Rethinking Tractability for Schedulability Analysis

Kunal Agrawal
Washington University in Saint Louis

Sanjoy Baruah
Washington University in Saint Louis

Pontus Ekberg
Uppsala University

RTSS 2023

Moving the goalposts for fun and profit

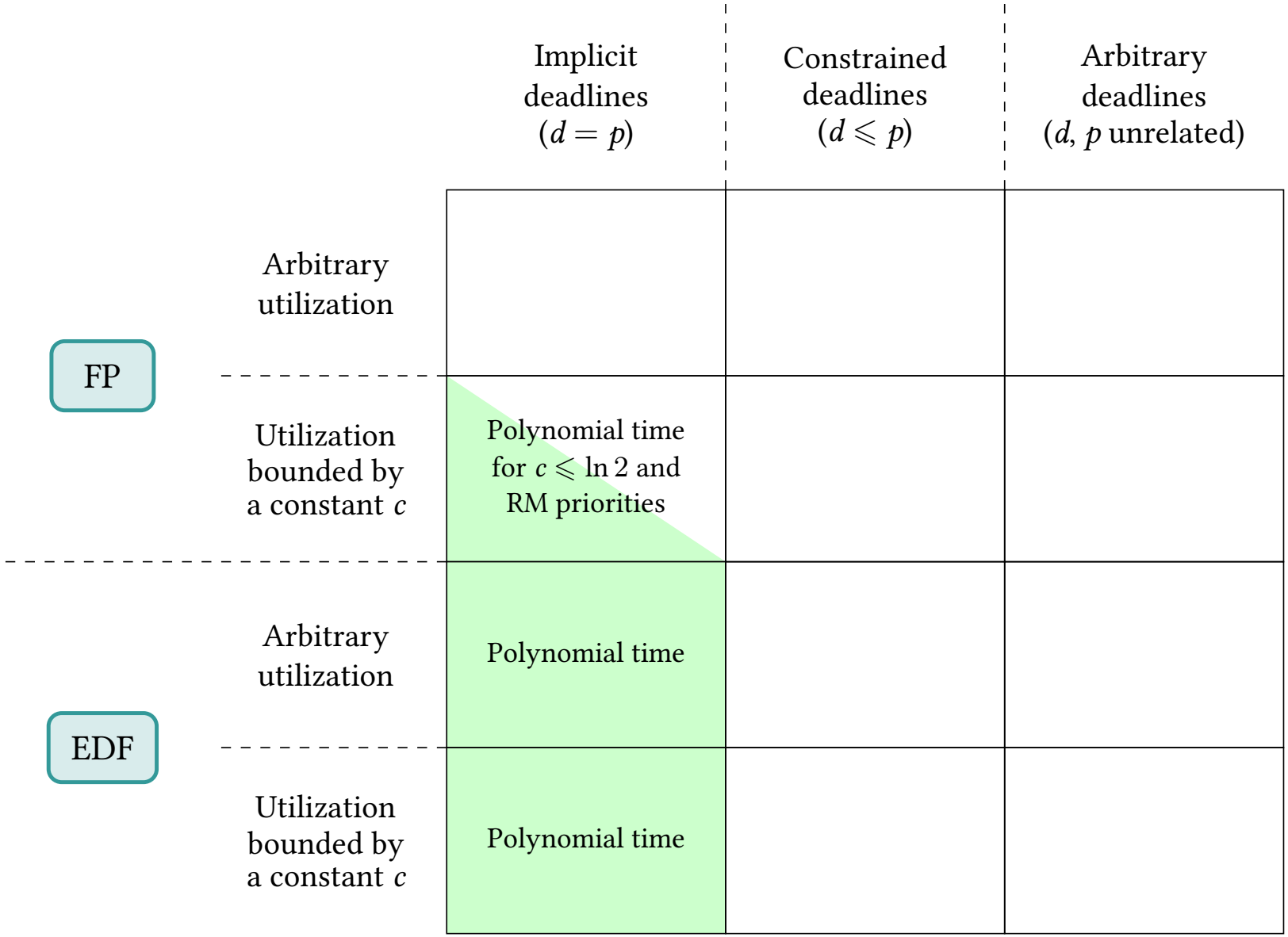
’70s and early ’80s

Polynomial-time schedulability tests (Liu and Layland’s utilization bounds for implicit-deadline EDF end FP, etc.)

Complexity of uniprocessor sporadic schedulability

		Implicit deadlines ($d = p$)	Constrained deadlines ($d \leqslant p$)	Arbitrary deadlines (d, p unrelated)
FP	Arbitrary utilization			
	Utilization bounded by a constant c	Polynomial time for $c \leqslant \ln 2$ and RM priorities		
EDF	Arbitrary utilization	Polynomial time		
	Utilization bounded by a constant c	Polynomial time		

Complexity of uniprocessor sporadic schedulability



Complexity of uniprocessor sporadic schedulability

		Implicit deadlines ($d = p$)	Constrained deadlines ($d \leqslant p$)	Arbitrary deadlines (d, p unrelated)
FP	Arbitrary utilization	Weakly NP-complete	Weakly NP-complete	Weakly NP-hard
	Utilization bounded by a constant c	Polynomial time for $c \leqslant \ln 2$ and RM priorities Else NP-complete	Weakly NP-complete for $0 < c < 1$	Weakly NP-hard for $0 < c < 1$
EDF	Arbitrary utilization	Polynomial time	Strongly coNP-complete	Strongly coNP-complete
	Utilization bounded by a constant c	Polynomial time	Weakly coNP-complete for $0 < c < 1$	Weakly coNP-complete for $0 < c < 1$

Complexity of uniprocessor sporadic schedulability

		Implicit deadlines ($d = p$)	Constrained deadlines ($d \leqslant p$)	Arbitrary deadlines (d, p unrelated)
FP	Arbitrary utilization	Weakly NP-complete	Weakly NP-complete	Weakly NP-hard
	Utilization bounded by a constant c	Polynomial time for $c \leqslant \ln 2$ and RM priorities Else NP-complete	Weakly NP-complete for $0 < c < 1$	Weakly NP-hard for $0 < c < 1$
EDF	Arbitrary utilization	Polynomial time	Strongly coNP-complete	Strongly coNP-complete
	Utilization bounded by a constant c	Polynomial time	Weakly coNP-complete for $0 < c < 1$	Weakly coNP-complete for $0 < c < 1$

Moving the goalposts for fun and profit

’70s and early ’80s

Polynomial-time schedulability tests (Liu and Layland’s utilization bounds for implicit-deadline EDF end FP, etc.)

Moving the goalposts for fun and profit

’70s and early ’80s

Polynomial-time schedulability tests (Liu and Layland’s utilization bounds for implicit-deadline EDF end FP, etc.)

Late ’80s and ’90s

Pseudo-polynomial time tests (Response-time analysis for FP, processor-demand analysis for EDF, etc.)

Complexity of uniprocessor sporadic schedulability

		Implicit deadlines ($d = p$)	Constrained deadlines ($d \leqslant p$)	Arbitrary deadlines (d, p unrelated)
FP	Arbitrary utilization	Weakly NP-complete	Weakly NP-complete	Weakly NP-hard
	Utilization bounded by a constant c	Polynomial time for $c \leqslant \ln 2$ and RM priorities Else NP-complete	Weakly NP-complete for $0 < c < 1$	Weakly NP-hard for $0 < c < 1$
EDF	Arbitrary utilization	Polynomial time	Strongly coNP-complete	Strongly coNP-complete
	Utilization bounded by a constant c	Polynomial time	Weakly coNP-complete for $0 < c < 1$	Weakly coNP-complete for $0 < c < 1$

Complexity of uniprocessor sporadic schedulability

		Implicit deadlines ($d = p$)	Constrained deadlines ($d \leqslant p$)	Arbitrary deadlines (d, p unrelated)
FP	Arbitrary utilization	Weakly NP-complete Pseudo-poly. time	Weakly NP-complete Pseudo-poly. time	Weakly NP-hard
	Utilization bounded by a constant c	Polynomial time for $c \leqslant \ln 2$ and RM priorities Else NP-complete	Weakly NP-complete for $0 < c < 1$ Pseudo-poly. time	Weakly NP-hard for $0 < c < 1$ Pseudo-poly. time
EDF	Arbitrary utilization	Polynomial time	Strongly coNP-complete	Strongly coNP-complete
	Utilization bounded by a constant c	Polynomial time	Weakly coNP-complete for $0 < c < 1$ Pseudo-poly. time	Weakly coNP-complete for $0 < c < 1$ Pseudo-poly. time

Moving the goalposts for fun and profit

'70s and early '80s

Polynomial-time schedulability tests (Liu and Layland's utilization bounds for implicit-deadline EDF end FP, etc.)

Late '80s and '90s

Pseudo-polynomial time tests (Response-time analysis for FP, processor-demand analysis for EDF, etc.)

Moving the goalposts for fun and profit

'70s and early '80s

Polynomial-time schedulability tests (Liu and Layland's utilization bounds for implicit-deadline EDF end FP, etc.)

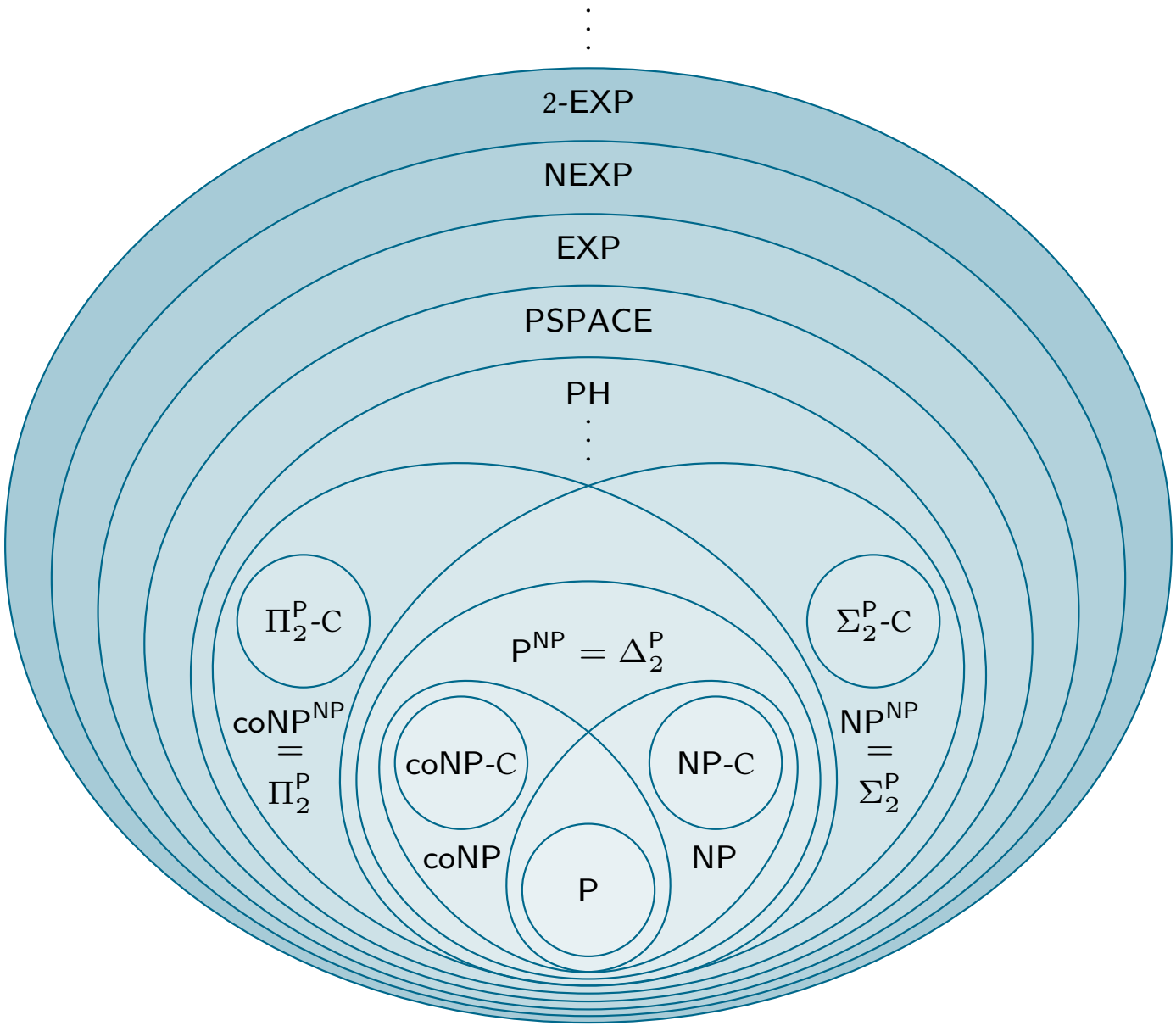
Late '80s and '90s

Pseudo-polynomial time tests (Response-time analysis for FP, processor-demand analysis for EDF, etc.)

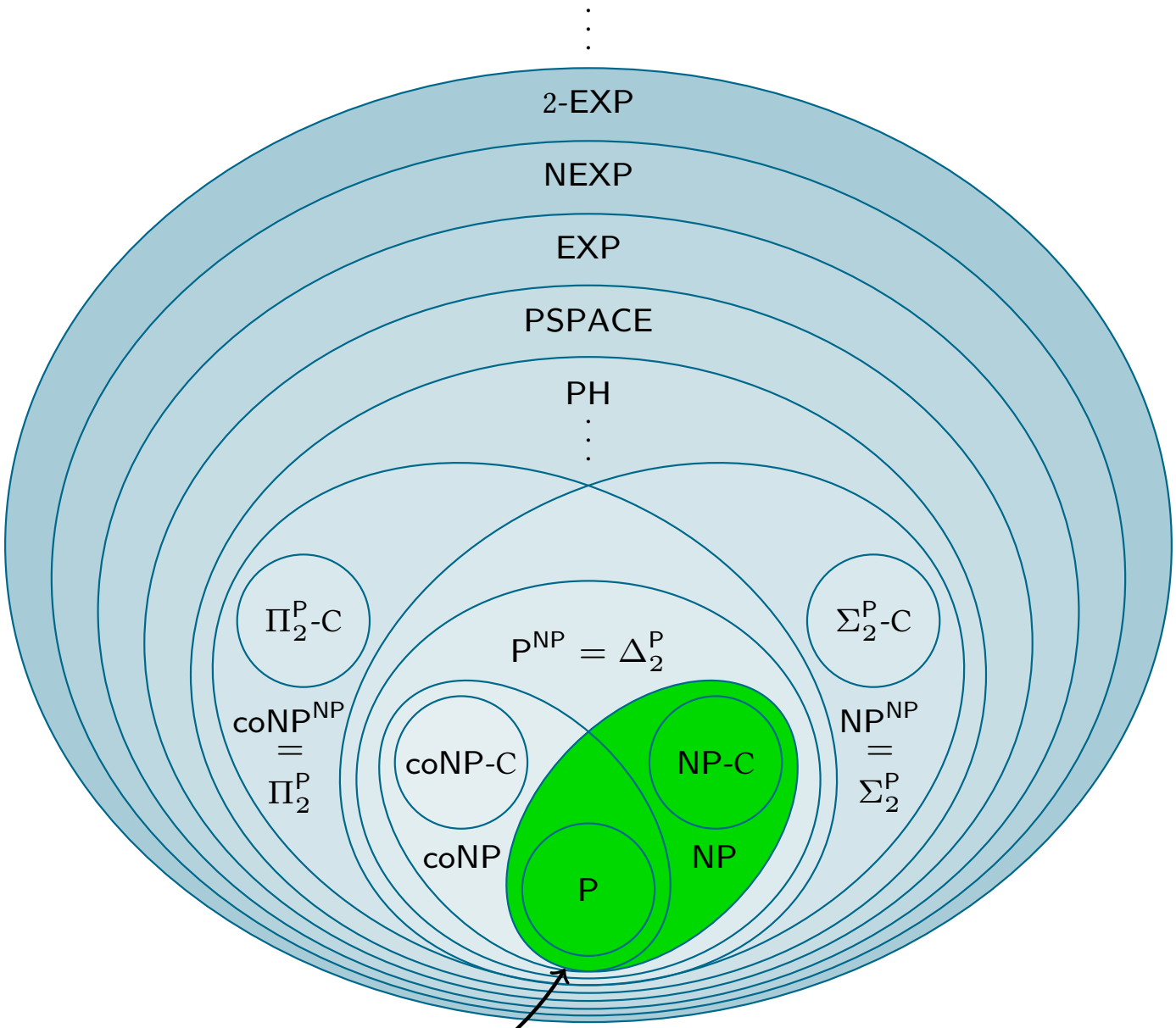
More recently

Integer Linear Programming (ILP) and similar optimized tools to implement non-pseudo-polynomial time tests.

What can be solved with ILPs?

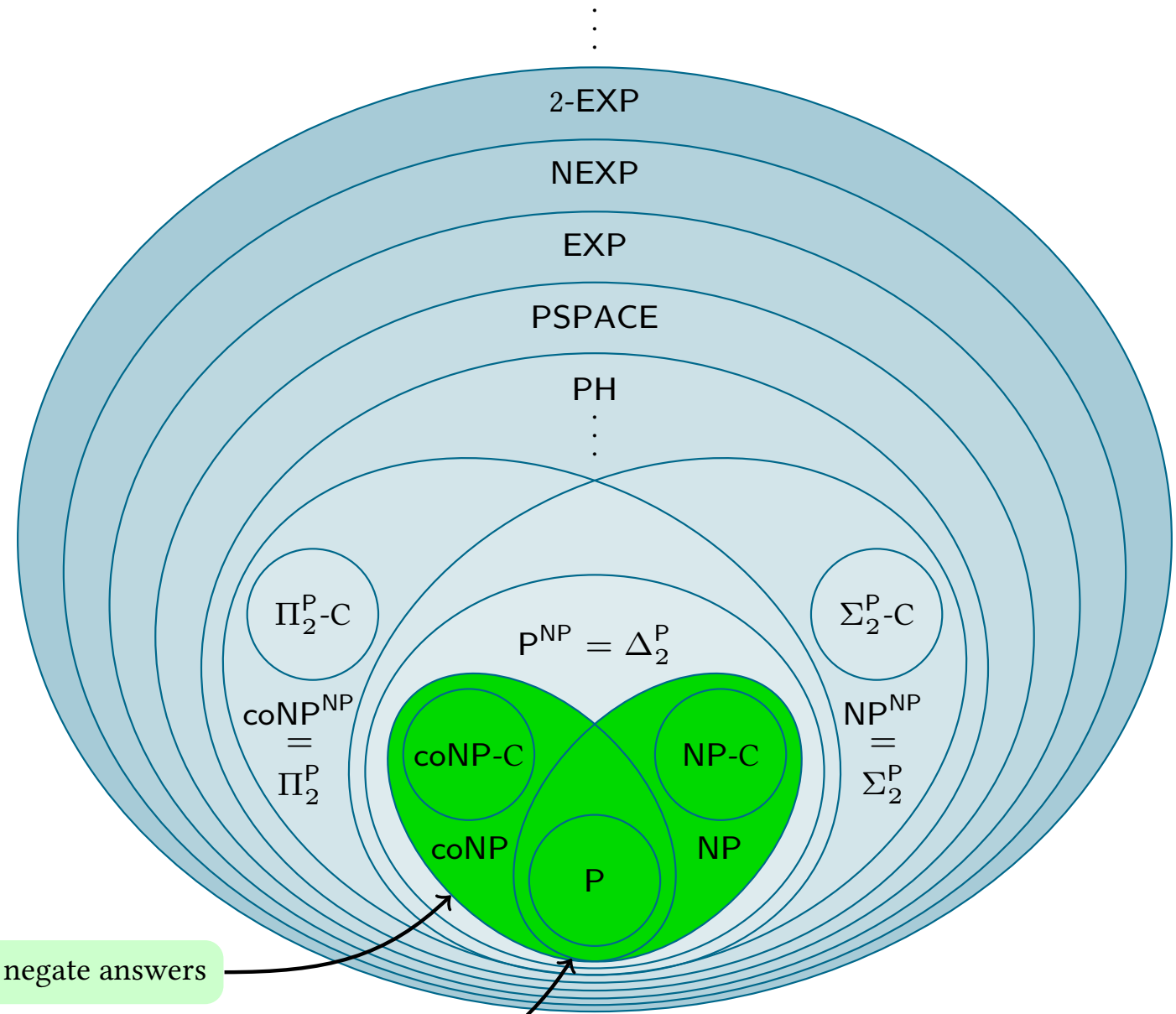


What can be solved with ILPs?



Solvable by one call to ILP

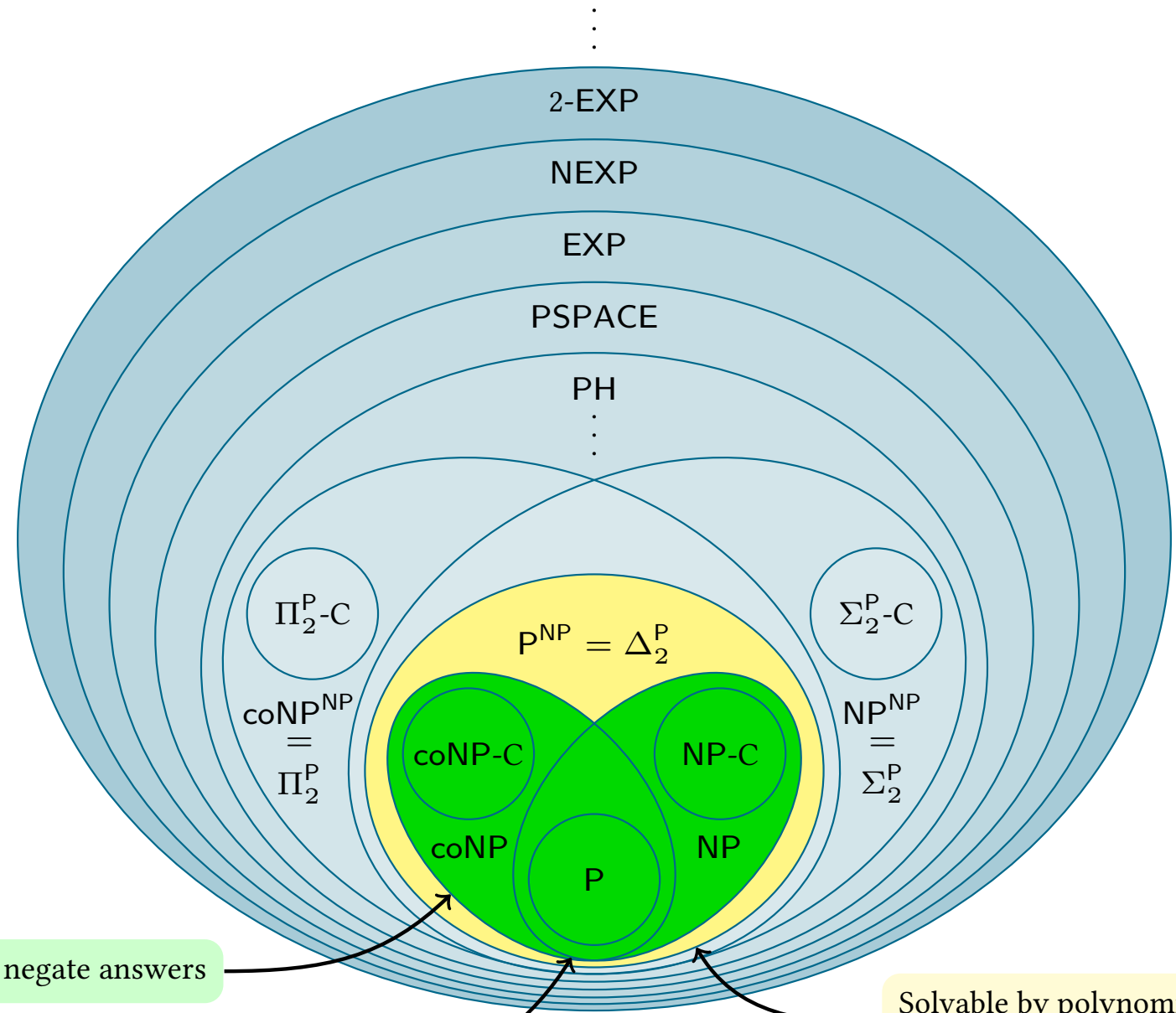
What can be solved with ILPs?



Ditto if we negate answers

Solvable by one call to ILP

What can be solved with ILPs?

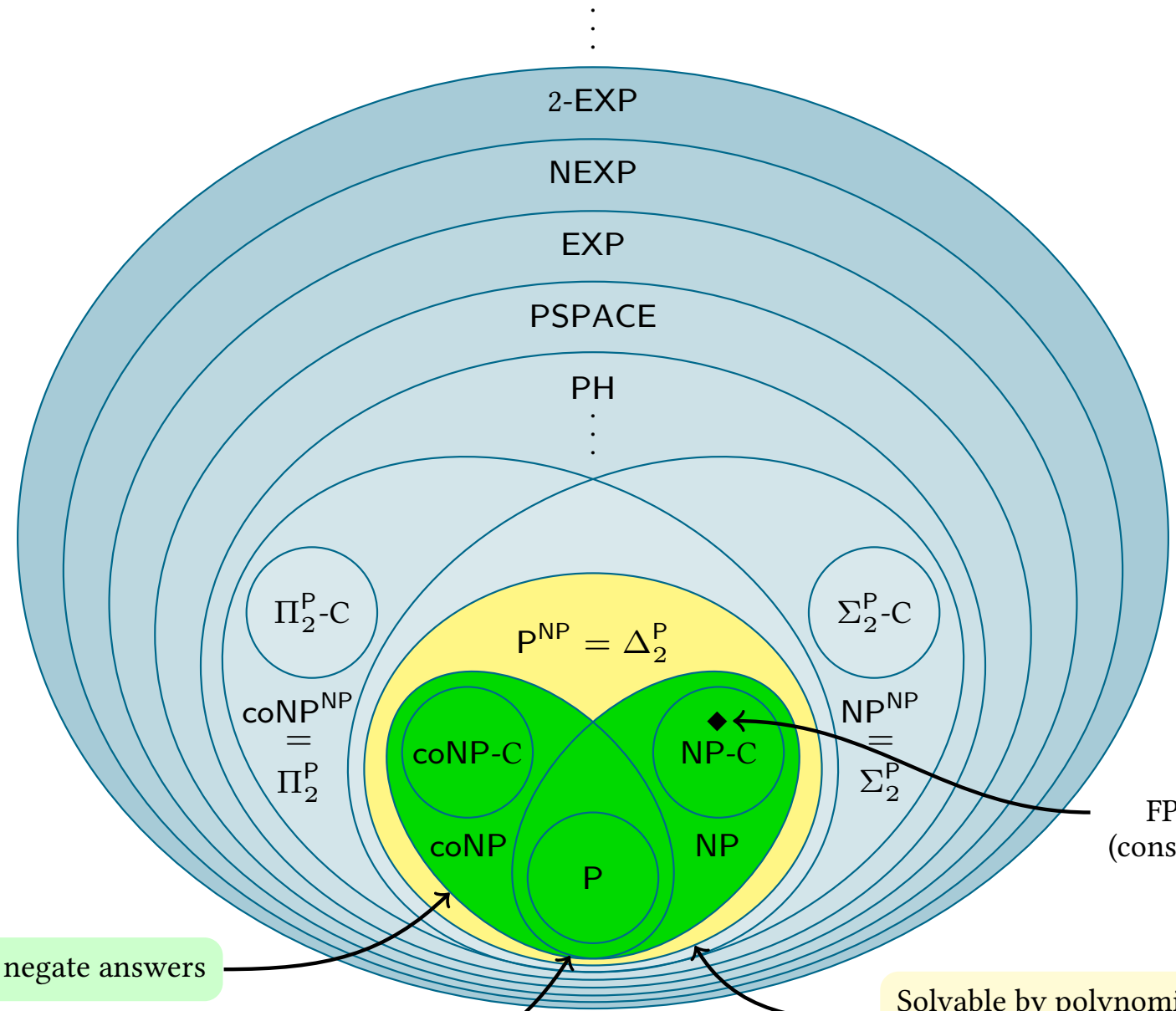


Ditto if we negate answers

Solvable by one call to ILP

Solvable by polynomial number of calls to ILP

What can be solved with ILPs?



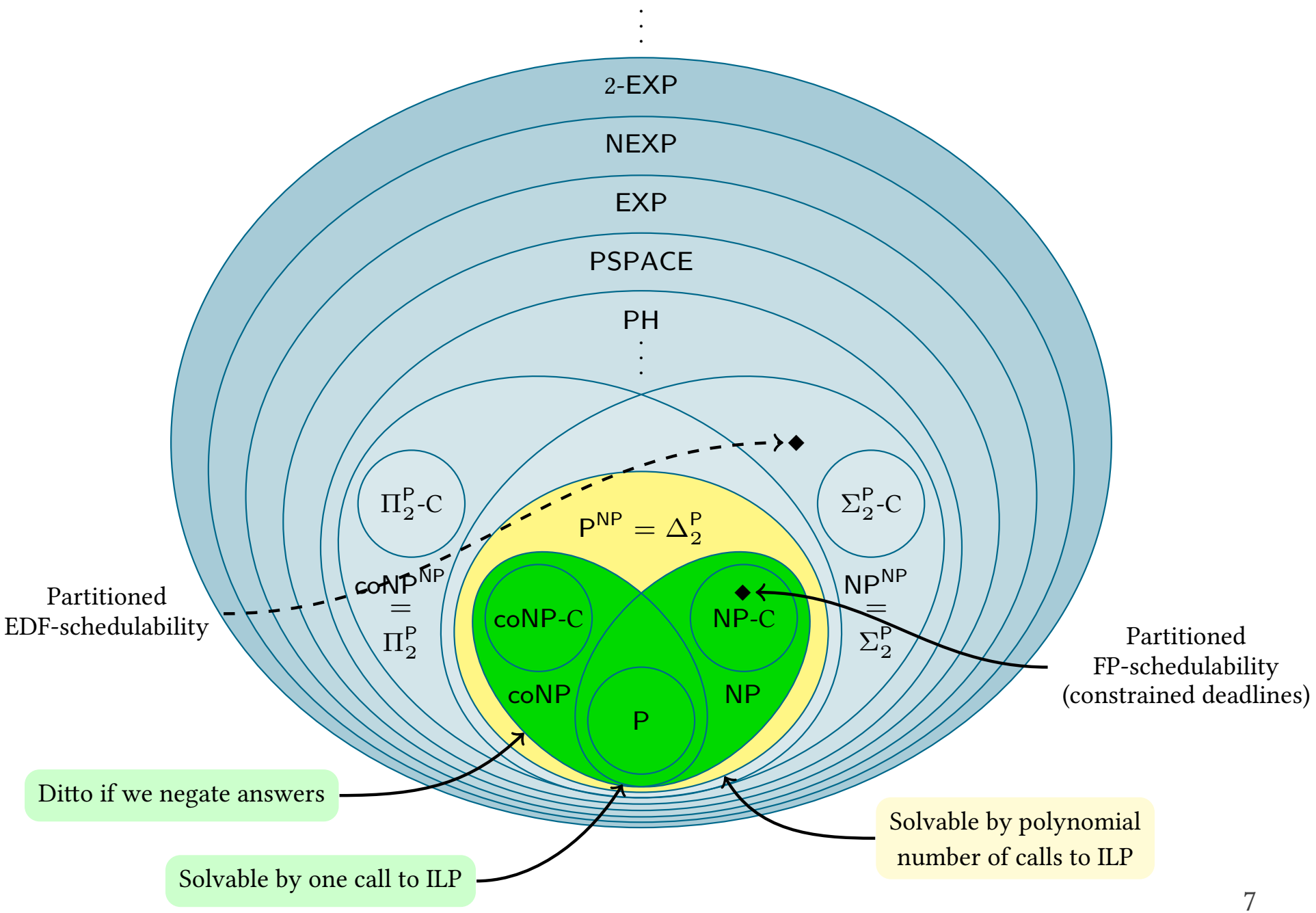
Partitioned FP-schedulability (constrained deadlines)

Ditto if we negate answers

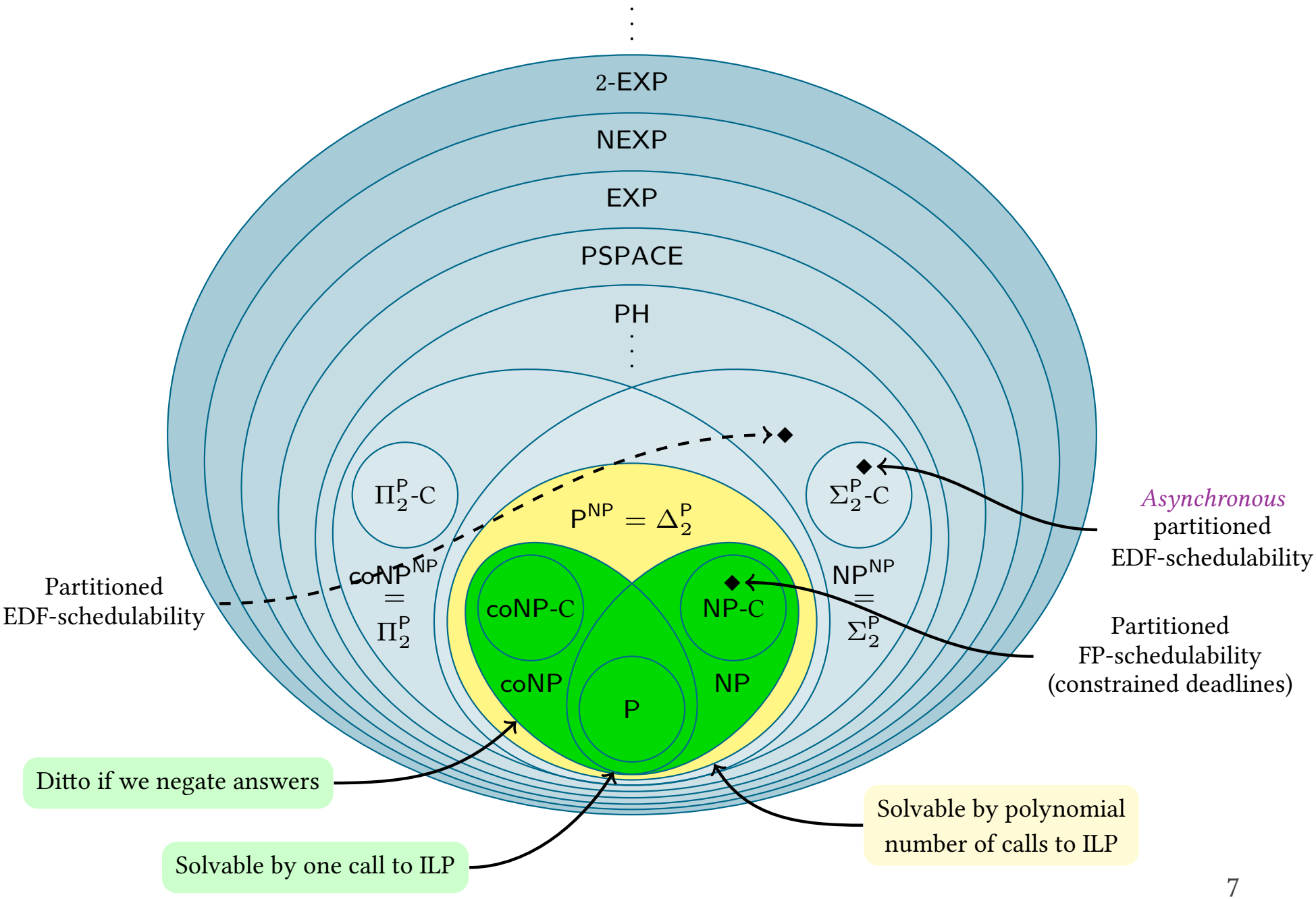
Solvable by one call to ILP

Solvable by polynomial number of calls to ILP

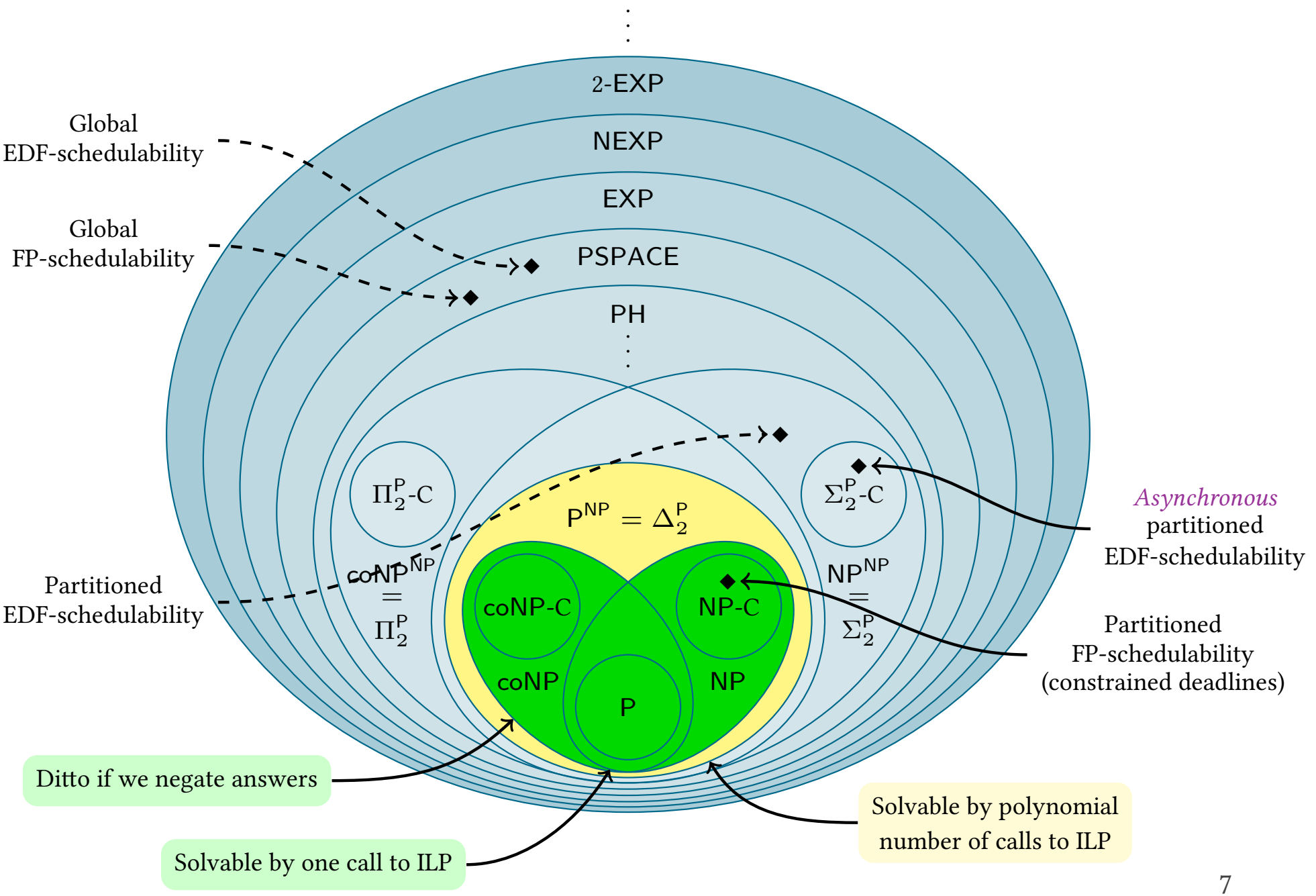
What can be solved with ILPs?



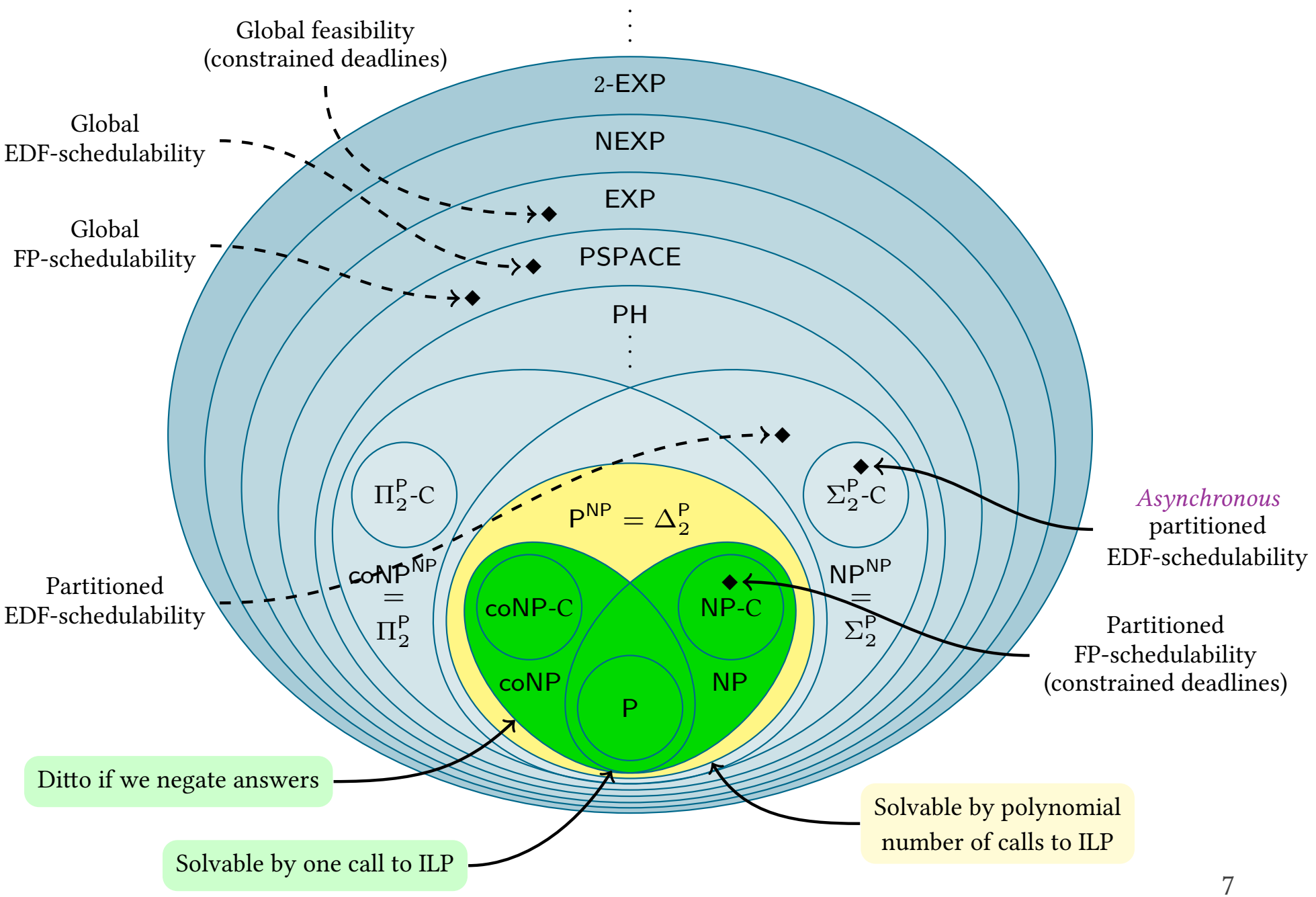
What can be solved with ILPs?



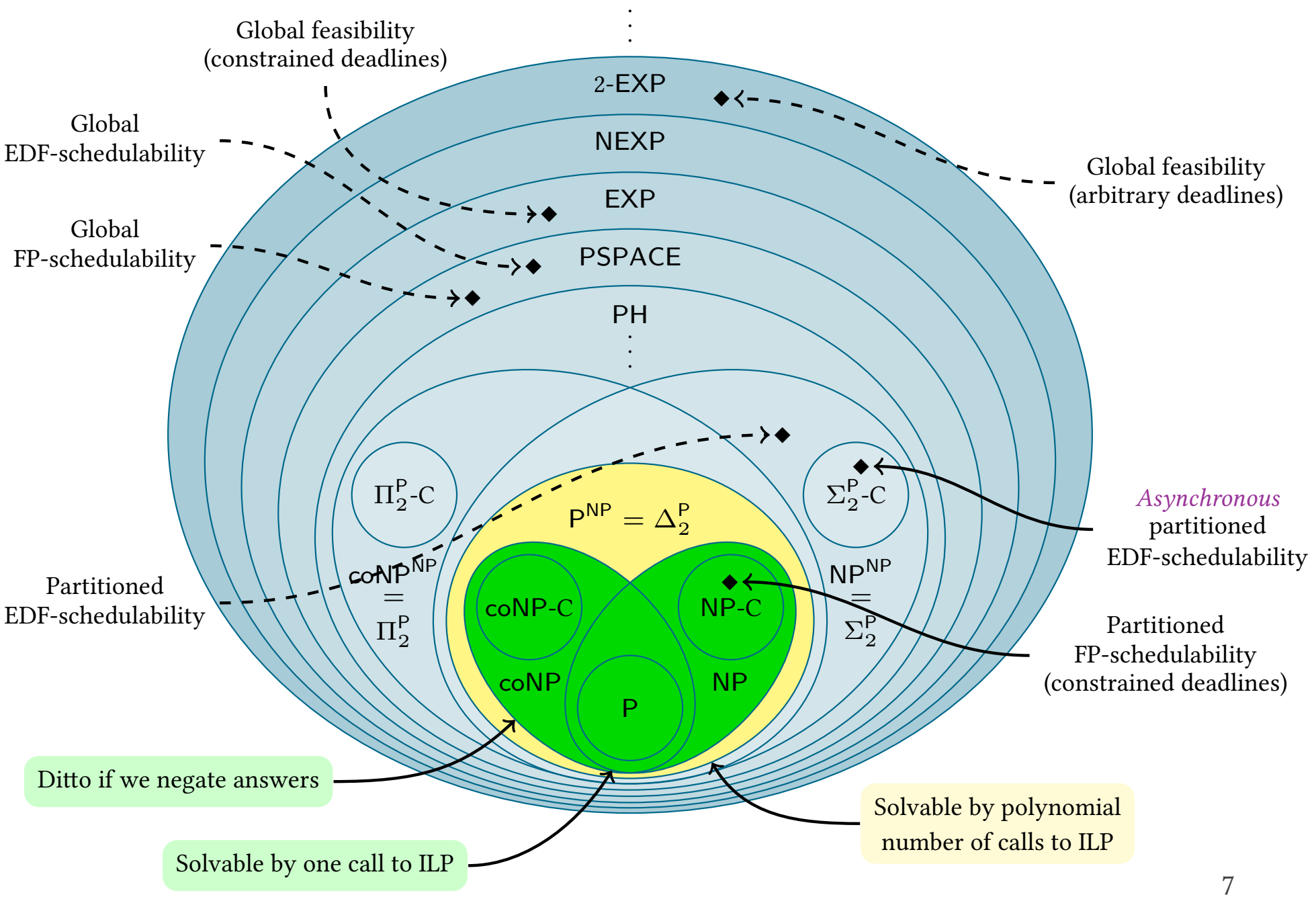
What can be solved with ILPs?



What can be solved with ILPs?



What can be solved with ILPs?



How hard is Σ_2^{P}-hard?

How hard is Σ_2^P-hard?

The trouble with the second quantifier

Gerhard J. Woeginger[1]

Received: 6 February 2021 / Revised: 3 April 2021 / Accepted: 16 April 2021 / Published online: 26 April 2021

Abstract
We survey optimization problems that allow natural simple formulations with one existential and one universal quantifier. We summarize the theoretical background from computational complexity theory, and we present a multitude of illustrating examples. We discuss the connections to robust optimization and to bilevel optimization, and we explain the reasons why the operational research community should be interested in the theoretical aspects of this area.

Keywords Combinatorial optimization · Complexity theory · Polynomial hierarchy · Bilevel optimization

1 Introduction

The United Nations Security Council consists of 15 members: there are five permanent members (China, France, Russia, the United Kingdom, and the USA) and there are ten non-permanent members (which respectively serve for two-year terms). In order to pass a decision (i) at least nine of the fifteen members must agree, and furthermore

How hard is Σ_2^P-hard?

The trouble with the second quantifier

Gerhard J. Woeginger[1]

"Σ_2^P-complete problems are much, much, much, much, much harder than any problem in NP or coNP and anything that can be attacked via ILP solvers [...]."

the theoretical aspects of this area.

Keywords Combinatorial optimization · Complexity theory · Polynomial hierarchy · Bilevel optimization

1 Introduction

The United Nations Security Council consists of 15 members: there are five permanent members (China, France, Russia, the United Kingdom, and the USA) and there are ten non-permanent members (which respectively serve for two-year terms). In order to pass a decision (i) at least nine of the fifteen members must agree, and furthermore

How hard is Σ_2^P-hard?

The trouble with the second quantifier

Gerhard J. Woeginger[1]

"Σ_2^P-complete problems are much, much, much, much, much harder than any problem in NP or coNP and anything that can be attacked via ILP solvers [...]."

the theoretical aspects of this area.

Keywords Combinatorial optimizati... ...y theory · Polynomial hierarchy · Bilevel optimization

1 Introduction

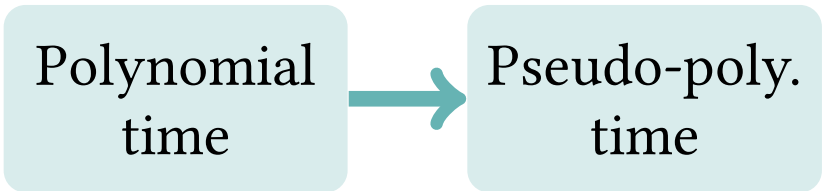
The United Nations Security Council consists of 15 members: there are five permanent members (China, France, Russia, the United Kingdom, and the USA) and there are ten non-permanent members (which respectively serve for two-year terms). In order to pass a decision (i) at least nine of the fifteen members must agree, and furthermore

So…

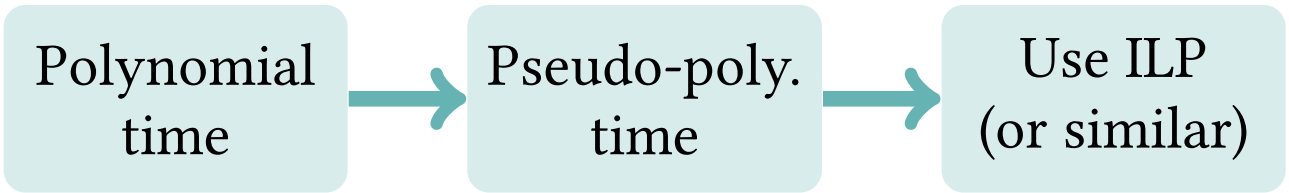
So…

Polynomial time

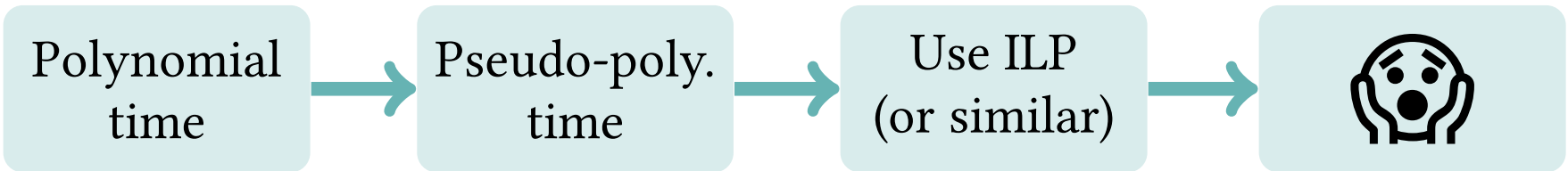
So…



So…



So…



So...

Polynomial time → Pseudo-poly. time → Use ILP (or similar) → 

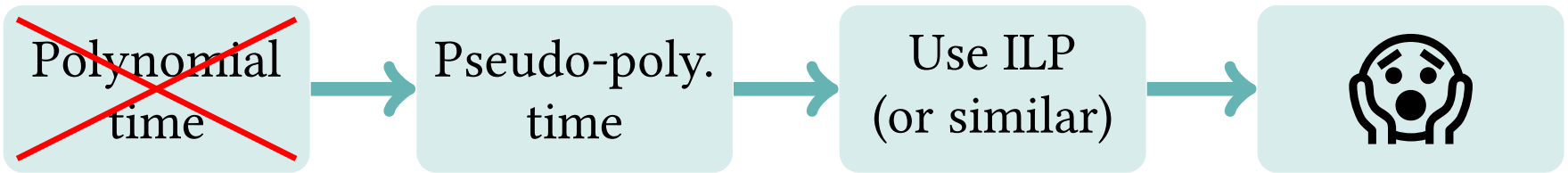
Weakly (co)NP-hard

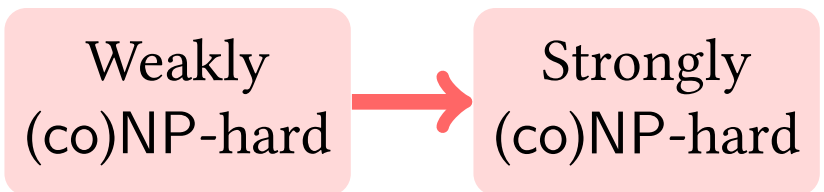
So...

~~Polynomial time~~ → Pseudo-poly. time → Use ILP (or similar) → 

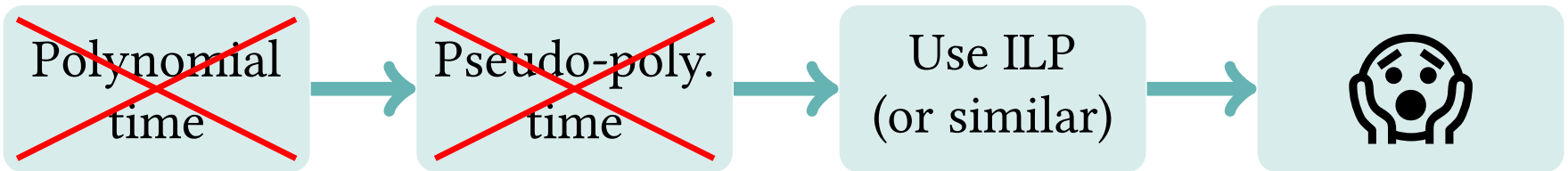
Weakly (co)NP-hard

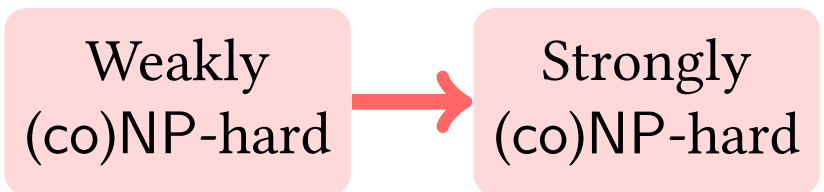
So...


Polynomial time
Pseudo-poly. time
Use ILP (or similar)

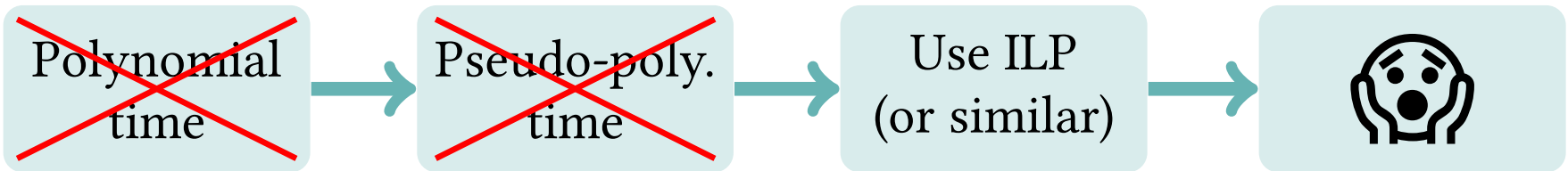

Weakly (co)NP-hard
Strongly (co)NP-hard

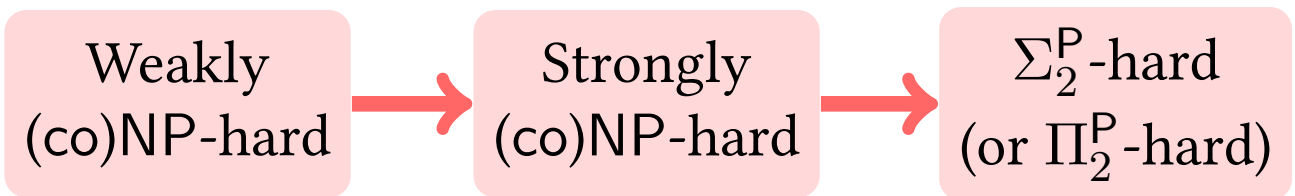
So...


Polynomial time
Pseudo-poly. time
Use ILP (or similar)

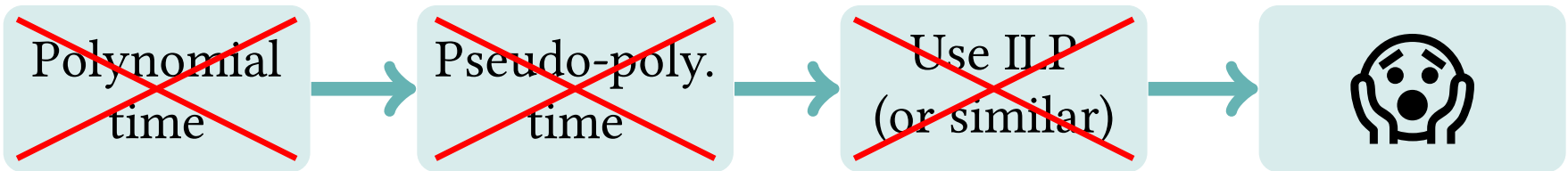

Weakly (co)NP-hard
Strongly (co)NP-hard

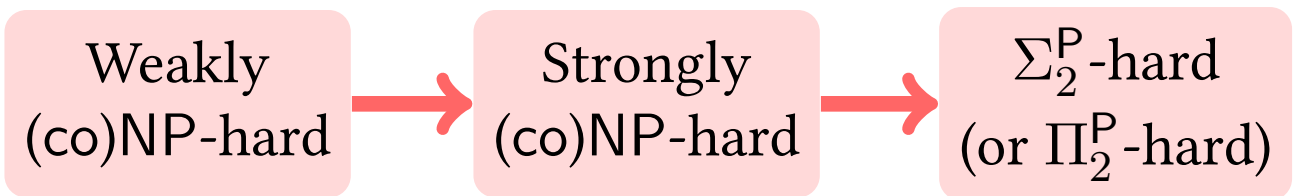
So...


Polynomial time
Pseudo-poly. time
Use ILP (or similar)

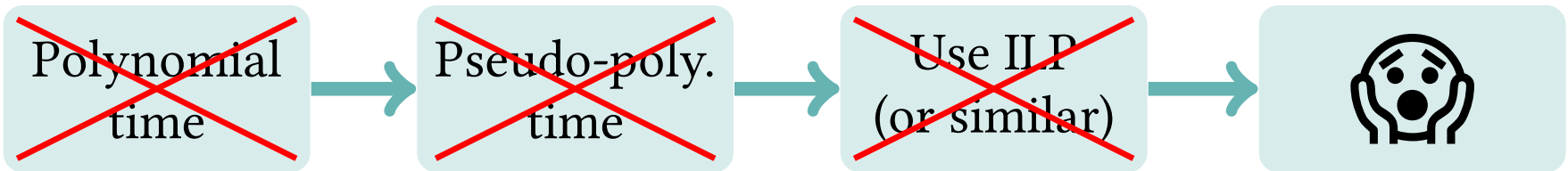

Weakly (co)NP-hard
Strongly (co)NP-hard
Σ_2^P-hard (or Π_2^P-hard)

So...


Polynomial time
Pseudo-poly. time
Use ILP (or similar)

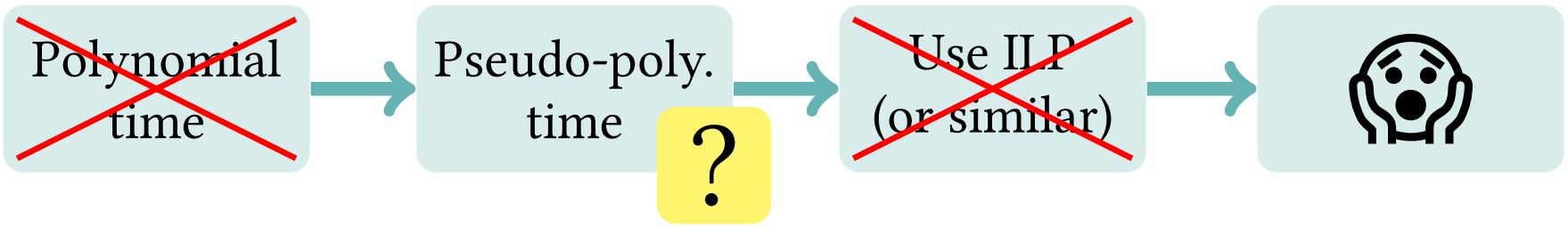

Weakly (co)NP-hard
Strongly (co)NP-hard
Σ_2^P-hard (or Π_2^P-hard)

So...


Polynomial time
Pseudo-poly. time
Use ILP (or similar)

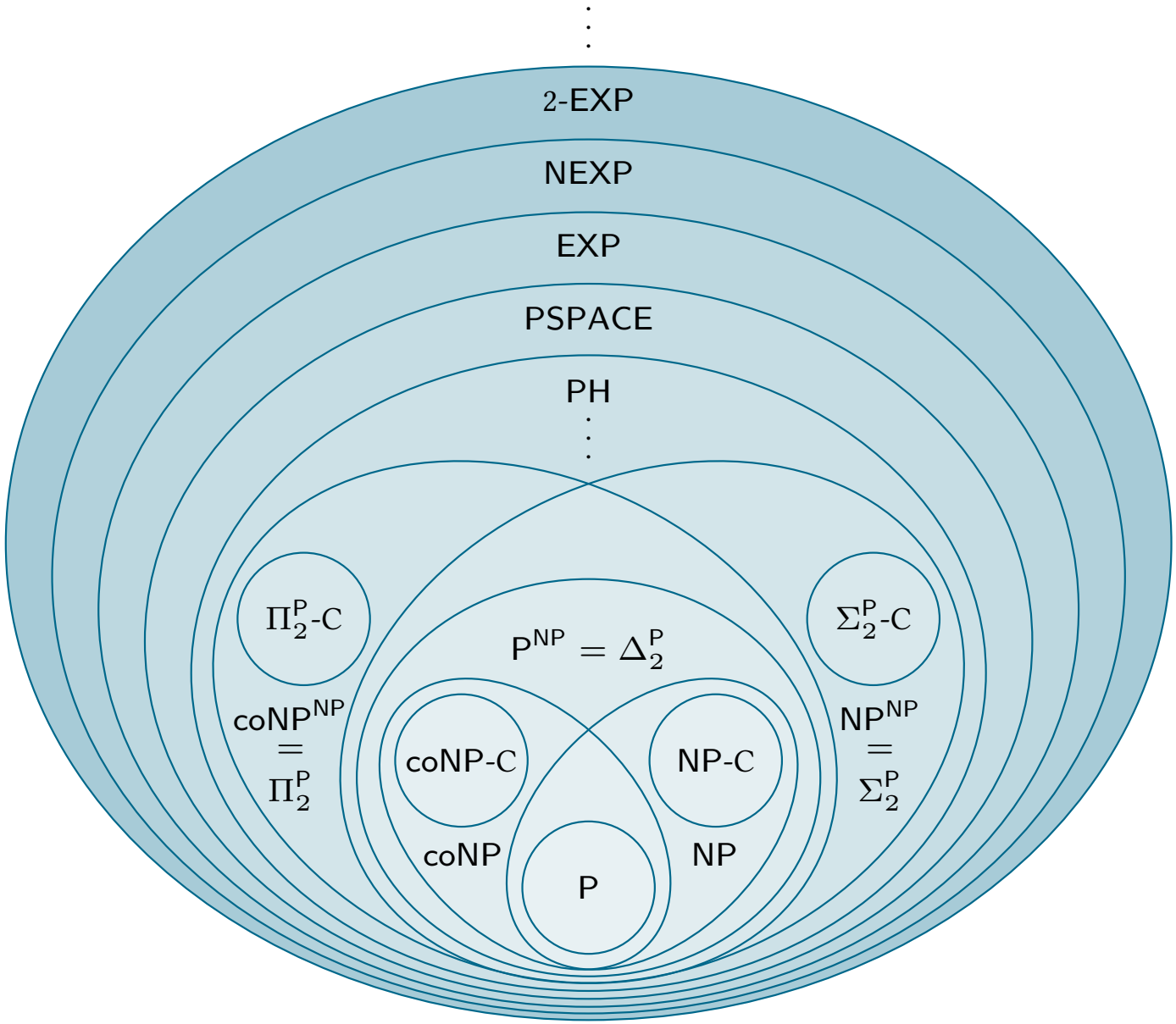

Weakly (co)NP-hard
Strongly (co)NP-hard
Σ_2^P-hard (or Π_2^P-hard)

So...

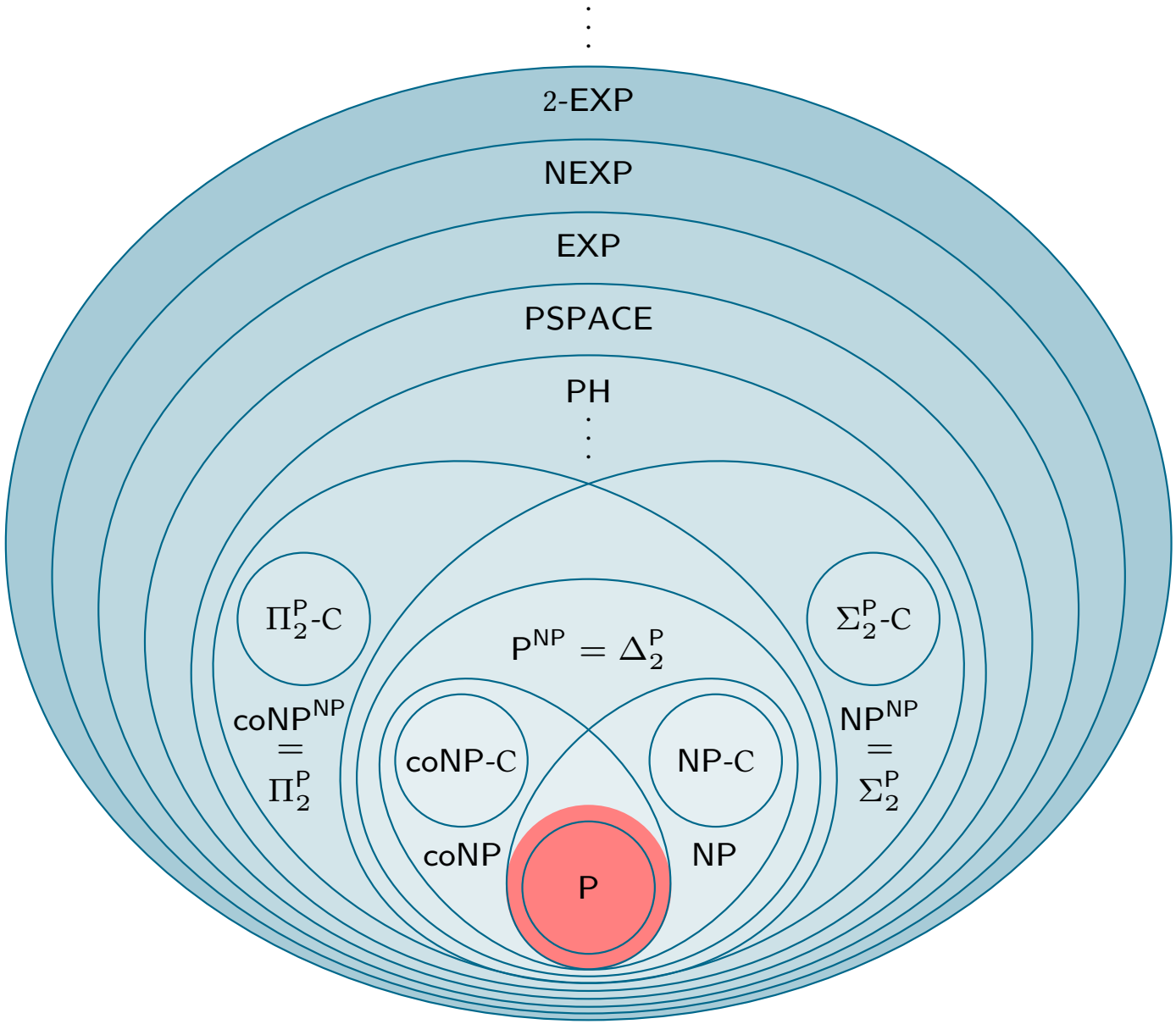

Polynomial time
Pseudo-poly. time
?
Use IIP (or similar)


Weakly (co)NP-hard
Strongly (co)NP-hard
Σ_2^P-hard (or Π_2^P-hard)

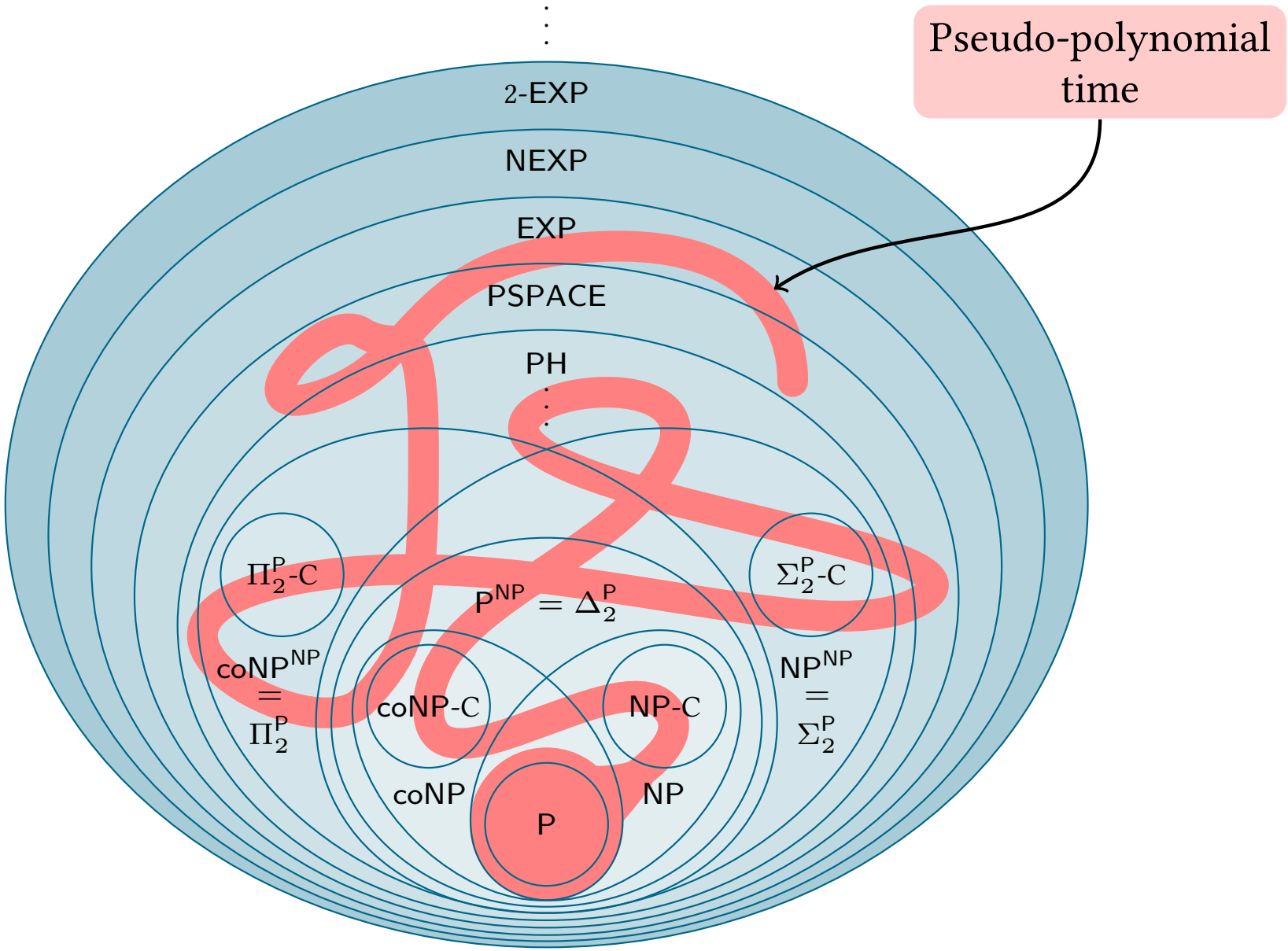
Where is pseudo-polynomial time?



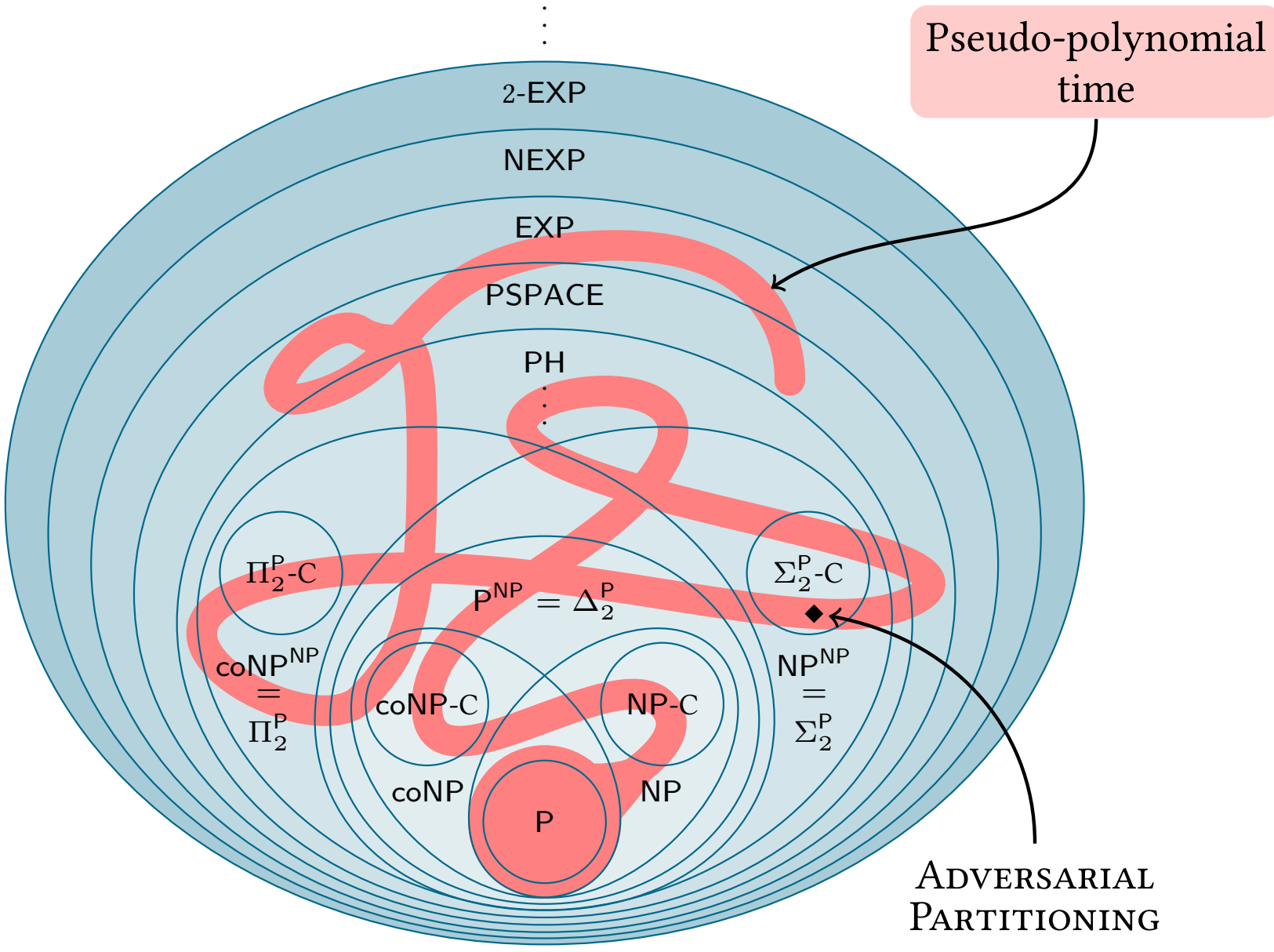
Where is pseudo-polynomial time?



Where is pseudo-polynomial time?



Where is pseudo-polynomial time?



Adversarial Partitioning

Definition

Instance: Sets A and $B \subset \mathbb{N}$, and two bins of capacity S.

Question: Can the items in A be partitioned upon the bins such that the items in B cannot be partitioned upon the remaining capacities?

Adversarial Partitioning

Definition

Instance: Sets A and $B \subset \mathbb{N}$, and two bins of capacity S.

Question: Can the items in A be partitioned upon the bins such that the items in B cannot be partitioned upon the remaining capacities?

Berit Johannes, PhD thesis, 2011

Adversarial Partitioning is Σ_2^{P}-complete.

Adversarial Partitioning

Definition

Instance: Sets A and $B \subset \mathbb{N}$, and two bins of capacity S.

Question: Can the items in A be partitioned upon the bins such that the items in B cannot be partitioned upon the remaining capacities?

Berit Johannes, PhD thesis, 2011

Adversarial Partitioning is Σ_2^{P}-complete.

"Adversarial" problems in general are relevant for security.
Many are Σ_2^{P}-complete.

Adversarial Partitioning

Pseudo-polynomial time algorithm

1. Generate the possible sizes of a partitioning of A (using dynamic programming).
2. Ditto for B.
3. Scan the possible sizes of a partition of A for one that prevents partitioning of B.

Adversarial Partitioning

Pseudo-polynomial time algorithm

1. Generate the possible sizes of a partitioning of A (using dynamic programming).
2. Ditto for B.
3. Scan the possible sizes of a partition of A for one that prevents partitioning of B.

Adversarial Partitioning

Pseudo-polynomial time algorithm

1. Generate the possible sizes of a partitioning of A (using dynamic programming).
2. Ditto for B.
3. Scan the possible sizes of a partition of A for one that prevents partitioning of B.

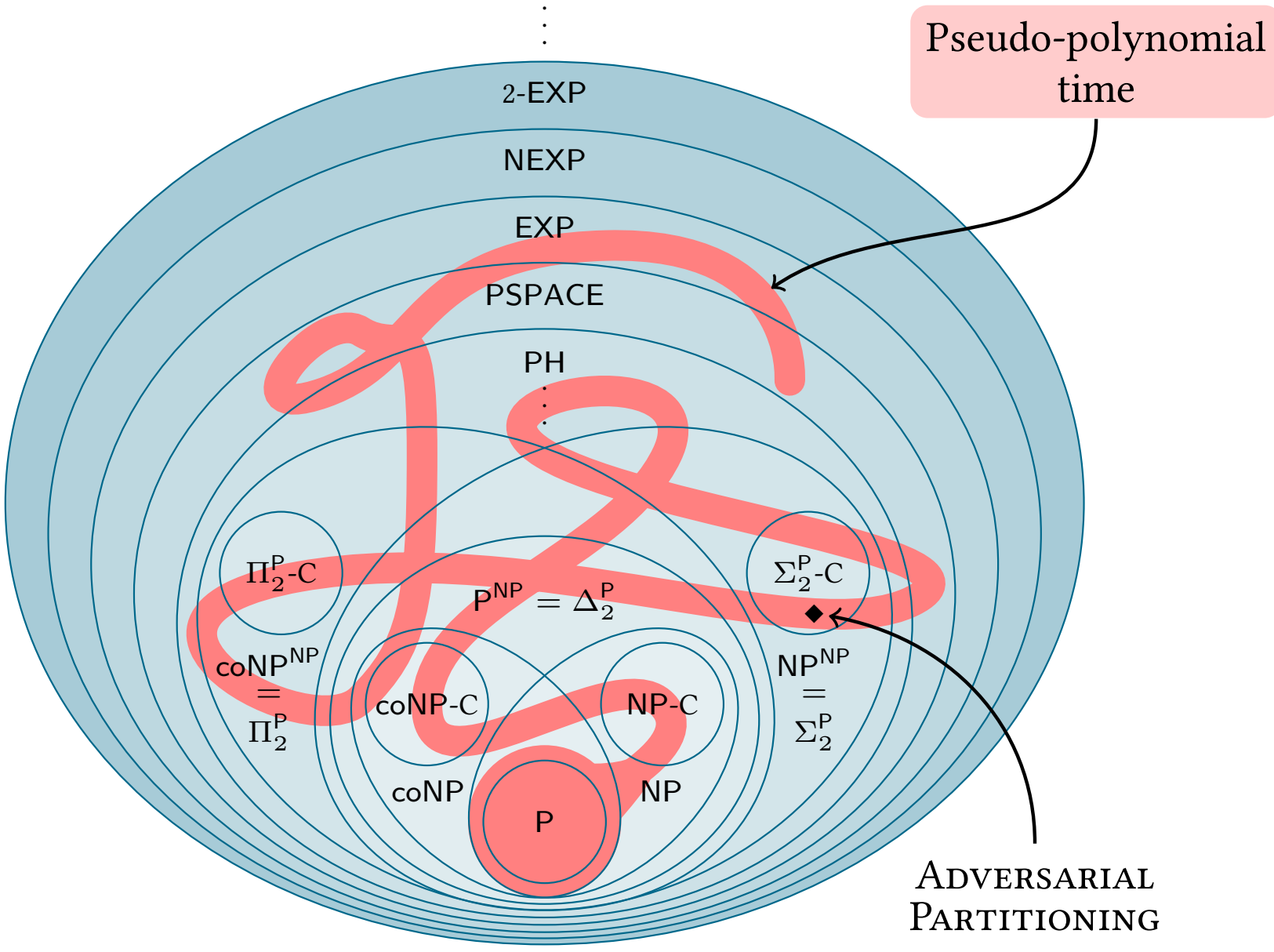
Adversarial Partitioning

Pseudo-polynomial time algorithm

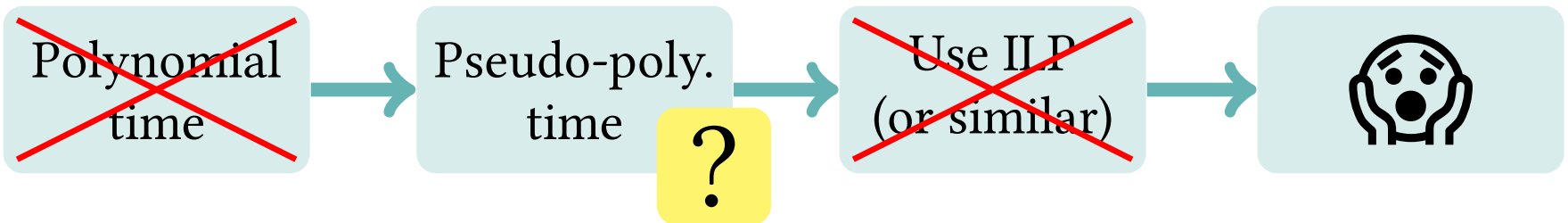
1. Generate the possible sizes of a partitioning of A (using dynamic programming).
2. Ditto for B.
3. Scan the possible sizes of a partition of A for one that prevents partitioning of B.

Runtime is $\mathcal{O}((|A| + |B| + \log S)S)$

Where is pseudo-polynomial time?

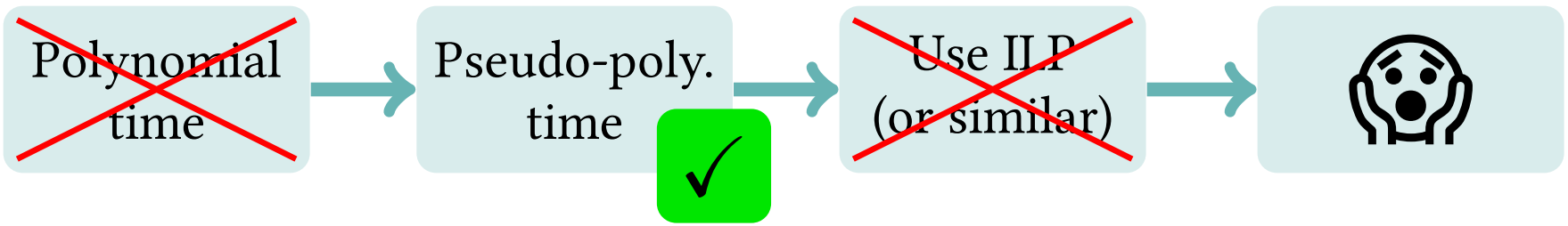


So...


Polynomial time
Pseudo-poly. time
?
Use ILP (or similar)

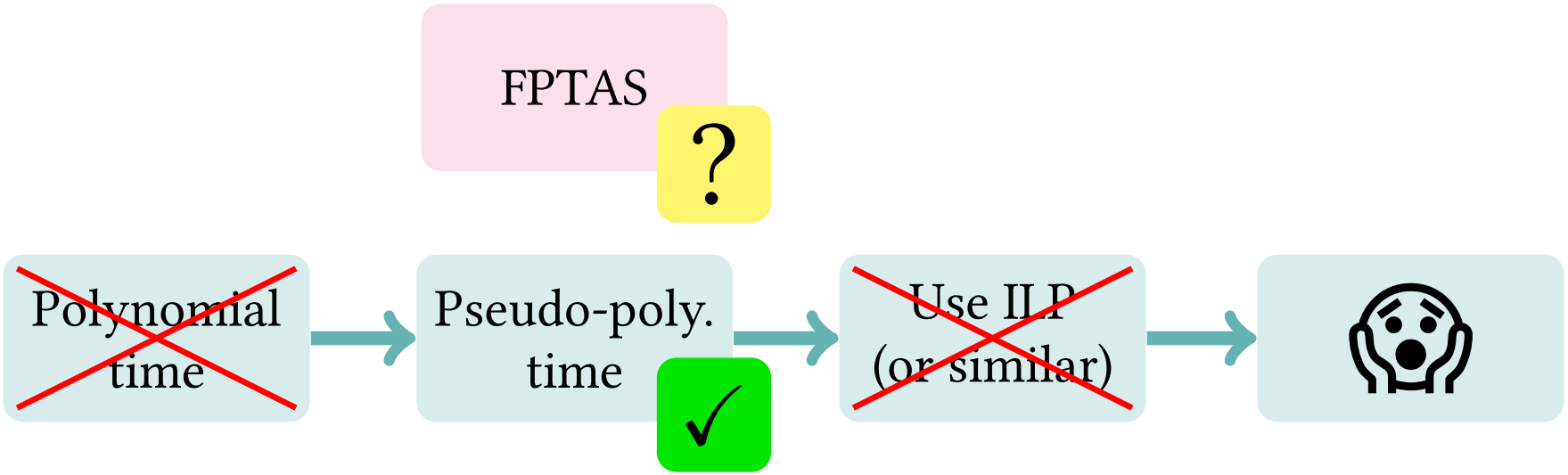

Weakly (co)NP-hard
Strongly (co)NP-hard
Σ_2^P-hard (or Π_2^P-hard)

So...



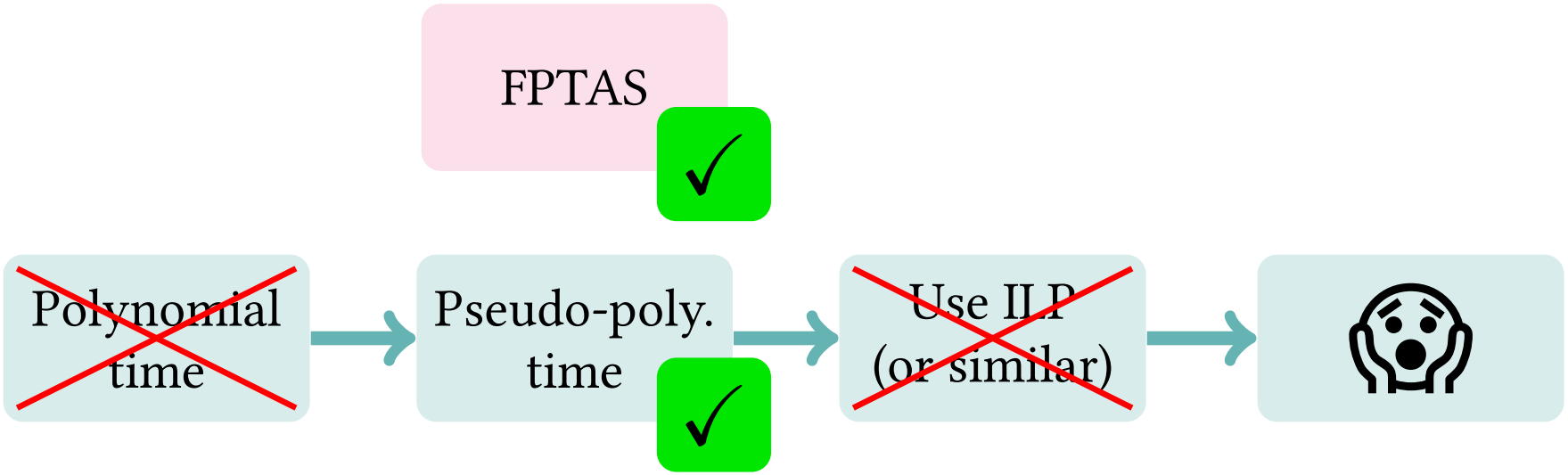
(Adversarial Partitioning)

So...



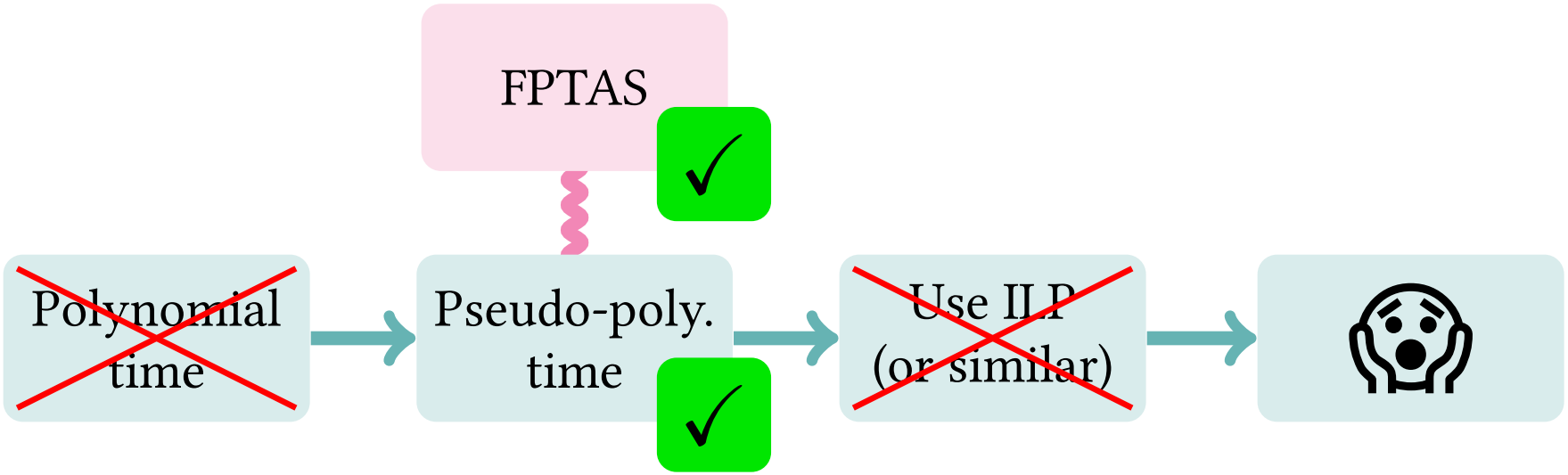
(Adversarial Partitioning)

So...



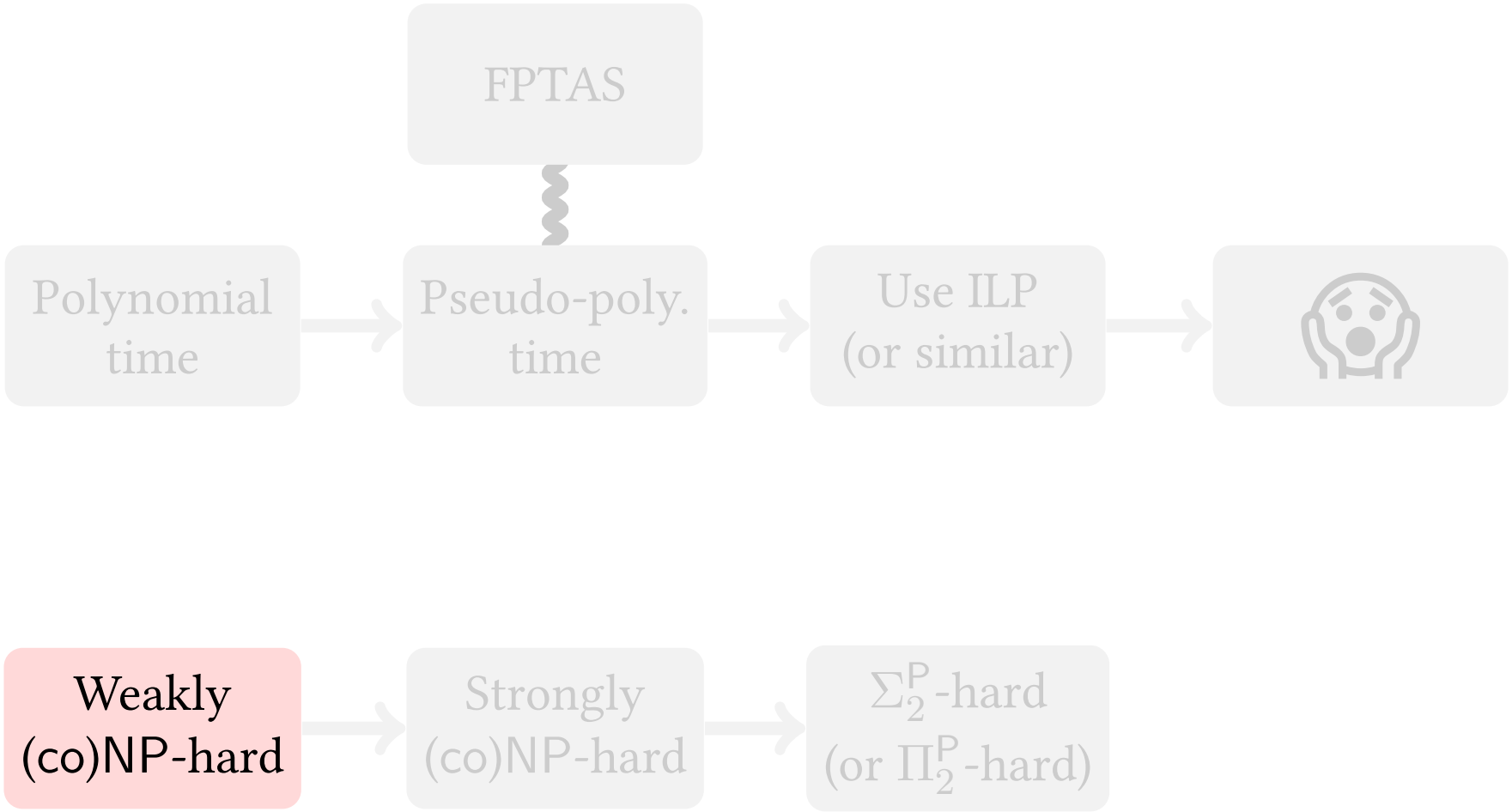
(Adversarial Partitioning)

So...



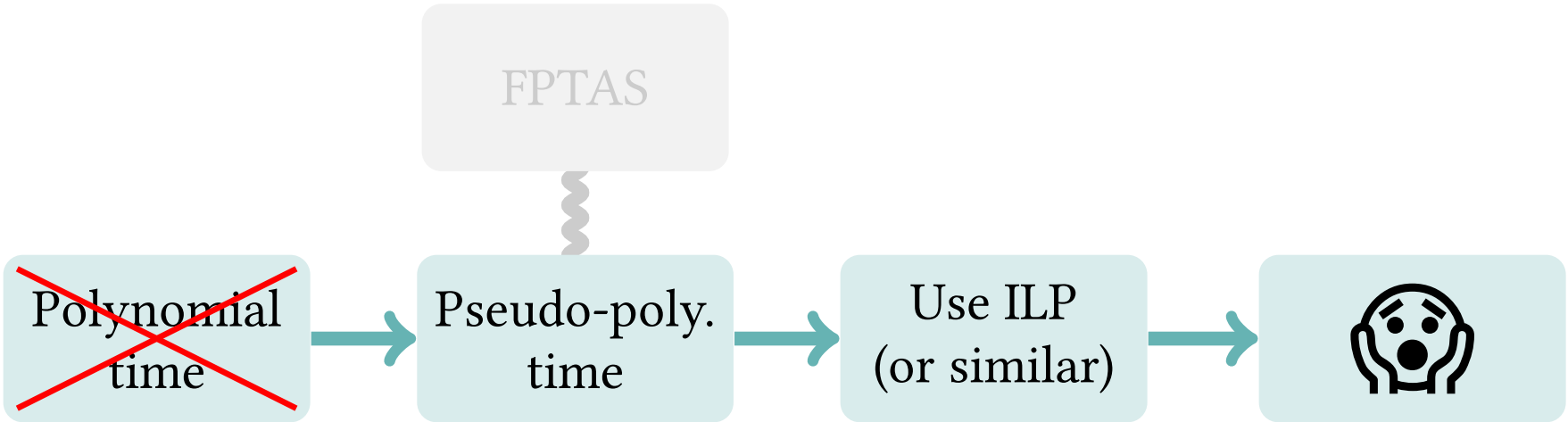
(Adversarial Partitioning)

So...



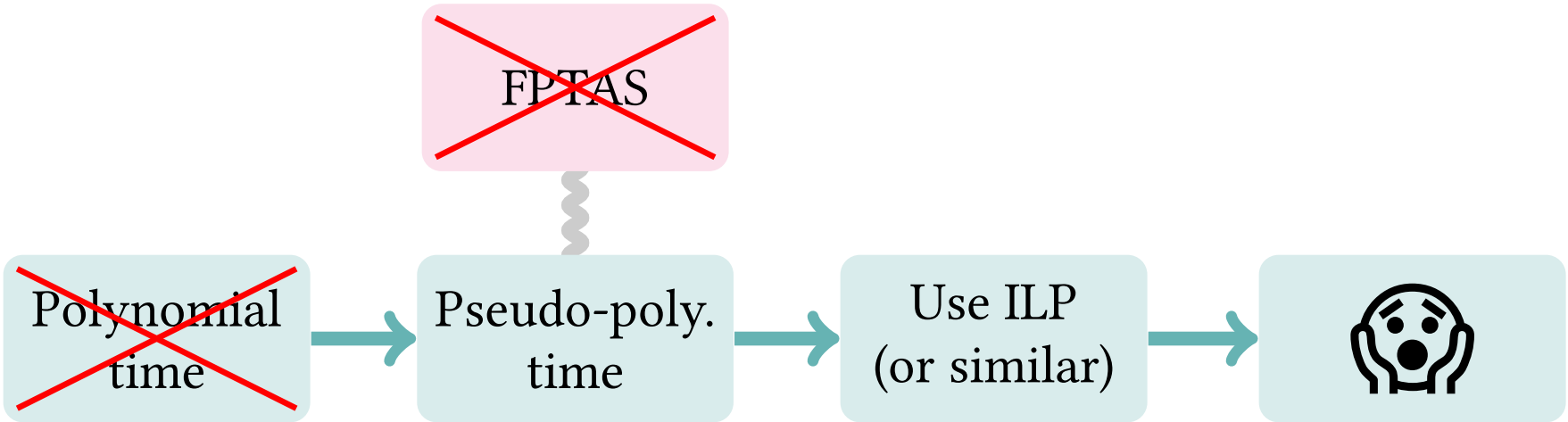
(Memory-Constrained Task Selection)

So...



(Memory-Constrained Task Selection)

So...



(Memory-Constrained Task Selection)

A more fine-grained take on pseudo-polynomial time

Pseudo-polynomial time: $\mathsf{poly}(n, N)$

A more fine-grained take on pseudo-polynomial time

Pseudo-polynomial time: $\mathsf{poly}(n, N)$

But even for "nice" problems, we expect $N \gg n$.

A more fine-grained take on pseudo-polynomial time

Pseudo-polynomial time: $\mathsf{poly}(n, N)$

But even for "nice" problems, we expect $N \gg n$.

$\Downarrow$

Anything but linear in N can quickly get out of hand.

A more fine-grained take on pseudo-polynomial time

Pseudo-polynomial time: $\mathsf{poly}(n, N)$

But even for "nice" problems, we expect $N \gg n$.

$$\Downarrow$$

Anything but linear in N can quickly get out of hand.

Definition

An algorithm is *pseudo-linear* if it is $\mathcal{O}(n^k \times N)$.

A more fine-grained take on pseudo-polynomial time

Pseudo-polynomial time: $\mathsf{poly}(n, N)$

But even for "nice" problems, we expect $N \gg n$.

$\Downarrow$

Anything but linear in N can quickly get out of hand.

Definition

An algorithm is *pseudo-linear* if it is $\mathcal{O}(n^k \times N)$.

A more fine-grained take on pseudo-polynomial time

A more fine-grained take on pseudo-polynomial time

Changing time units should preferably not change running time.

A more fine-grained take on pseudo-polynomial time

Changing time units should preferably not change running time.

Running times should be $\mathsf{poly}(n, N/G)$, where
G is the GCD of the numerical parameters.

A more fine-grained take on pseudo-polynomial time

Changing time units should preferably not change running time.

$$\Downarrow$$

Running times should be $\mathsf{poly}(n, N/G)$, where
G is the GCD of the numerical parameters.

Definition

A running time of $\mathsf{poly}(n, N/G)$ is *robust*.

A more fine-grained take on pseudo-polynomial time

Changing time units should preferably not change running time.

$\Downarrow$

Running times should be $\mathsf{poly}(n, N/G)$, where
G is the GCD of the numerical parameters.

Definition

A running time of $\mathsf{poly}(n, N/G)$ is *robust*.

We want
this as well

$\forall$Thank you!

$\exists$Questions?