Overview

Introduction

Modelling parallel systems

Linear Time Properties

Regular Properties

Linear Temporal Logic (LTL)

- syntax and semantics of LTL
- automata-based LTL model checking ⟵
- complexity of LTL model checking

Computation-Tree Logic

Equivalences and Abstraction

LTL model checking problem

LTL model checking problem

given: finite transition system $\mathcal{T}$ over *AP*
(without terminal states)
LTL-formula φ over *AP*

question: does $\mathcal{T} \models \varphi$ hold **?**

LTL model checking problem

given: finite transition system $\mathcal{T}$ over *AP*
(without terminal states)
LTL-formula φ over *AP*

question: does $\mathcal{T} \models \varphi$ hold ?

basic idea: try to refute $\mathcal{T} \models \varphi$

LTL model checking problem

given: finite transition system $\mathcal{T}$ over AP
(without terminal states)
LTL-formula φ over AP

question: does $\mathcal{T} \models \varphi$ hold **?**

basic idea: try to refute $\mathcal{T} \models \varphi$ by searching for a path π in $\mathcal{T}$ s.t.

$$\pi \not\models \varphi$$

LTL model checking problem

given: finite transition system $\mathcal{T}$ over AP
(without terminal states)
LTL-formula φ over AP

question: does $\mathcal{T} \models \varphi$ hold **?**

basic idea: try to refute $\mathcal{T} \models \varphi$ by searching for a path π in $\mathcal{T}$ s.t.

$$\pi \not\models \varphi, \text{ i.e., } \pi \models \neg\varphi$$

The LTL model checking problem

given: finite transition system $\mathcal{T}$ over *AP*
LTL-formula φ over *AP*

question: does $\mathcal{T} \models \varphi$ hold ?

1. construct an **NBA** $\mathcal{A}$ for $Words(\neg\varphi)$

The LTL model checking problem

given: finite transition system $\mathcal{T}$ over AP
LTL-formula φ over AP

question: does $\mathcal{T} \models \varphi$ hold ?

1. construct an **NBA** $\mathcal{A}$ for $Words(\neg\varphi)$
2. search a path π in $\mathcal{T}$ with

$$trace(\pi) \in Words(\neg\varphi)$$

The LTL model checking problem

given: finite transition system $\mathcal{T}$ over AP
LTL-formula φ over AP

question: does $\mathcal{T} \models \varphi$ hold **?**

1. construct an **NBA** $\mathcal{A}$ for $Words(\neg\varphi)$
2. search a path π in $\mathcal{T}$ with

$$trace(\pi) \in Words(\neg\varphi) = \mathcal{L}_\omega(\mathcal{A})$$

The LTL model checking problem

given: finite transition system $\mathcal{T}$ over AP
LTL-formula φ over AP

question: does $\mathcal{T} \models \varphi$ hold ?

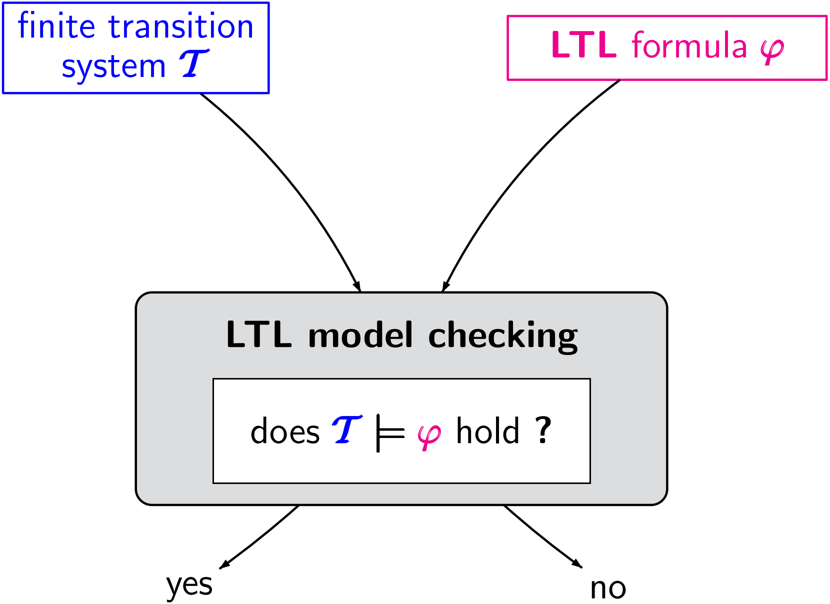
1. construct an **NBA** $\mathcal{A}$ for $Words(\neg\varphi)$
2. search a path π in $\mathcal{T}$ with

$$trace(\pi) \in Words(\neg\varphi) = \mathcal{L}_\omega(\mathcal{A})$$

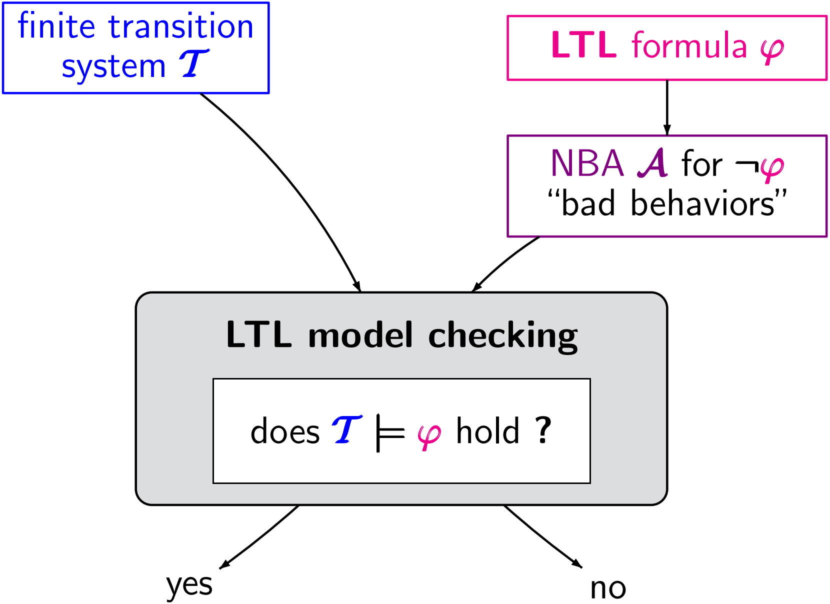
↑

construct the product-TS $\mathcal{T} \otimes \mathcal{A}$
search a path in the product that meets
the acceptance condition of $\mathcal{A}$

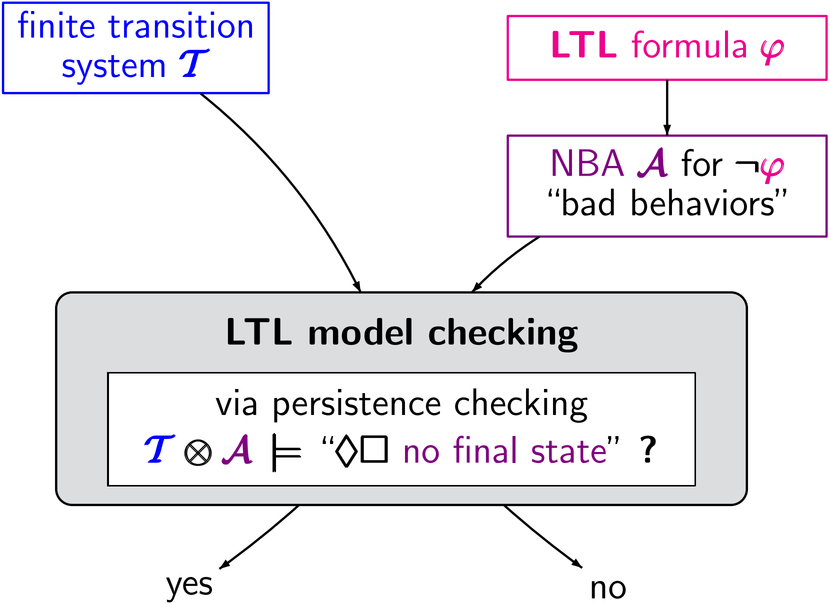
Automata-based LTL model checking



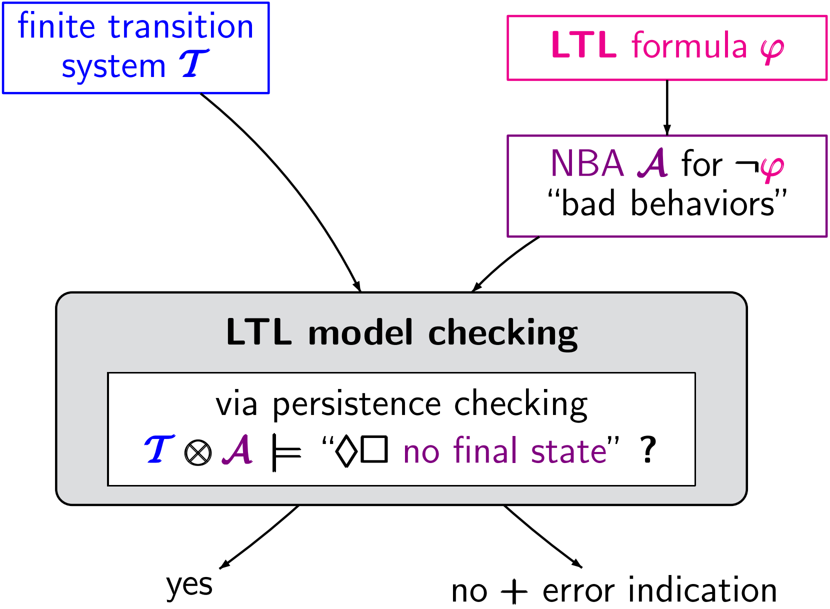
Automata-based LTL model checking



Automata-based LTL model checking



Automata-based LTL model checking



Safety and LTL model checking

Safety and LTL model checking

safety property E	LTL-formula φ

Safety and LTL model checking

safety property E	LTL-formula φ
NFA for the bad prefixes for E $\mathcal{L}(\mathcal{A}) \subseteq (2^{AP})^+$	

Safety and LTL model checking

safety property E	LTL-formula φ
NFA for the bad prefixes for E $\mathcal{L}(\mathcal{A}) \subseteq (2^{AP})^+$	**NBA** for the "bad behaviors" $\mathcal{L}_\omega(\mathcal{A}) = Words(\neg\varphi)$

Safety and LTL model checking

safety property E	LTL-formula φ
NFA for the bad prefixes for E $\mathcal{L}(\mathcal{A}) \subseteq (2^{AP})^+$	**NBA** for the "bad behaviors" $\mathcal{L}_\omega(\mathcal{A}) = Words(\neg\varphi)$
$Traces_{fin}(\mathcal{T}) \cap \mathcal{L}(\mathcal{A}) = \varnothing$	

Safety and LTL model checking

safety property E	LTL-formula φ
NFA for the bad prefixes for E $\mathcal{L}(\mathcal{A}) \subseteq (2^{AP})^+$	**NBA** for the "bad behaviors" $\mathcal{L}_\omega(\mathcal{A}) = Words(\neg\varphi)$
$Traces_{fin}(\mathcal{T}) \cap \mathcal{L}(\mathcal{A}) = \varnothing$	$Traces(\mathcal{T}) \cap \mathcal{L}_\omega(\mathcal{A}) = \varnothing$

Safety and LTL model checking

safety property E	LTL-formula φ
NFA for the bad prefixes for E $\mathcal{L}(\mathcal{A}) \subseteq (2^{AP})^+$	**NBA** for the "bad behaviors" $\mathcal{L}_\omega(\mathcal{A}) = Words(\neg\varphi)$
$Traces_{fin}(\mathcal{T}) \cap \mathcal{L}(\mathcal{A}) = \varnothing$	$Traces(\mathcal{T}) \cap \mathcal{L}_\omega(\mathcal{A}) = \varnothing$
invariant checking in the product $\mathcal{T} \otimes \mathcal{A} \models \Box\neg F$?	

Safety and LTL model checking

safety property E	LTL-formula φ
NFA for the bad prefixes for E $\mathcal{L}(\mathcal{A}) \subseteq (2^{AP})^+$	**NBA** for the "bad behaviors" $\mathcal{L}_\omega(\mathcal{A}) = Words(\neg\varphi)$
$Traces_{fin}(\mathcal{T}) \cap \mathcal{L}(\mathcal{A}) = \varnothing$	$Traces(\mathcal{T}) \cap \mathcal{L}_\omega(\mathcal{A}) = \varnothing$
invariant checking in the product $\mathcal{T} \otimes \mathcal{A} \models \Box\neg F$?	persistence checking in the product $\mathcal{T} \otimes \mathcal{A} \models \Diamond\Box\neg F$?

Safety and LTL model checking

safety property E	LTL-formula φ
NFA for the bad prefixes for E $\mathcal{L}(\mathcal{A}) \subseteq (2^{AP})^+$	**NBA** for the "bad behaviors" $\mathcal{L}_\omega(\mathcal{A}) = \mathit{Words}(\neg\varphi)$
$\mathit{Traces}_{fin}(\mathcal{T}) \cap \mathcal{L}(\mathcal{A}) = \varnothing$	$\mathit{Traces}(\mathcal{T}) \cap \mathcal{L}_\omega(\mathcal{A}) = \varnothing$
invariant checking in the product $\mathcal{T} \otimes \mathcal{A} \models \Box\neg F$?	persistence checking in the product $\mathcal{T} \otimes \mathcal{A} \models \Diamond\Box\neg F$?
error indication: $\widehat{\pi} \in \mathit{Paths}_{fin}(\mathcal{T})$ s.t. $\mathit{trace}(\widehat{\pi}) \in \mathcal{L}(\mathcal{A})$	

Safety and LTL model checking

safety property E	LTL-formula φ
NFA for the bad prefixes for E $\mathcal{L}(\mathcal{A}) \subseteq (2^{AP})^+$	**NBA** for the "bad behaviors" $\mathcal{L}_\omega(\mathcal{A}) = Words(\neg\varphi)$
$Traces_{fin}(\mathcal{T}) \cap \mathcal{L}(\mathcal{A}) = \varnothing$	$Traces(\mathcal{T}) \cap \mathcal{L}_\omega(\mathcal{A}) = \varnothing$
invariant checking in the product $\mathcal{T} \otimes \mathcal{A} \models \Box\neg F$?	persistence checking in the product $\mathcal{T} \otimes \mathcal{A} \models \Diamond\Box\neg F$?
error indication: $\widehat{\pi} \in Paths_{fin}(\mathcal{T})$ s.t. $trace(\widehat{\pi}) \in \mathcal{L}(\mathcal{A})$	error indication: prefix of a path π s.t. $trace(\pi) \in \mathcal{L}_\omega(\mathcal{A})$

Complexity of LTL model checking

main steps of automata-based LTL model checking:

construction of an NBA $\mathcal{A}$ for $\neg\varphi$

persistence checking in the product $\mathcal{T} \otimes \mathcal{A}$

Complexity of LTL model checking

main steps of automata-based LTL model checking:

construction of an NBA $\mathcal{A}$ for $\neg\varphi$	$\longleftarrow \mathcal{O}(\exp(|\varphi|))$
persistence checking in the product $\mathcal{T} \otimes \mathcal{A}$	

Complexity of LTL model checking

main steps of automata-based LTL model checking:

construction of an NBA $\mathcal{A}$ for $\neg\varphi$	$\longleftarrow \mathcal{O}(\exp(\|\varphi\|))$
persistence checking in the product $\mathcal{T} \otimes \mathcal{A}$	$\longleftarrow \mathcal{O}(\mathit{size}(\mathcal{T}) \cdot \mathit{size}(\mathcal{A}))$

Complexity of LTL model checking

main steps of automata-based LTL model checking:

construction of an NBA $\mathcal{A}$ for $\neg\varphi$	$\longleftarrow \mathcal{O}(\exp(\lvert\varphi\rvert))$
persistence checking in the product $\mathcal{T} \otimes \mathcal{A}$	$\longleftarrow \mathcal{O}(\mathit{size}(\mathcal{T}) \cdot \mathit{size}(\mathcal{A}))$

complexity: $\mathcal{O}(\mathit{size}(\mathcal{T}) \cdot \exp(\lvert\varphi\rvert))$

main steps of automata-based LTL model checking:

construction of an NBA $\mathcal{A}$ for $\neg\varphi$	$\longleftarrow$	$\mathcal{O}(\exp(\lvert\varphi\rvert))$
persistence checking in the product $\mathcal{T} \otimes \mathcal{A}$	$\longleftarrow$	$\mathcal{O}(size(\mathcal{T}) \cdot size(\mathcal{A}))$

complexity: $\mathcal{O}(size(\mathcal{T}) \cdot \exp(\lvert\varphi\rvert))$

> The **LTL** model checking problem is **PSPACE**-complete

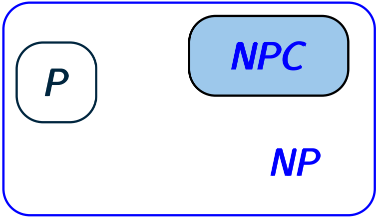
Recall: complexity classes

Complexity classes *P*, *NP*

P = class of decision problem solvable in deterministic polynomial time

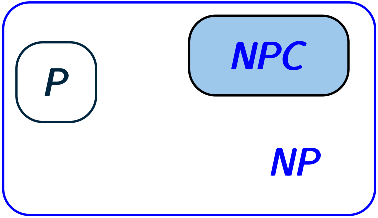
NP = class of decision problem solvable in nondeterministic polynomial time

Complexity classes *P*, *NP*



NPC = class of ***NP***-complete problems

Complexity classes P, NP

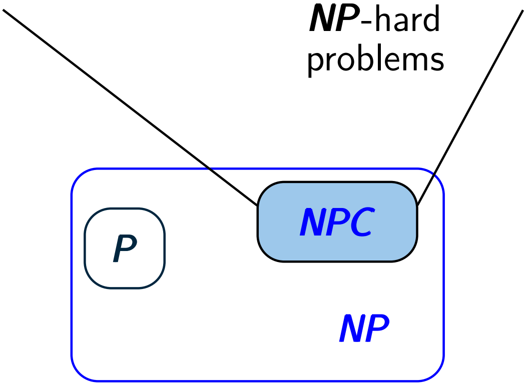


NPC = class of NP-complete problems

(1) $L \in NP$

(2) L is NP-hard, i.e., $K \leq_{poly} L$ for all $K \in NP$

Complexity classes *P*, *NP*



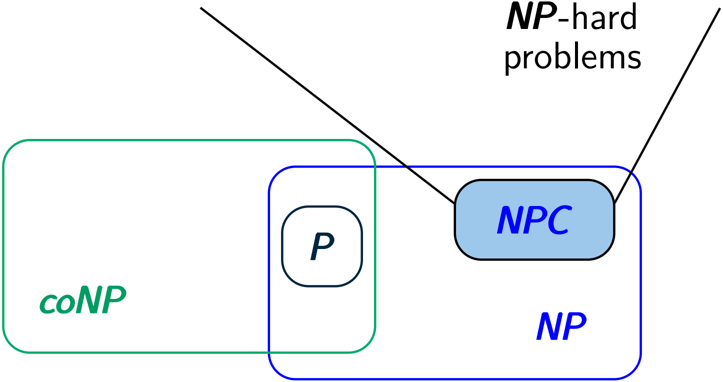
NPC = class of ***NP***-complete problems

↑

(1) $L \in NP$

(2) L is ***NP***-hard, i.e., $K \leq_{poly} L$ for all $K \in NP$

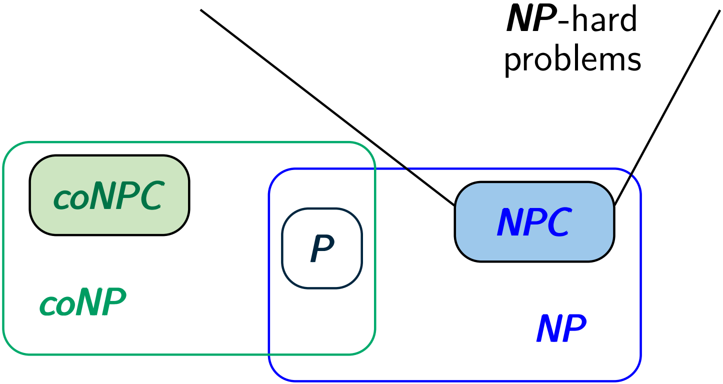
Complexity classes *P*, *NP*, *coNP*



$$coNP \;=\; \{\, \overline{L} : L \in NP \,\}$$

↑ complement of L

Complexity classes *P*, *NP*, *coNP*

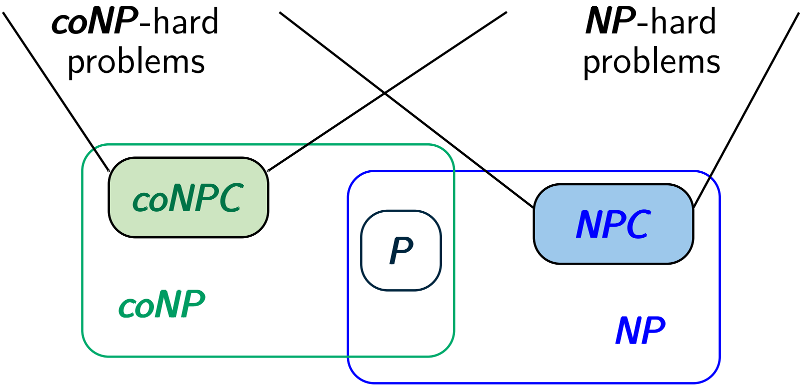


coNPC = class of ***coNP***-complete problems

(1) $L \in$ ***coNP***

(2) L is ***coNP***-hard, i.e., $K \leq_{poly} L$ for all $K \in$ ***coNP***

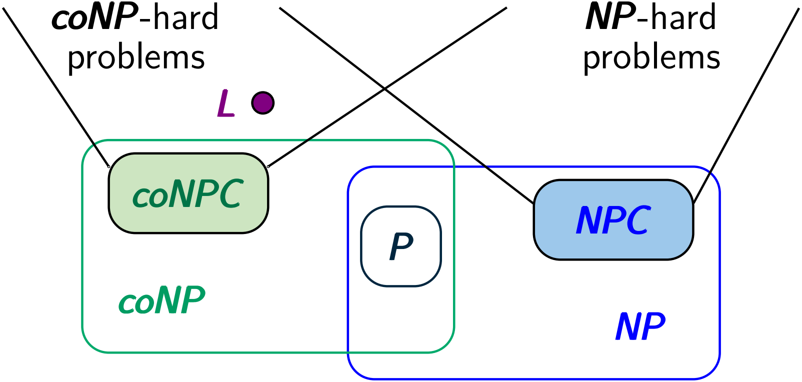
Complexity classes *P*, *NP*, *coNP*



coNPC = class of ***coNP***-complete problems

L is ***coNP***-hard iff $\overline{L}$ is ***NP***-hard

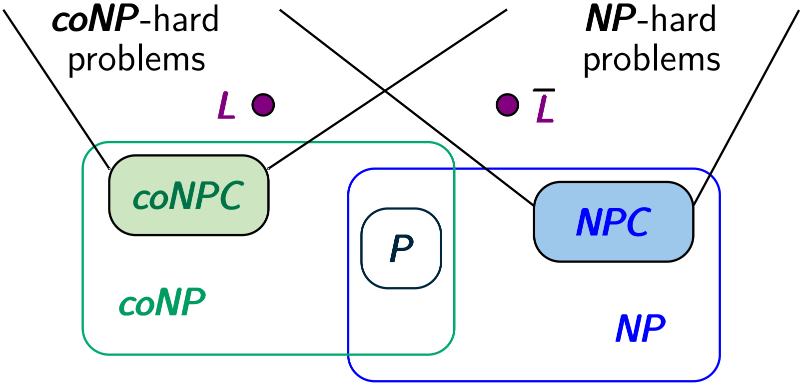
Complexity classes *P*, *NP*, *coNP*



coNPC = class of ***coNP***-complete problems

L is ***coNP***-hard iff $\overline{L}$ is ***NP***-hard

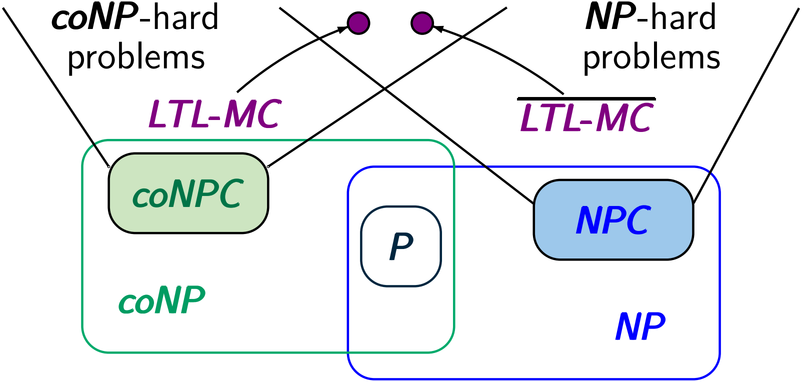
Complexity classes *P*, *NP*, *coNP*



coNPC = class of ***coNP***-complete problems

L is ***coNP***-hard iff $\overline{L}$ is ***NP***-hard

Complexity classes *P*, *NP*, *coNP*



coNPC = class of ***coNP***-complete problems

L is ***coNP***-hard iff $\overline{L}$ is ***NP***-hard

***coNP*-hardness**

The **LTL** model checking problem is ***coNP***-hard

The **LTL** model checking problem is ***coNP***-hard

proof by a polynomial reduction

$$HP \leq_{poly} \overline{LTL\text{-}MC}$$

Hamilton path problem ↗ (HP)

complement of the **LTL** model checking problem ↖ ($\overline{LTL\text{-}MC}$)

> The **LTL** model checking problem is ***coNP***-hard

proof by a polynomial reduction

$$HP \le_{poly} \overline{LTL\text{-}MC}$$

HP: Hamilton path problem

$\overline{LTL\text{-}MC}$: complement of the **LTL** model checking problem

complement of the **LTL** model checking problem:

given: finite transition system $\mathcal{T}$, LTL-formula φ

question: does $\mathcal{T} \not\models \varphi$ hold ?

> The **LTL** model checking problem is ***coNP***-hard

proof by a polynomial reduction

$$HP \leq_{poly} \overline{LTL\text{-}MC}$$

HP: Hamilton path problem (**NP**-complete)

$\overline{LTL\text{-}MC}$: complement of the **LTL** model checking problem

complement of the **LTL** model checking problem:

given: finite transition system $\mathcal{T}$, LTL-formula φ

question: does $\mathcal{T} \not\models \varphi$ hold ?

The **LTL** model checking problem is ***coNP***-hard

proof by a polynomial reduction

$$HP \leq_{poly} \overline{LTL\text{-}MC}$$

Hamilton path problem ← **NP**-complete

complement of the **LTL** model checking problem ← **NP**-hard

complement of the **LTL** model checking problem:

given: finite transition system $\mathcal{T}$, LTL-formula φ

question: does $\mathcal{T} \not\models \varphi$ hold ?

Complexity of LTL model checking

We just saw:

> The **LTL** model checking problem is ***coNP***-hard

We now prove:

> The **LTL** model checking problem is ***PSPACE***-complete

The complexity class *PSPACE*

The complexity class *PSPACE*

PSPACE = class of decision problems solvable by a deterministic polynomially space-bounded algorithm

The complexity class *PSPACE*

PSPACE = class of decision problems solvable by a deterministic polynomially space-bounded algorithm

- ***NP*** ⊆ ***PSPACE***

The complexity class *PSPACE*

> ***PSPACE*** = class of decision problems solvable by a deterministic polynomially space-bounded algorithm

- ***NP*** ⊆ ***PSPACE***

  ↑

  > **DFS**-based analysis of the computation tree of an ***NP***-algorithm

The complexity class *PSPACE*

PSPACE = class of decision problems solvable by a deterministic polynomially space-bounded algorithm

- ***NP*** $\subseteq$ ***PSPACE***

  ↑

  DFS-based analysis of the computation tree of an ***NP***-algorithm

  space requirements:

  recursion depth $\widehat{=}$ height of computation tree

The complexity class *PSPACE*

PSPACE = class of decision problems solvable by a deterministic polynomially space-bounded algorithm

- ***NP*** $\subseteq$ ***PSPACE***
- ***PSPACE*** = ***coPSPACE***

  (holds for any deterministic complexity class)

The complexity class *PSPACE*

PSPACE = class of decision problems solvable by a deterministic polynomially space-bounded algorithm

- ***NP*** ⊆ ***PSPACE***
- ***PSPACE*** = ***coPSPACE***
  (holds for any deterministic complexity class)
- ***PSPACE*** = ***NPSPACE*** (Savitch's Theorem)

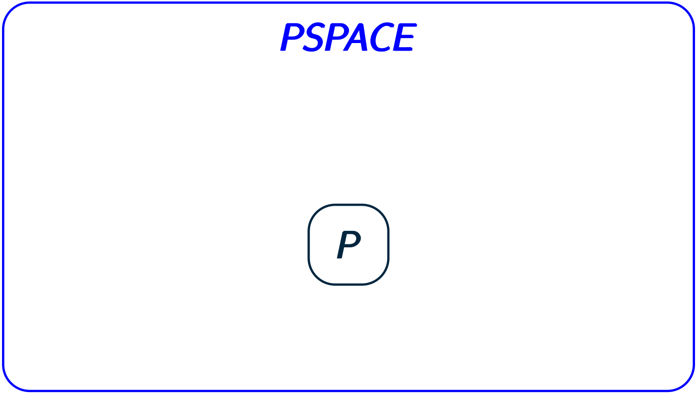
PSPACE = class of decision problems solvable by a deterministic polynomially space-bounded algorithm

- ***NP*** $\subseteq$ ***PSPACE***
- ***PSPACE*** = ***coPSPACE***
  (holds for any deterministic complexity class)
- ***PSPACE*** = ***NPSPACE*** (Savitch's Theorem)

↑

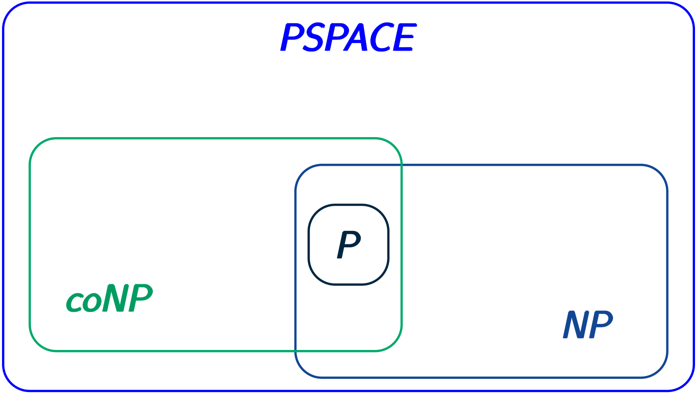
To prove $L \in$ ***PSPACE*** it suffices to provide a nondeterministic polynomially space-bounded algorithm for the complement $\overline{L}$ of L

Complexity classes *P*, *NP*, *coNP*, *PSPACE*



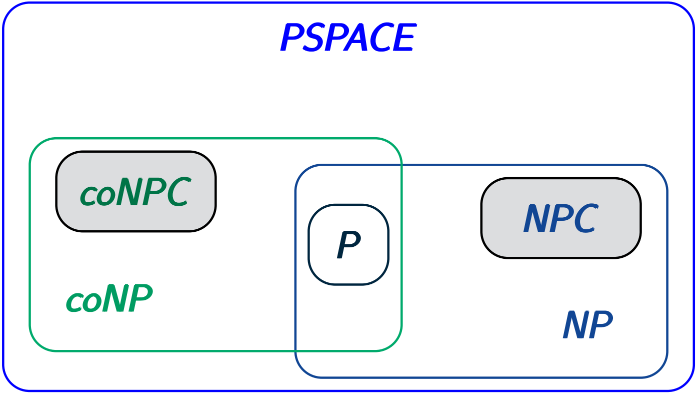
PSPACE = class of decision problems that are decidable in deterministic polynomial space

Complexity classes *P*, *NP*, *coNP*, *PSPACE*



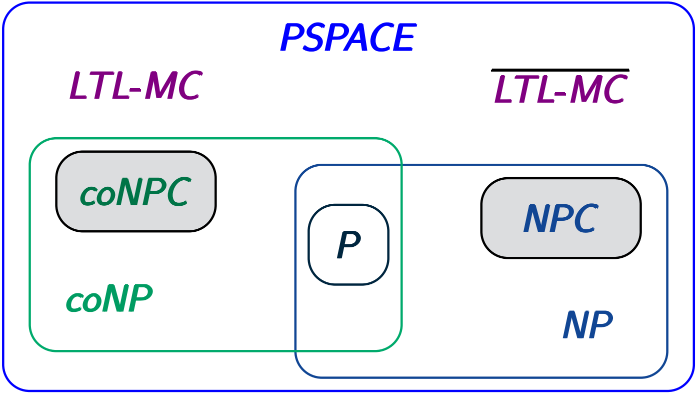
PSPACE = class of decision problems that are decidable in deterministic polynomial space

Complexity classes *P*, *NP*, *coNP*, *PSPACE*



PSPACE = class of decision problems that are decidable in deterministic polynomial space

Complexity classes *P*, *NP*, *coNP*, *PSPACE*



PSPACE = class of decision problems that are decidable in deterministic polynomial space