Complexity Classes

Andreas Klappenecker

Texas A&M University

Decision Problems and Formal Languages

We will now define a few **complexity classes** that are used to characterize **efficient randomized algorithms**.

The complexity classes are defined in terms of decision problems.

The set of inputs of 'yes' instances to the decision problem corresponds to a language $L \subseteq \{0, 1\}^*$.

Thus, each decision problem corresponds to a formal language.

Common Deterministic Complexity Classes

The Class **P** of Deterministic Polynomial Time DP

Definition

The class **P** consists of all languages L that do have a polynomial-time algorithm A such that

1. $x \in L$ implies $A(x)$ accepts,
2. $x \notin L$ implies $A(x)$ rejects.

Definition

The class **P** consists of all languages L that do have a polynomial-time algorithm A such that

1. $x \in L$ implies $A(x)$ accepts,
2. $x \notin L$ implies $A(x)$ rejects.

Remark

The two conditions imply

1. $x \in L$ *implies* $\Pr[A(x)\ \textit{accepts}] = 1$,
2. $x \notin L$ *implies* $\Pr[A(x)\ \textit{rejects}] = 1$.

We are going to relax these conditions for the classes **RP**, **co-RP**, *and* **BPP**.

Definition

A language L is in **NP** if and only if there exists a polynomial $p(x)$ and an polynomial-time algorithm A such that for every input $x \in \{0,1\}^*$, we have

$$x \in L \text{ if and only if } \exists w \in \{0,1\}^{p(|x|)}[A(x,w) \textbf{ accepts}]$$

A string w such that $A(x,w)$ accepts is called a **certificate**, a **proof**, or a **witness**.

We may consider w as a concise proof of the fact that $x \in L$. This proof can be verified by A.

Definition

Given a decision problem L, its **complement** is the same problem with the yes and no answers reversed.

Example

Suppose that L is the language of squares

$$L = \{0, 1, 4, 9, 16, 25, \ldots\}.$$

Then the complement of L is given by the set of non-squares

$$\overline{L} = \{2, 3, 5, 6, 7, 8, 10, \ldots\}.$$

The Class **co-NP**

Definition

$$\textbf{co-NP} = \{L \colon \overline{L} \in \textbf{NP}\}$$

Example

The language UNSAT of unsatisfiable boolean formulas is in co-NP.

The Class **co-NP**

Definition

$$\mathbf{co\text{-}NP} = \{L : \overline{L} \in \mathbf{NP}\}$$

Example

The language UNSAT of unsatisfiable boolean formulas is in co-NP.

Careful!

Need to check all possible witness strings!

The Class **co-NP**

Definition

A language L is in **co-NP** if and only if there exists a polynomial $p(x)$ and an polynomial-time algorithm A such that for every input $x \in \{0,1\}^*$, we have

$$x \in L \text{ if and only if } \forall w \in \{0,1\}^{p(|x|)} [A(x,w) \textbf{ accepts}]$$

Examples

Example (INDSET)

Given a graph $G = (V, E)$ and a positive integer k, does G have an independent set of size k?

The independent set problem (INDSET) is in **NP**.

Example ($\overline{\text{INDSET}}$)

Given a graph $G = (V, E)$ and a positive integer k, does G have no independent set of size k?

The co-independent set problem ($\overline{\text{INDSET}}$) is in **co-NP**.

The Polynomial Hierarchy **PH**

The polynomial hierarchy generalizes **NP** and **co-NP**. We have $\Sigma_0 = \mathbf{P} = \Pi_0$.

Definition

We have $\Sigma_1 = \mathbf{NP}$. A language L belongs to Σ_1 iff

$$x \in L \text{ if and only if } \exists w[A(x, w) \text{ accepts}]$$

Definition

We have $\Pi_1 = \textbf{co-NP}$. A language L belongs to Π_1 iff

$$x \in L \text{ if and only if } \forall w[A(x, w) \text{ accepts}]$$

The Polynomial Hierarchy **PH**

Definition

A language L belongs to Σ_2 iff

$$x \in L \text{ if and only if } \exists w_1 \forall w_2 [A(x, w_1, w_2) \text{ accepts}]$$

Definition

A language L belongs to Π_2 iff

$$x \in L \text{ if and only if } \forall w_1 \exists w_2 [A(x, w_1, w_2) \text{ accepts}]$$

The domain of each quantifier is $\{0, 1\}^{p(|x|)}$ for some polynomial p.

The Polynomial Hierarchy **PH**

Definition

A language L belongs to Σ_{2k} iff

$$x \in L \text{ if and only if } \exists w_1 \forall w_2 \cdots \exists w_{2k-1} \forall w_{2k} [A(x, w_1, w_2, \ldots, w_{2k-1}, w_{2k}) \text{ accepts}]$$

Definition

A language L belongs to Σ_{2k+1} iff

$$x \in L \text{ if and only if } \exists w_1 \forall w_2 \cdots \forall w_{2k} \exists w_{2k+1} [A(x, w_1, w_2, \ldots, w_{2k}, w_{2k+1}) \text{ accepts}]$$

For Σ_k, we alternate between $\exists$ and $\forall$ quantifiers, starting with an $\exists$ quantifier.

The Polynomial Hierarchy **PH**

Definition

A language L belongs to Π_{2k} iff

$$x \in L \text{ if and only if } \forall w_1 \exists w_2 \cdots \forall w_{2k-1} \exists w_{2k} [A(x, w_1, w_2, \ldots, w_{2k-1}, w_{2k}) \text{ accepts}]$$

Definition

A language L belongs to Π_{2k+1} iff

$$x \in L \text{ if and only if } \forall w_1 \exists w_2 \cdots \exists w_{2k} \forall w_{2k+1} [A(x, w_1, w_2, \ldots, w_{2k}, w_{2k+1}) \text{ accepts}]$$

For Π_k, we alternate between $\forall$ and $\exists$ quantifiers, starting with a $\forall$ quantifier.

Example (MAX-INDSET)

The language

$$\text{MAX-INDSET} = \{(G, k) \mid \text{the max. indep. set of } G \text{ is of size } k\}$$

does not seem to be contained in **NP** or **co-NP**. However, we have MAX-INDSET in Σ_2, since (G, k) is in MAX-INDSET if and only if

there **exists** *a set S of k vertices, such that*
for all *sets T containing more than k vertices,*
S is an independent set and
T is not an independent set.

Proposition

$$\Pi_k = co\Sigma_k$$

Basic Properties

Proposition

$$\Pi_k = co\Sigma_k$$

Proposition

$$\Sigma_k \subseteq \Sigma_{k+1}, \quad \Sigma_k \subseteq \Pi_{k+1}, \quad \Pi_k \subseteq \Sigma_{k+1}, \quad \Pi_k \subseteq \Pi_{k+1}.$$

Basic Properties

Proposition

$$\Pi_k = co\Sigma_k$$

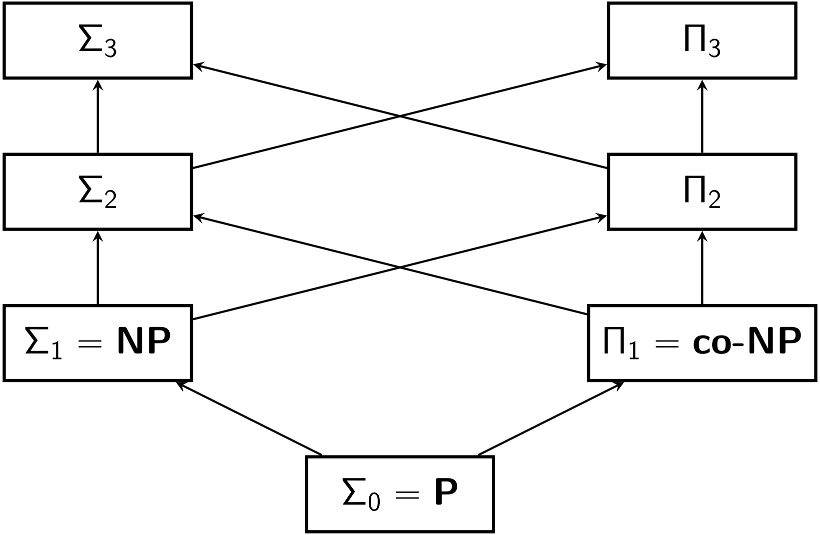
Proposition

$$\Sigma_k \subseteq \Sigma_{k+1}, \quad \Sigma_k \subseteq \Pi_{k+1}, \quad \Pi_k \subseteq \Sigma_{k+1}, \quad \Pi_k \subseteq \Pi_{k+1}.$$

Proposition

$$\Sigma_k, \Pi_k \subseteq \mathbf{PSPACE}$$

The Polynomial Hierarchy



We can also formulate the polynomial hierarchy with the help of oracles.

Definition

$$\Sigma_k = \begin{cases} \mathbf{P} & \text{if } k = 0, \\ \mathbf{NP}^{\Sigma_{k-1}} & \text{if } k > 0. \end{cases}$$

In other words, we have

$$\Sigma_0 = \mathbf{P}, \Sigma_1 = \mathbf{NP}, \Sigma_2 = \mathbf{NP}^{\mathbf{NP}}, \Sigma_3 = \mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}}}, \ldots$$

The Polynomial Hierarchy **PH**

Definition

$$\mathbf{PH} = \bigcup_{k=1}^{\infty} \Sigma_k$$

The Class **PSPACE** of Polynomial-Space Bounded DP

Definition

The class of all decision problems that can be solved using a polynomial amount of space.

Proposition

$$\mathbf{PH} \subseteq \mathbf{PSPACE}.$$

Randomized Complexity Classes

The run-time of a randomized algorithm can vary for each run, even when the input stays the same. So how do we define running time?

Definition

For a randomized algorithm A, we say that A runs in time

$$t\colon \mathbf{N} \to \mathbf{N}$$

if and only if A takes at most $t(|s|)$ steps on every input s and every sequence of random coin tosses.

The Class **RP** of Randomized Polynomial Time DP

Definition

Let ε be a constant in the range $0 \leqslant \varepsilon \leqslant 1/2$.

The class **RP** consists of all languages L that do have a polynomial-time randomized algorithm A such that

1. $x \in L$ implies $\Pr[A(x) \text{ accepts}] \geqslant 1 - \varepsilon$,
2. $x \notin L$ implies $\Pr[A(x) \text{ rejects}] = 1$.

One-Sided Error

Randomized algorithms in **RP** may err on 'yes' instances, but not on 'no' instances.

Reducing the Probability of Error

We can decrease the probability of error of our one-sided error algorithm A as follows.

Given an input x, run $A(x)$ k-times and return 'yes' if one of the k runs returned yes.

The error probability of the new algorithm is at most 2^{-k}.

Proposition

$$\mathbf{P} \subseteq \mathbf{RP}.$$

This is clear from the definitions.

Proposition

$$\mathbf{RP} \subseteq \mathbf{NP}.$$

Proof.

If L belongs to **RP**, then for x in L there exists a sequence r of coin flips such that $A(x, r)$ accepts. We can use the sequence r as a witness for $L \in \mathbf{NP}$. □

Open Problems

Problem

Does

$$\mathbf{P} = \mathbf{RP}?$$

Open Problems

Problem

Does

$$\mathbf{P} = \mathbf{RP}?$$

Problem

Is

$$\mathbf{RP} \subsetneq \mathbf{NP}?$$

The Class **co-RP** of Randomized Polynomial Time DP

Definition

Let ε be a constant in the range $0 \leqslant \varepsilon \leqslant 1/2$.
The class **co-RP** consists of all languages L whose complement $\overline{L}$ is in **RP**. In other words, L is in **co-RP** if and only if there exists a polynomial-time randomized algorithm A such that

1. $x \in L$ implies $\Pr[A(x) \text{ accepts}] = 1$,
2. $x \notin L$ implies $\Pr[A(x) \text{ rejects}] \geqslant 1 - \varepsilon$.

One-Sided Error

Randomized algorithms in **co-RP** may err on 'no' instances, but not on 'yes' instances.

By repeating a **co-RP** algorithm k times, we can reduce the error probability ε to 2^{-k} or less.

Definition

Let ε be a constant in the range $0 \leqslant \varepsilon < 1/2$.

The class **BPP** consists of all languages L such that there exists a polynomial-time randomized algorithm A such that

1. $x \in L$ implies $\Pr[A(x) \text{ accepts}] \geqslant 1 - \varepsilon$,
2. $x \notin L$ implies $\Pr[A(x) \text{ rejects}] \geqslant 1 - \varepsilon$.

Containments

Proposition

We have

$$\mathbf{RP} \subseteq \mathbf{BPP}$$

and

$$\mathbf{co\text{-}RP} \subseteq \mathbf{BPP}.$$

This follows from the definitions and error reduction.

Containments

Proposition

$$\mathbf{BPP} \subseteq \mathbf{PSPACE}.$$

Open Problem

Proposition

$$\mathbf{P} \subseteq \mathbf{BPP}.$$

Open Problem

Proposition

$$\mathbf{P} \subseteq \mathbf{BPP}.$$

Problem

Is

$$\mathbf{P} = \mathbf{BPP}\ ?$$

Relationship Between **NP** and **BPP**

Proposition (Ko)

If $\mathbf{NP} \subseteq \mathbf{BPP}$, *then* $\mathbf{NP} = \mathbf{RP}$.

Proposition

If $\mathbf{NP} \subseteq \mathbf{BPP}$, *then* $\mathbf{PH} = \mathbf{BPP}$.

BPP and the Polynomial Hierarchy

Proposition

$$\mathbf{BPP} \subseteq \Sigma_2 \cap \Pi_2.$$

Definition

The class **ZPP** consists of all languages L such that there exists a randomized algorithm A that always decides L correctly and runs in expected polynomial time.

ZPP and its relation to **RP** and **co-RP**

Proposition

$$\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{co\text{-}RP}.$$

Proof.

Suppose that $L \in \mathbf{ZPP}$. There exists Las Vegas algorithm A to decide in expected polynomial-time whether an input x belongs to L.

Algorithm for RP:
Run the algorithm A on input x for twice the expected running time steps. If A returned an answer, give that answer. Otherwise **return** 'no'.
By Markov's Inequality, the probability that it will yield an answer before we stop it is at least $1/2$. This means the probability that the algorithm will give a wrong answer on a yes instance is at most $1/2$.

Algorithm for co-RP:
The co-RP algorithm is identical, except that we return 'yes' if A does not return an answer.

Thus $\mathbf{ZPP} \subseteq \mathbf{RP} \cap \mathbf{co\text{-}RP}$.

($\mathbf{ZPP} \supseteq \mathbf{RP} \cap \mathbf{co\text{-}RP}$)

Proof. (Continued)

Suppose that the language L belongs to $\mathbf{RP} \cap \mathbf{co\text{-}RP}$.

This means that there exists

1. a polynomial-time randomized algorithm A recognizing $L \in \mathbf{RP}$, and
2. a polynomial-time randomized algorithm B recognizing $L \in \mathbf{co\text{-}RP}$.

Given an input x, do the following:

loop

1. run $A(x)$. **return** 'yes' if A returns yes.
2. run $B(x)$. **return** 'no' if $B(x)$ returns no.

end

If an answer is given, then it is correct.

Proof. (Continued)

Let T denote the worst-case runtime of $A(x); B(x)$, that is, T denotes the worst-case runtime of one iteration of the loop. Then $T = p(|x|)$ for some polynomial p.

The expected running time of the loop is bounded from above by

$$\sum_{k=0}^{\infty} T \frac{1}{2^k} \leqslant 2T.$$

Therefore, the expected running time of the loop is polynomial in the size of the input x. Thus, $\mathbf{RP} \cap \mathbf{co\text{-}RP} \subseteq \mathbf{ZPP}$.

Overview