martin Fürer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stathis Zachos. On completeness and soundness in interactive proof systems. *Advances in Computing Research: Randomness and Computation (S. Micali, editor), JAI Press, Greenwich, CT*, 5:25–32, 1989.

[Zac86] S Zachos. Probabilistic quantifiers, adversaries, and complexity classes: an overview. In *Proc. of the conference on Structure in*

complexity theory, pages 383–400, New York, NY, USA, 1986. Springer-Verlag New York, Inc.

[Zac88] Stathis Zachos. Probabilistic quantifiers and games. *J. Comput. Syst. Sci.*, 36:433–451, June 1988.

[ZF87] Stathis Zachos and Martin Fürer. Probabilistic quantifiers vs. distrustful adversaries. In Kesav Nori, editor, *Foundations of Software Technology and Theoretical Computer Science*, volume 287 of *Lecture Notes in Computer Science*, pages 443–455. Springer Berlin / Heidelberg, 1987.

[ZH86] Stathis Zachos and Hans Heller. A decisive characterization of BPP. *Information and Control*, 69(1-3):125–135, 1986.

A Operators on Complexity Classes

Definition A.1 (Operators on Complexity Classes)**.** *Let* $\mathbf{C}$ *be an arbitrary complexity class. We define:*

1. *The* ***complement*** *operator co*$\mathbf{C}$*:*
   A language $L \in co\mathbf{C}$ *if there exists an* $L' \in \mathbf{C}$ *such that:*
   - *If* $x \in L \Rightarrow x \notin L'$
   - *If* $x \notin L \Rightarrow x \in L'$
2. *The* ***nondeterministic*** *operator* $\mathcal{N}$*:*
   A language $L \in \mathcal{N} \cdot \mathbf{C}$ *if there exists an* $L' \in \mathbf{C}$ *such that:*
   - *If* $x \in L \Rightarrow \exists y \ R_{L'}(x, y)$
   - *If* $x \notin L \Rightarrow \forall y \ \neg R_{L'}(x, y)$
3. *The* ***intersection*** *operator* Δ*:*
   A language $L \in \Delta \cdot \mathbf{C}$ *if* $L \in \mathbf{C}$ *and also* $\overline{L} \in \mathbf{C}$*, that is if* $L \in \mathbf{C} \cap co\mathbf{C}$*.*
4. *The* ***bounded-probabilistic*** *operator* $\mathcal{BP}$*:*
   A language $L \in \mathcal{BP} \cdot \mathbf{C}$ *if there exists an* $L' \in \mathbf{C}$ *such that:*
   - *If* $x \in L \Rightarrow \exists^{+} y \ R_{L'}(x, y)$
   - *If* $x \notin L \Rightarrow \exists^{+} y \ \neg R_{L'}(x, y)$
5. *The* ***probabilistic*** *operator* $\mathcal{P}$*:*
   A language $L \in \mathcal{P} \cdot \mathbf{C}$ *if there exists an* $L' \in \mathbf{C}$ *such that:*
   - *If* $x \in L \Rightarrow \exists_{1/2} y \ R_{L'}(x, y)$
   - *If* $x \notin L \Rightarrow \exists_{1/2} y \ \neg R_{L'}(x, y)$
6. *The* ***probabilistic*** *operator* $\mathcal{R}$*:*
   A language $L \in \mathcal{R} \cdot \mathbf{C}$ *if there exists an* $L' \in \mathbf{C}$ *such that:*
   - *If* $x \in L \Rightarrow \exists^{+} y \ R_{L'}(x, y)$
   - *If* $x \notin L \Rightarrow \forall y \ \neg R_{L'}(x, y)$

In the above definitions, $|y| \leq poly(|x|)$, and R_L is a polynomial-time computable predicate responding to the membership question for L. That is, $R_L(x) = 1$ iff $x \in L$ and $R_L(x, y) = 1$ iff $x; y \in L$. Note that the above operations require that $\mathbf{C}$ is closed under padding.

	$(\mathbf{Q}_1/\mathbf{Q}_2)$	**P**	**NP**	*co***NP**	Σ_i^p	Π_i^p	**PP**	**BPP**	**RP**	**ZPP**
$co\cdot$	$(\mathbf{Q}_2/\mathbf{Q}_1)$	**P**	*co***NP**	**NP**	Π_i^p	Σ_i^p	**PP**	**BPP**	*co***RP**	**ZPP**
$\mathcal{N}\cdot$	$(\exists\mathbf{Q}_1/\forall\mathbf{Q}_2)$	**NP**	**NP**	Σ_2^p	Σ_i^p	Σ_{i+1}^p		**MA**		
$\Delta\cdot$		**P**	**NP** $\cap$ *co***NP**	**NP** $\cap$ *co***NP**	$\Sigma_i^p \cap \Pi_i^p$	$\Sigma_i^p \cap \Pi_i^p$	**PP**	**BPP**	**ZPP**	**ZPP**
$\mathcal{BP}\cdot$	$(\exists^+\mathbf{Q}_1/\exists^+\mathbf{Q}_2)$	**BPP**	**AM**	*co***AM**				**BPP**		
$\mathcal{P}\cdot$	$(\exists_{1/2}\mathbf{Q}_1/\exists_{1/2}\mathbf{Q}_2)$	**PP**					**PP**			
$\mathcal{RP}\cdot$	$(\exists^+\mathbf{Q}_1/\forall\mathbf{Q}_2)$	**RP**							**RP**	