Introduction to Theoretical Computer Science

Lecture 13: The Polynomial Hierarchy, Alternation and PSPACE

Dr. Liam O'Connor

University of Edinburgh
Semester 1, 2023/2024

The Class **CoNP**

trick **Question**

Is the class **NP** closed under complement?

The Class **CoNP**

trick Question

Is the class **NP** closed under complement?

We don't know.

The Class **CoNP**

trick Question

Is the class **NP** closed under complement?

We don't know.

Question

Why can't we just flip the answer, like we do for **P**?

The Class **CoNP**

trick Question

Is the class **NP** closed under complement?

We don't know.

Question

Why can't we just flip the answer, like we do for **P**?

Nondeterministic machines accept if just one path accepts. To flip the answer of an NRM, we'd have to accept an answer if all paths reject. This is no longer an (angelic) NRM.

The Class **CoNP**

trick Question

Is the class **NP** closed under complement?

We don't know.

Question

Why can't we just flip the answer, like we do for **P**?

Nondeterministic machines accept if just one path accepts. To flip the answer of an NRM, we'd have to accept an answer if all paths reject. This is no longer an (angelic) NRM.

Question

What is **NP** $\cap$ **CoNP**?

The Class **CoNP**

trick Question

Is the class **NP** closed under complement?

We don't know.

Question

Why can't we just flip the answer, like we do for **P**?

Nondeterministic machines accept if just one path accepts. To flip the answer of an NRM, we'd have to accept an answer if all paths reject. This is no longer an (angelic) NRM.

Question

What is **NP** $\cap$ **CoNP**? is it **P**?

The Class **CoNP**

trick Question

Is the class **NP** closed under complement?

We don't know.

Question

Why can't we just flip the answer, like we do for **P**?

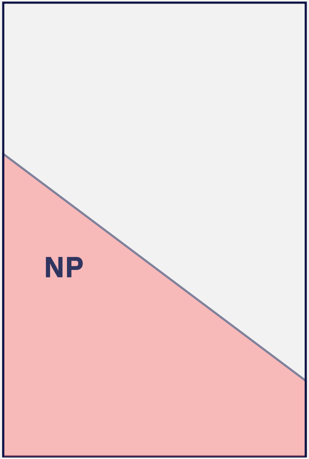
Nondeterministic machines accept if just one path accepts. To flip the answer of an NRM, we'd have to accept an answer if all paths reject. This is no longer an (angelic) NRM.

Question

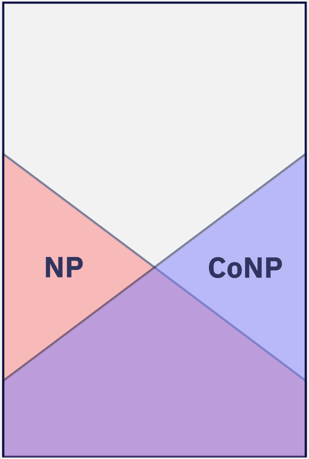
What is **NP** $\cap$ **CoNP**? is it **P**? (we don't know)

What we have now

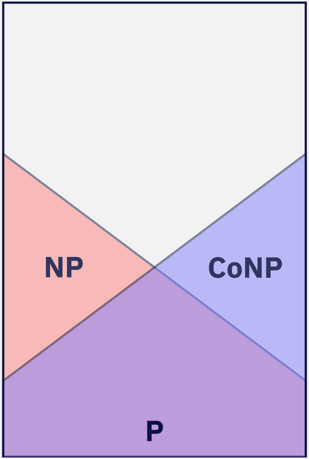
What we have now



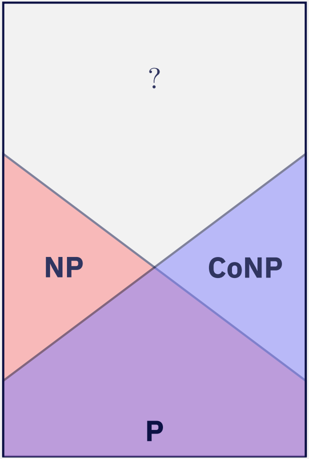
What we have now



What we have now



What we have now



Sigmas

We shall introduce notation to describe polynomial problems.

Sigma

The set $\Sigma_1^{\mathbf{P}}$ describes all problems that can be phrased as $\{y \mid \exists^{\mathbf{P}} x \in \mathbb{N}.\ R(x, y)\}$, where R is a **P**-decidable predicate and $\exists^{\mathbf{P}} x \ldots$ indicates that x is of size polynomial in the size of y.

Sigmas

We shall introduce notation to describe polynomial problems.

Sigma

The set $\Sigma_1^{\mathbf{P}}$ describes all problems that can be phrased as $\{y \mid \exists^{\mathbf{P}} x \in \mathbb{N}.\ R(x, y)\}$, where R is a **P**-decidable predicate and $\exists^{\mathbf{P}} x \ldots$ indicates that x is of size polynomial in the size of y.

- If a problem $Q \in \Sigma_1^{\mathbf{P}}$ then Q is in **NP**. **Why**?

Sigmas

We shall introduce notation to describe polynomial problems.

Sigma

The set $\Sigma_1^{\mathbf{P}}$ describes all problems that can be phrased as $\{y \mid \exists^{\mathbf{P}} x \in \mathbb{N}.\ R(x, y)\}$, where R is a **P**-decidable predicate and $\exists^{\mathbf{P}} x \ldots$ indicates that x is of size polynomial in the size of y.

- If a problem $Q \in \Sigma_1^{\mathbf{P}}$ then Q is in **NP**. **Why**?
  (we can "guess" an x and polynomially test $R(x, y)$)

Sigmas

We shall introduce notation to describe polynomial problems.

Sigma

The set $\Sigma_1^{\mathbf{P}}$ describes all problems that can be phrased as $\{y \mid \exists^{\mathbf{P}} x \in \mathbb{N}.\ R(x, y)\}$, where R is a **P**-decidable predicate and $\exists^{\mathbf{P}} x \ldots$ indicates that x is of size polynomial in the size of y.

- If a problem $Q \in \Sigma_1^{\mathbf{P}}$ then Q is in **NP**. **Why**?
  (we can “guess” an x and polynomially test $R(x, y)$)
- If a problem Q is in **NP** then $P \in \Sigma_1^{\mathbf{P}}$. **Why?**

Sigmas

We shall introduce notation to describe polynomial problems.

Sigma

The set $\Sigma_1^{\mathbf{P}}$ describes all problems that can be phrased as $\{y \mid \exists^{\mathbf{P}} x \in \mathbb{N}.\ R(x,y)\}$, where R is a **P**-decidable predicate and $\exists^{\mathbf{P}} x \ldots$ indicates that x is of size polynomial in the size of y.

- If a problem $Q \in \Sigma_1^{\mathbf{P}}$ then Q is in **NP**. **Why**?
  (we can "guess" an x and polynomially test $R(x,y)$)
- If a problem Q is in **NP** then $P \in \Sigma_1^{\mathbf{P}}$. **Why?**

Certificates

We can say that x is a *certificate* showing which "guesses" can made by our NRM giving an accepting run.

So, **NP** $= \Sigma_1^{\mathbf{P}}$.

Pis

Pi

The set $\Pi_1^{\mathbf{P}}$ describes all problems that can be phrased as $\{y \mid \forall^{\mathbf{P}} x \in \mathbb{N}.\ R(x, y)\}$, where R is a **P**-decidable predicate and $\forall^{\mathbf{P}} x \ldots$ indicates that x is of size polynomial in the size of y.

Pis

Pi

The set $\Pi_1^{\mathbf{P}}$ describes all problems that can be phrased as $\{y \mid \forall^{\mathbf{P}} x \in \mathbb{N}.\ R(x, y)\}$, where R is a **P**-decidable predicate and $\forall^{\mathbf{P}} x \ldots$ indicates that x is of size polynomial in the size of y.

$$\begin{aligned}
\overline{\Sigma_1^{\mathbf{P}}} &= \overline{\{x \mid \exists^{\mathbf{P}} y.\ R(x, y)\}} \\
&= \{x \mid \neg\exists^{\mathbf{P}} y.\ R(x, y)\} \\
&= \{x \mid \forall^{\mathbf{P}} y.\ \neg R(x, y)\} \\
&= \Pi_1^{\mathbf{P}}
\end{aligned}$$

Pis

Pi

The set $\Pi_1^{\mathbf{P}}$ describes all problems that can be phrased as $\{y \mid \forall^{\mathbf{P}} x \in \mathbb{N}.\ R(x, y)\}$, where R is a **P**-decidable predicate and $\forall^{\mathbf{P}} x \ldots$ indicates that x is of size polynomial in the size of y.

$$\begin{aligned}
\overline{\Sigma_1^{\mathbf{P}}} &= \overline{\{x \mid \exists^{\mathbf{P}} y.\ R(x, y)\}} \\
&= \{x \mid \neg\exists^{\mathbf{P}} y.\ R(x, y)\} \\
&= \{x \mid \forall^{\mathbf{P}} y.\ \neg R(x, y)\} \\
&= \Pi_1^{\mathbf{P}}
\end{aligned}$$

As $\Sigma_1^{\mathbf{P}}$ is **NP**, $\Pi_1^{\mathbf{P}}$ is **CoNP**.

Deltas

For reasons that are unknown to me, some books have:

Delta

The set Δ_1^{P} describes the intersection of Σ_1^{P} and Π_1^{P}.

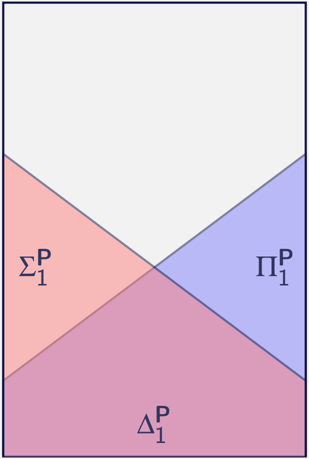
While others have

Delta

The set Δ_1^{P} describes the set P

From our characterisations of Σ_1^{P} and Π_1^{P}, we have that $\Delta_1^{\mathsf{P}} \supseteq \mathbf{P}$, but we don't know if these definitions are equal.

Relabeling



Moving Higher

Definitions

- Σ_2^{P} is all problems of form $\{x \mid \exists^{\mathsf{P}} y . \forall^{\mathsf{P}} z.\ R(x, y, z)\}$.
- Π_2^{P} is all problems of form $\{x \mid \forall^{\mathsf{P}} y . \exists^{\mathsf{P}} z.\ R(x, y, z)\}$.
- $\Delta_2^{\mathsf{P}} = \Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$

Moving Higher

Definitions

- Σ_2^{P} is all problems of form $\{x \mid \exists^{\mathsf{P}} y. \forall^{\mathsf{P}} z.\ R(x, y, z)\}$.
- Π_2^{P} is all problems of form $\{x \mid \forall^{\mathsf{P}} y. \exists^{\mathsf{P}} z.\ R(x, y, z)\}$.
- $\Delta_2^{\mathsf{P}} = \Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$

Note that $\Sigma_1^{\mathsf{P}}, \Pi_1^{\mathsf{P}}, \Delta_1^{\mathsf{P}}$ are all $\subseteq \Delta_2^{\mathsf{P}}$ (and therefore $\subseteq \Sigma_2^{\mathsf{P}}$ and $\subseteq \Pi_2^{\mathsf{P}}$). **Why?**

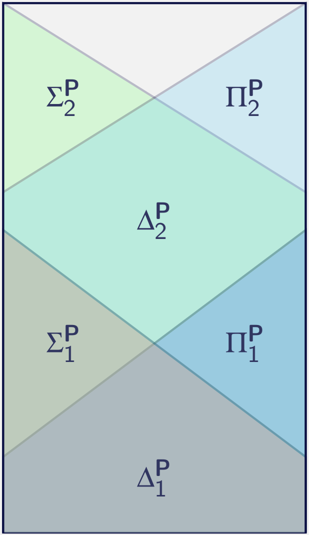
Moving Higher

Definitions

- Σ_2^{P} is all problems of form $\{x \mid \exists^{\mathsf{P}} y.\forall^{\mathsf{P}} z.\ R(x,y,z)\}$.
- Π_2^{P} is all problems of form $\{x \mid \forall^{\mathsf{P}} y.\exists^{\mathsf{P}} z.\ R(x,y,z)\}$.
- $\Delta_2^{\mathsf{P}} = \Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$

Note that $\Sigma_1^{\mathsf{P}}, \Pi_1^{\mathsf{P}}, \Delta_1^{\mathsf{P}}$ are all $\subseteq \Delta_2^{\mathsf{P}}$ (and therefore $\subseteq \Sigma_2^{\mathsf{P}}$ and $\subseteq \Pi_2^{\mathsf{P}}$). **Why?**
(our R can simply "ignore" one of the parameters)

The Polynomial Hierarchy



An equivalent characterisation

We can define in terms of oracles:

- Δ_2^P is all problems that are decidable in polynomial time by some deterministic TM/RM with an $\mathcal{O}(1)$ oracle for some complete problem in Σ_1^P, i.e. it is **P** with an $\mathcal{O}(1)$ oracle for **NP**.
- Σ_2^P allows the TM/RM to be nondeterministic, i.e. it is **NP** with an $\mathcal{O}(1)$ oracle for **NP**.
- Π_2^P is **CoNP** with an oracle for **NP**.

Building up

In general, for any $n > 1$:

- $\Delta_n^{\mathbf{P}}$ is all problems that are decidable by some deterministic, polynomially bounded TM/RM with an $\mathcal{O}(1)$ oracle for some problem $\in \Sigma_{n-1}^{\mathbf{P}}$.

Building up

In general, for any $n > 1$:

- Δ_n^{P} is all problems that are decidable by some deterministic, polynomially bounded TM/RM with an $\mathcal{O}(1)$ oracle for some problem $\in \Sigma_{n-1}^{\mathsf{P}}$.
- Σ_n^{P} are all problems that are decidable by some nondeterministic, polynomially bounded TM/RM with an $\mathcal{O}(1)$ oracle for some problem $\in \Sigma_{n-1}^{\mathsf{P}}$.

Building up

In general, for any $n > 1$:

- Δ_n^{P} is all problems that are decidable by some deterministic, polynomially bounded TM/RM with an $\mathcal{O}(1)$ oracle for some problem $\in \Sigma_{n-1}^{\mathsf{P}}$.
- Σ_n^{P} are all problems that are decidable by some nondeterministic, polynomially bounded TM/RM with an $\mathcal{O}(1)$ oracle for some problem $\in \Sigma_{n-1}^{\mathsf{P}}$.
- Π_n^{P} are all problems decidable by some *co-nondeterministic*, polynomially bounded TM/RM with an $\mathcal{O}(1)$ oracle for some problem $\in \Sigma_{n-1}^{\mathsf{P}}$.

Co-nondeterminism

Could also be called *demonic* nondeterminism. Like our normal (angelic) nondeterminism but only accepts if *all* paths accept.

Alternation

Alternation

Equivalently $\Sigma_n^{\mathbf{P}}$ are all problems that can be phrased as some alternation of (**P**-bounded) quantifiers, starting with $\exists^{\mathbf{P}}$:

$$\{w \mid \exists^{\mathbf{P}} x_1.\forall^{\mathbf{P}} x_2.\exists^{\mathbf{P}} x_3.\forall^{\mathbf{P}} x_4.\ldots x_n.\ R(w, x_1, \ldots, x_n)\}$$

$\Pi_n^{\mathbf{P}}$ starts instead with $\forall^{\mathbf{P}}$:

$$\{w \mid \forall^{\mathbf{P}} x_1.\exists^{\mathbf{P}} x_2.\forall^{\mathbf{P}} x_3.\exists^{\mathbf{P}} x_4.\ldots x_n.\ R(w, x_1, \ldots, x_n)\}$$

Alternating Machines

Alternating Machines combine the acceptance modes of both angelic and demonic nondeterministic machines.

Alternating Machines

Alternating Machines combine the acceptance modes of both angelic and demonic nondeterministic machines.

Alternating Register Machines

Consider NRMs where instead of just a MAYBE instruction we have a $\text{MAYBE}^{\forall}$ instruction and a $\text{MAYBE}^{\exists}$ instruction.

- $\text{MAYBE}^{\exists}$ is a nondeterministic choice where we accept if one branch accepts.
- $\text{MAYBE}^{\forall}$ is a nondeterministic choice where we accept only when both branches accept.

Alternating Machines

Alternating Machines combine the acceptance modes of both angelic and demonic nondeterministic machines.

Alternating Register Machines

Consider NRMs where instead of just a MAYBE instruction we have a $\mathsf{MAYBE}^{\forall}$ instruction and a $\mathsf{MAYBE}^{\exists}$ instruction.

- $\mathsf{MAYBE}^{\exists}$ is a nondeterministic choice where we accept if one branch accepts.
- $\mathsf{MAYBE}^{\forall}$ is a nondeterministic choice where we accept only when both branches accept.

Alternating Turing Machines are defined by labelling states with either $\forall$ or $\exists$.

Alternating Machines and the Polynomial Hierarchy

- The class $\Sigma_n^{\mathbf{P}}$ could equivalently be defined as the class of problems decided in polynomial time by an alternating machine that initially uses $\exists$-nondeterminism, and every path in the machine swaps quantifiers (i.e. to $\forall$ or back to $\exists$) at most $n - 1$ times.

Alternating Machines and the Polynomial Hierarchy

- The class $\Sigma_n^{\mathbf{P}}$ could equivalently be defined as the class of problems decided in polynomial time by an alternating machine that initially uses $\exists$-nondeterminism, and every path in the machine swaps quantifiers (i.e. to $\forall$ or back to $\exists$) at most $n-1$ times.
- $\Pi_n^{\mathbf{P}}$ is the same, except that we start with $\forall$ instead.

Alternating Machines and the Polynomial Hierarchy

- The class $\Sigma_n^{\mathbf{P}}$ could equivalently be defined as the class of problems decided in polynomial time by an alternating machine that initially uses $\exists$-nondeterminism, and every path in the machine swaps quantifiers (i.e. to $\forall$ or back to $\exists$) at most $n - 1$ times.
- $\Pi_n^{\mathbf{P}}$ is the same, except that we start with $\forall$ instead.

The class **AP**

AP is the class of all problems decidable by an alternating machine in polynomial time, without any restriction on swapping quantifiers.

Alternating Machines and the Polynomial Hierarchy

- The class $\Sigma_n^{\mathbf{P}}$ could equivalently be defined as the class of problems decided in polynomial time by an alternating machine that initially uses $\exists$-nondeterminism, and every path in the machine swaps quantifiers (i.e. to $\forall$ or back to $\exists$) at most $n-1$ times.
- $\Pi_n^{\mathbf{P}}$ is the same, except that we start with $\forall$ instead.

The class **AP**

AP is the class of all problems decidable by an alternating machine in polynomial time, without any restriction on swapping quantifiers.
AP is known to be equal to **PSPACE** (more on this in a moment)

A Fragile House of Cards

Warning

The polynomial hierarchy could collapse at any point.
(i.e. all of the classes in the PH could be equal)

A Fragile House of Cards

Warning

The polynomial hierarchy could collapse at any point.
(i.e. all of the classes in the PH could be equal)

We don't know that $\mathbf{P} \neq \mathbf{PSPACE}$, and the entire polynomial hierarchy is contained inside **AP** which $=$ **PSPACE**.

A Fragile House of Cards

Warning

The polynomial hierarchy could collapse at any point.
(i.e. all of the classes in the PH could be equal)

We don't know that **P** $\neq$ **PSPACE**, and the entire polynomial hierarchy is contained inside **AP** which $=$ **PSPACE**.

Wait, what's **PSPACE**?

An RM/TM is $f(n)$*-space-bounded* if it may use only $f(inputsize)$ space. For TMs, space means cells on tape; for RMs, number of bits in registers.

A Fragile House of Cards

Warning

The polynomial hierarchy could collapse at any point.
(i.e. all of the classes in the PH could be equal)

We don't know that **P** $\neq$ **PSPACE**, and the entire polynomial hierarchy is contained inside **AP** which $=$ **PSPACE**.

Wait, what's **PSPACE**?

An RM/TM is $f(n)$*-space-bounded* if it may use only $f(inputsize)$ space. For TMs, space means cells on tape; for RMs, number of bits in registers.
PSPACE is the class of problems solvable by polynomially-space-bounded machines.

A Fragile House of Cards

Warning

The polynomial hierarchy could collapse at any point.
(i.e. all of the classes in the PH could be equal)

We don't know that **P** $\neq$ **PSPACE**, and the entire polynomial hierarchy is contained inside **AP** which $=$ **PSPACE**.

Wait, what's PSPACE?

An RM/TM is $f(n)$*-space-bounded* if it may use only $f(inputsize)$ space. For TMs, space means cells on tape; for RMs, number of bits in registers.
PSPACE is the class of problems solvable by polynomially-space-bounded machines.

The following are obvious (**Exercise**: why?):
PSPACE $\supseteq$ **P** ?

A Fragile House of Cards

Warning

The polynomial hierarchy could collapse at any point.
(i.e. all of the classes in the PH could be equal)

We don't know that **P** $\neq$ **PSPACE**, and the entire polynomial hierarchy is contained inside **AP** which $=$ **PSPACE**.

Wait, what's **PSPACE**?

An RM/TM is $f(n)$-*space-bounded* if it may use only $f(inputsize)$ space. For TMs, space means cells on tape; for RMs, number of bits in registers.
PSPACE is the class of problems solvable by polynomially-space-bounded machines.

The following are obvious (**Exercise**: why?):
PSPACE $\supseteq$ **P** ? **PSPACE** $\supseteq$ **NP** ?

A Fragile House of Cards

Warning

The polynomial hierarchy could collapse at any point.
(i.e. all of the classes in the PH could be equal)

We don't know that **P** $\neq$ **PSPACE**, and the entire polynomial hierarchy is contained inside **AP** which $=$ **PSPACE**.

Wait, what's **PSPACE**?

An RM/TM is $f(n)$*-space-bounded* if it may use only $f(inputsize)$ space. For TMs, space means cells on tape; for RMs, number of bits in registers.
PSPACE is the class of problems solvable by polynomially-space-bounded machines.

The following are obvious (**Exercise**: why?):
PSPACE $\supseteq$ **P** ? **PSPACE** $\supseteq$ **NP** ? **PSPACE** $\subseteq$ **EXPTIME**?

Conclusions

This concludes our study of complexity theory. Next week, we start on a new (or rather, very old) model of computation that is the foundation for modern studies of programming languages and their semantics, the λ-calculus.