On the Hardness of Almost–Sure Termination

Benjamin Kaminski Joost-Pieter Katoen

Software Modeling and Verification Chair | RWTH AACHEN UNIVERSITY

40th International Symposium on
Mathematical Foundations of Computer Science
August 27, 2015

Our Setting: Probabilistic Programs

Our Setting: Probabilistic Programs

- Like ordinary programs . . .

Our Setting: Probabilistic Programs

- Like ordinary programs . . .
  - Assignment: var `:=` $arith\_expr$
  - Concatenation: P_1 `;` P_2
  - While–loop: `while (`$bool\_expr$`)` $\{P\}$

Our Setting: Probabilistic Programs

- Like ordinary programs ...
  - Assignment: var `:=` $arith\_expr$
  - Concatenation: P_1 `;` P_2
  - While–loop: `while (`$bool\_expr$`)` $\{P\}$
- Variable valuations range over $\mathbb{Q}$

Our Setting: Probabilistic Programs

- Like ordinary programs ...
  - Assignment: var `:=` $arith\_expr$
  - Concatenation: P_1 `;` P_2
  - While–loop: `while (`$bool\_expr$`)` $\{P\}$
- Variable valuations range over $\mathbb{Q}$
- Probabilistic nature: the program can toss a (possibly biased) coin and proceed its execution depending on the outcome

Our Setting: Probabilistic Programs

- Like ordinary programs ...
  - Assignment: var `:=` $arith\_expr$
  - Concatenation: P_1 `;` P_2
  - While–loop: `while (`$bool\_expr$`)` $\{P\}$
- Variable valuations range over $\mathbb{Q}$
- Probabilistic nature: the program can toss a (possibly biased) coin and proceed its execution depending on the outcome
  - Probabilistic choice: $\{P_1\}\ [p]\ \{P_2\}$, for $p \in [0,\, 1] \cap \mathbb{Q}$

Our Setting: Probabilistic Programs

- Like ordinary programs ...
  - Assignment: var `:=` $arith\_expr$
  - Concatenation: P_1 `;` P_2
  - While–loop: `while (`$bool\_expr$`)` $\{P\}$
- Variable valuations range over $\mathbb{Q}$
- Probabilistic nature: the program can toss a (possibly biased) coin and proceed its execution depending on the outcome
  - Probabilistic choice: $\{P_1\}\,[p]\,\{P_2\}$, for $p \in [0,\,1] \cap \mathbb{Q}$
- We consider only discrete probabilistic coin tosses; no sampling from continuous distributions or alike

Our Setting: Probabilistic Programs

- Like ordinary programs ...
  - Assignment: var := $arith\_expr$
  - Concatenation: P_1 ; P_2
  - While–loop: `while` $(bool\_expr)$ $\{P\}$
- Variable valuations range over $\mathbb{Q}$
- Probabilistic nature: the program can toss a (possibly biased) coin and proceed its execution depending on the outcome
  - Probabilistic choice: $\{P_1\}\ [p]\ \{P_2\}$, for $p \in [0,\, 1] \cap \mathbb{Q}$
- We consider only discrete probabilistic coin tosses; no sampling from continuous distributions or alike
  - In particular: var := $\mathcal{N}(\mu,\, \sigma^2)$

Our Setting: Probabilistic Programs

- Like ordinary programs ...
  - Assignment: var `:=` $arith\_expr$
  - Concatenation: P_1 `;` P_2
  - While–loop: `while (`$bool\_expr$`)` $\{P\}$
- Variable valuations range over $\mathbb{Q}$
- Probabilistic nature: the program can toss a (possibly biased) coin and proceed its execution depending on the outcome
  - Probabilistic choice: $\{P_1\}\,[p]\,\{P_2\}$, for $p \in [0,\,1] \cap \mathbb{Q}$
- We consider only discrete probabilistic coin tosses; no sampling from continuous distributions or alike
  - In particular: ~~var `:=` $\mathcal{N}(\mu,\,\sigma^2)$~~

Analysis Problems We Consider

Analysis Problems We Consider

- Determine the value of a variable after program execution

Analysis Problems We Consider

- Determine the value of a variable after program execution

  $\Longrightarrow$ Determine expected values

Analysis Problems We Consider

- Determine the value of a variable after program execution

  $\Longrightarrow$ Determine expected values (expected outcomes)

Analysis Problems We Consider

- Determine the value of a variable after program execution

  $\Longrightarrow$ Determine expected values (expected outcomes)

- Decide whether the program terminates

Analysis Problems We Consider

- Determine the value of a variable after program execution

  $\Longrightarrow$ Determine expected values (expected outcomes)

- Decide whether the program terminates

  $\Longrightarrow$ Decide almost–sure termination

Analysis Problems We Consider

- Determine the value of a variable after program execution

  $\Longrightarrow$ Determine expected values (expected outcomes)

- Decide whether the program terminates (in an expected finite number of steps)

  $\Longrightarrow$ Decide almost–sure termination

Analysis Problems We Consider

- Determine the value of a variable after program execution

  $\Longrightarrow$ Determine expected values (expected outcomes)

- Decide whether the program terminates (in an expected finite number of steps)

  $\Longrightarrow$ Decide (positive) almost–sure termination

Analysis Problems We Consider

- Determine the value of a variable after program execution

  $\Longrightarrow$ Determine expected values (expected outcomes)

- Decide whether the program terminates (in an expected finite number of steps) [on all inputs]

  $\Longrightarrow$ Decide (positive) almost–sure termination

Analysis Problems We Consider

- Determine the value of a variable after program execution

  $\Longrightarrow$ Determine expected values (expected outcomes)

- Decide whether the program terminates (in an expected finite number of steps) [on all inputs]

  $\Longrightarrow$ Decide [universal] (positive) almost–sure termination

Analysis Problems We Consider

- Determine the value of a variable after program execution

  $\Longrightarrow$ Determine expected values (expected outcomes)

- Decide whether the program terminates (in an expected finite number of steps) [on all inputs]

  $\Longrightarrow$ Decide [universal] (positive) almost–sure termination

How hard is it to solve these analysis problems?

Dissent in the Literature

[Morgan 1996]

"[...] probabilistic reasoning for partial correctness [...] is not much more complex than standard reasoning."

Dissent in the Literature

[Morgan 1996]

"[...] probabilistic reasoning for partial correctness [...] is not much more complex than standard reasoning."

[Esparza *et al.* 2012]

"[Ordinary] termination is a purely topological property [...], but almost–sure termination is not.

Dissent in the Literature

[Morgan 1996]

"[...] probabilistic reasoning for partial correctness [...] is not much more complex than standard reasoning."

[Esparza *et al.* 2012]

"[Ordinary] termination is a purely topological property [...], but almost–sure termination is not. [...] Proving almost–sure termination requires arithmetic reasoning not offered by termination provers."

Decision Problems We Studied

Lower and Upper Bounds, and Exact Expected Outcomes

$(P, \eta, v, q) \in \mathcal{LEXP} \quad :\Longleftrightarrow \quad$ "The e. v. of v after termination of P on input η is $> q$"

Lower and Upper Bounds, and Exact Expected Outcomes

$(P, \eta, v, q) \in \mathcal{LEXP} \quad :\Longleftrightarrow \quad$ "The e. v. of v after termination of P on input η is $> q$"

$(P, \eta, v, q) \in \mathcal{REXP} \quad :\Longleftrightarrow \quad$ "The e. v. of v after termination of P on input η is $< q$"

Lower and Upper Bounds, and Exact Expected Outcomes

$$(P, \eta, v, q) \in \mathcal{LEXP} \quad :\Longleftrightarrow \quad \text{“The e. v. of } v \text{ after termination of } P \text{ on input } \eta \text{ is } > q\text{”}$$

$$(P, \eta, v, q) \in \mathcal{REXP} \quad :\Longleftrightarrow \quad \text{“The e. v. of } v \text{ after termination of } P \text{ on input } \eta \text{ is } < q\text{”}$$

$$(P, \eta, v, q) \in \mathcal{EXP} \quad :\Longleftrightarrow \quad \text{“The e. v. of } v \text{ after termination of } P \text{ on input } \eta \text{ is } = q\text{”}$$

(Positive) Almost–Sure Termination $\mathcal{PAST}$

$$(P, \eta) \in \mathcal{AST} \quad :\Longleftrightarrow \quad \text{“}P \text{ terminates with probability 1 on input } \eta\text{”}$$

(Positive) Almost–Sure Termination $\mathcal{PAST}$

$$(P, \eta) \in \mathcal{AST} \quad :\Longleftrightarrow \quad \text{“}P \text{ terminates with probability 1 on input } \eta\text{”}$$

$$(P, \eta) \in \mathcal{PAST} \quad :\Longleftrightarrow \quad \text{“The expected execution time of } P \text{ on input } \eta \text{ is finite”}$$

(Positive) Almost–Sure Termination $\mathcal{PAST}$

$$(P, \eta) \in \mathcal{AST} \quad :\Longleftrightarrow \quad \text{“}P \text{ terminates with probability 1 on input } \eta\text{”}$$

$$(P, \eta) \in \mathcal{PAST} \quad :\Longleftrightarrow \quad \text{“The expected execution time of } P \text{ on input } \eta \text{ is finite”}$$

Notice $\mathcal{PAST} \subsetneq \mathcal{AST}$.

(Positive) Almost–Sure Termination $\mathcal{PAST}$

$$(P, \eta) \in \mathcal{AST} \quad :\Longleftrightarrow \quad \text{“}P \text{ terminates with probability 1 on input } \eta\text{”}$$

$$(P, \eta) \in \mathcal{PAST} \quad :\Longleftrightarrow \quad \text{“The expected execution time of } P \text{ on input } \eta \text{ is finite”}$$

Notice $\mathcal{PAST} \subsetneq \mathcal{AST}$.

Universal Versions of $\mathcal{AST}$ and $\mathcal{PAST}$

$$P \in \mathcal{UAST} \quad :\Longleftrightarrow \forall \eta\colon (P, \eta) \in \mathcal{AST}$$
$$P \in \mathcal{UPAST} \quad :\Longleftrightarrow \forall \eta\colon (P, \eta) \in \mathcal{PAST}$$

Hardness Results

The Arithmetical Hierarchy

The Arithmetical Hierarchy

- Class Σ_n^0 is defined as

$$\Sigma_n^0 = \Big\{ A \;\Big|\; A = \{ \vec{x} \mid \exists y_1 \, \forall y_2 \, \exists y_3 \, \cdots \exists/\forall y_n : (\vec{x}, y_1, y_2, y_3, \ldots, y_n) \in R \}, R \text{ is a decidable relation} \Big\}$$

The Arithmetical Hierarchy

- Class Σ_n^0 is defined as

$$\Sigma_n^0 = \Big\{ A \;\Big|\; A = \big\{ \vec{x} \;\big|\; \exists y_1\, \forall y_2\, \exists y_3\, \cdots\, \exists/\forall y_n : (\vec{x}, y_1, y_2, y_3, \ldots, y_n) \in R \big\}, R \text{ is a decidable relation} \Big\}$$

- Class Π_n^0 is defined as

$$\Pi_n^0 = \Big\{ A \;\Big|\; A = \big\{ \vec{x} \;\big|\; \forall y_1\, \exists y_2\, \forall y_3\, \cdots\, \exists/\forall y_n : (\vec{x}, y_1, y_2, y_3, \ldots, y_n) \in R \big\}, R \text{ is a decidable relation} \Big\}$$

The Arithmetical Hierarchy

- Class Σ_n^0 is defined as

$$\Sigma_n^0 = \Big\{ A \;\Big|\; A = \big\{ \vec{x} \;\big|\; \exists y_1\, \forall y_2\, \exists y_3\, \cdots\, \exists/\forall y_n : (\vec{x}, y_1, y_2, y_3, \ldots, y_n) \in R \big\}, R \text{ is a decidable relation} \Big\}$$

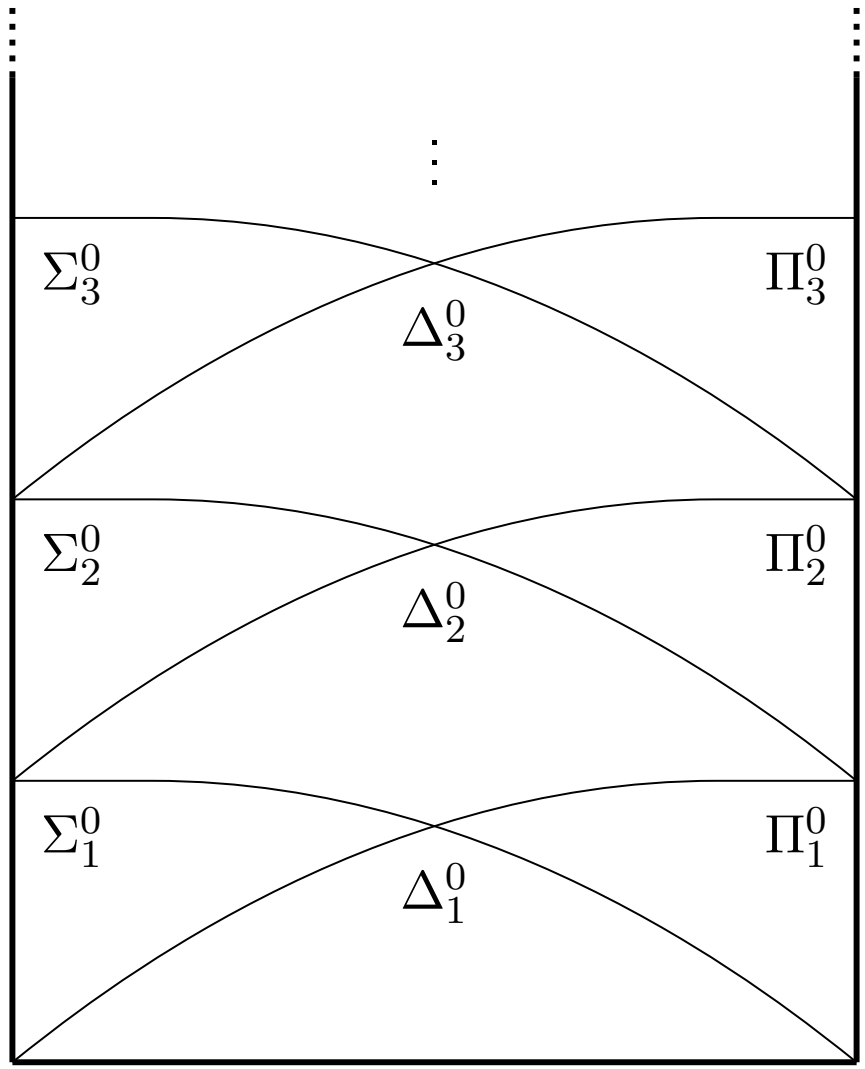
- Class Π_n^0 is defined as

$$\Pi_n^0 = \Big\{ A \;\Big|\; A = \big\{ \vec{x} \;\big|\; \forall y_1\, \exists y_2\, \forall y_3\, \cdots\, \exists/\forall y_n : (\vec{x}, y_1, y_2, y_3, \ldots, y_n) \in R \big\}, R \text{ is a decidable relation} \Big\}$$

- Class Δ_n^0 is defined as $\Delta_n^0 = \Sigma_n^0 \cap \Pi_n^0$

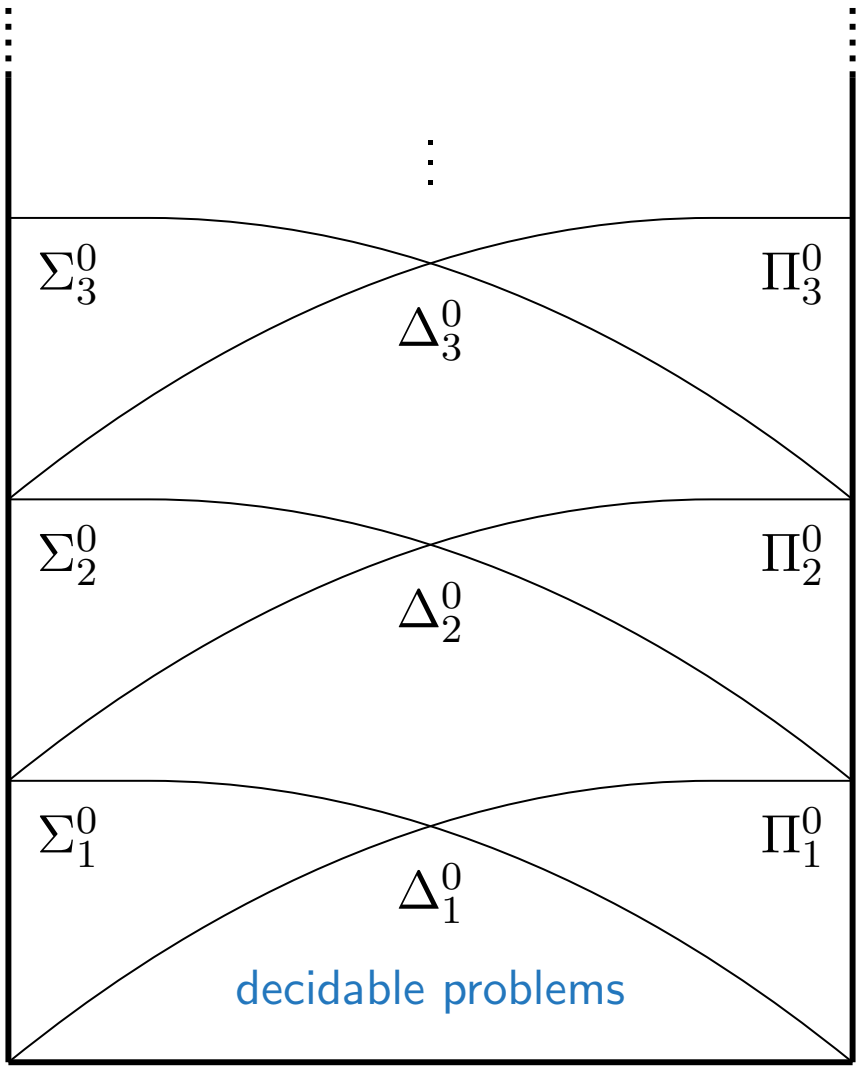
The Arithmetical Hierarchy — The Bigger Picture

The following inclusion diagram holds (all inclusions are strict):



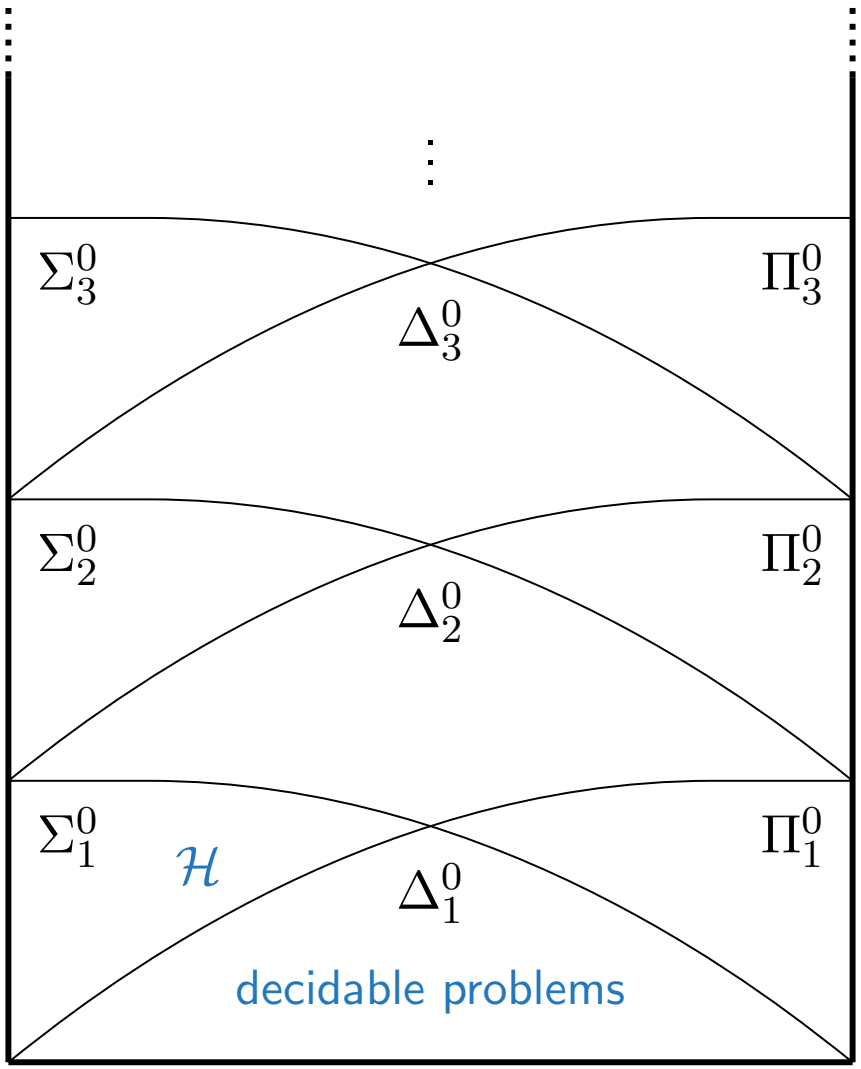
The Arithmetical Hierarchy — The Bigger Picture

The following inclusion diagram holds (all inclusions are strict):



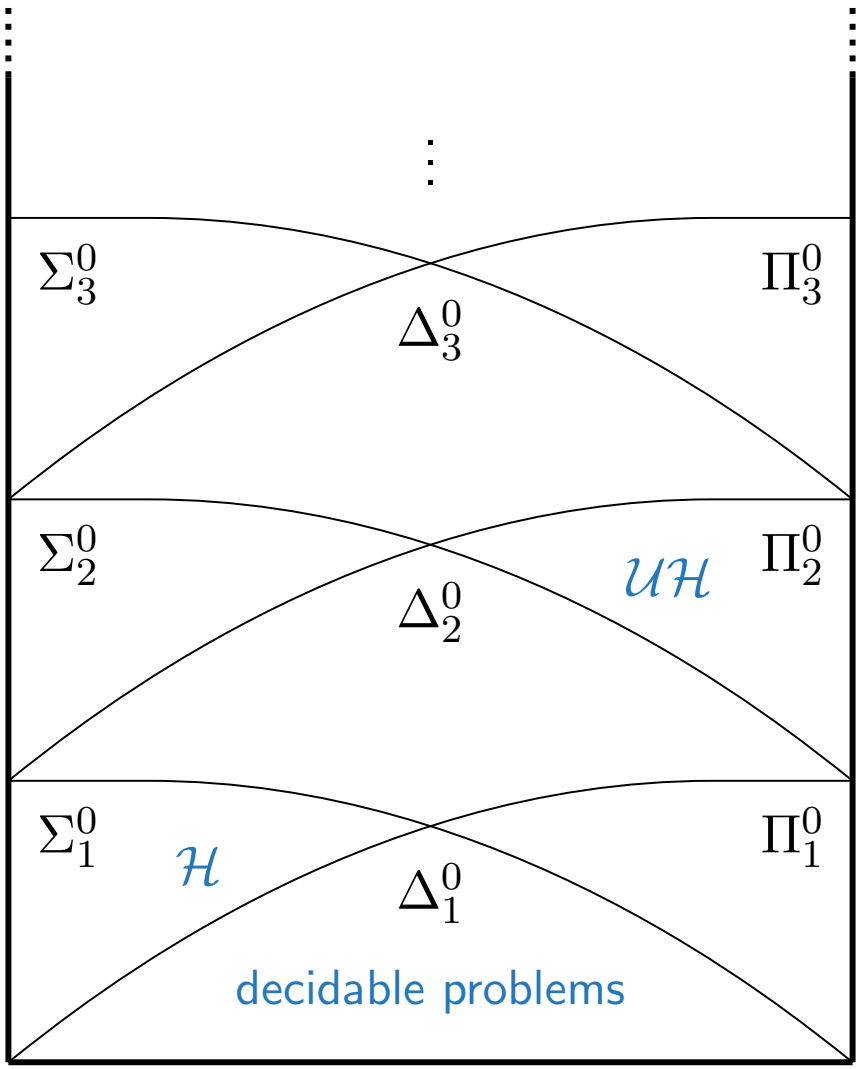
The Arithmetical Hierarchy — The Bigger Picture

The following inclusion diagram holds (all inclusions are strict):



The Arithmetical Hierarchy — The Bigger Picture

The following inclusion diagram holds (all inclusions are strict):



Hardness Results for Expected Outcomes

Hardness Results for Expected Outcomes

Problem	Hardness
$\mathcal{LEXP}$	
$\mathcal{REXP}$	
$\mathcal{EXP}$	

Hardness Results for Expected Outcomes

Problem	Hardness
$\mathcal{LEXP}$	Σ_1^0–complete
$\mathcal{REXP}$	
$\mathcal{EXP}$	

Hardness Results for Expected Outcomes

Problem	Hardness
$\mathcal{LEXP}$	Σ_1^0–complete
$\mathcal{REXP}$	Σ_2^0–complete
$\mathcal{EXP}$	

Hardness Results for Expected Outcomes

Problem	Hardness
$\mathcal{LEXP}$	Σ_1^0–complete
$\mathcal{REXP}$	Σ_2^0–complete
$\mathcal{EXP}$	Π_2^0–complete

Hardness Results for Probabilistic Termination

Hardness Results for Probabilistic Termination

Problem	Hardness
$\mathcal{H}$	
$\mathcal{UH}$	

Hardness Results for Probabilistic Termination

Problem	Hardness
$\mathcal{H}$	Σ_1^0–complete
$\mathcal{UH}$	

Hardness Results for Probabilistic Termination

Problem	Hardness
$\mathcal{H}$	Σ_1^0–complete
$\mathcal{UH}$	Π_2^0–complete

Hardness Results for Probabilistic Termination

Problem	Hardness
$\mathcal{H}$	Σ_1^0–complete
$\mathcal{UH}$	Π_2^0–complete
$\mathcal{AST}$	
$\mathcal{UAST}$	

Hardness Results for Probabilistic Termination

Problem	Hardness
$\mathcal{H}$	Σ_1^0–complete
$\mathcal{UH}$	Π_2^0–complete
$\mathcal{AST}$	Π_2^0–complete
$\mathcal{UAST}$	

Hardness Results for Probabilistic Termination

Problem	Hardness
$\mathcal{H}$	Σ_1^0–complete
$\mathcal{UH}$	Π_2^0–complete
$\mathcal{AST}$	Π_2^0–complete
$\mathcal{UAST}$	Π_2^0–complete

Hardness Results for Probabilistic Termination

Problem	Hardness
$\mathcal{H}$	Σ_1^0–complete
$\mathcal{UH}$	Π_2^0–complete
$\mathcal{AST}$	Π_2^0–complete
$\mathcal{UAST}$	Π_2^0–complete
$\mathcal{PAST}$	
$\mathcal{UPAST}$	

Hardness Results for Probabilistic Termination

Problem	Hardness
$\mathcal{H}$	Σ_1^0–complete
$\mathcal{UH}$	Π_2^0–complete
$\mathcal{AST}$	Π_2^0–complete
$\mathcal{UAST}$	Π_2^0–complete
$\mathcal{PAST}$	Σ_2^0–complete
$\mathcal{UPAST}$	

Hardness Results for Probabilistic Termination

Problem	Hardness
$\mathcal{H}$	Σ_1^0–complete
$\mathcal{UH}$	Π_2^0–complete
$\mathcal{AST}$	Π_2^0–complete
$\mathcal{UAST}$	Π_2^0–complete
$\mathcal{PAST}$	Σ_2^0–complete
$\mathcal{UPAST}$	Π_3^0–complete

Hardness Results for Probabilistic Termination

Problem	Hardness
$\mathcal{H}$	Σ_1^0–complete
$\mathcal{UH}$	Π_2^0–complete
$\mathcal{AST}$	Π_2^0–complete
$\mathcal{UAST}$	Π_2^0–complete
$\mathcal{PAST}$	Σ_2^0–complete
$\mathcal{UPAST}$	Π_3^0–complete

Hardness of Deciding Positive Almost–Sure Termination

Theorem

$\mathcal{PAST}$ is Σ_2^0–complete

Hardness of Deciding Positive Almost–Sure Termination

Theorem

$\mathcal{PAST}$ is Σ_2^0–complete

Proof: Prove $\mathcal{PAST} \in \Sigma_2^0$ and prove $\overline{\mathcal{UH}} \leq_{\mathsf{m}} \mathcal{PAST}$.

Hardness of Deciding Positive Almost–Sure Termination

Theorem

$\mathcal{PAST}$ is Σ_2^0–complete

Proof: Prove $\mathcal{PAST} \in \Sigma_2^0$ and prove $\overline{\mathcal{UH}} \leq_{\mathsf{m}} \mathcal{PAST}$.

The (Un)intuition for $\overline{\mathcal{UH}} \leq_{\mathsf{m}} \mathcal{PAST}$

Hardness of Deciding Positive Almost–Sure Termination

Theorem

$\mathcal{PAST}$ is Σ_2^0–complete

Proof: Prove $\mathcal{PAST} \in \Sigma_2^0$ and prove $\overline{\mathcal{UH}} \leq_{\mathsf{m}} \mathcal{PAST}$.

The (Un)intuition for $\overline{\mathcal{UH}} \leq_{\mathsf{m}} \mathcal{PAST}$

Reduction function computes from every ordinary program Q that does *not* terminate on all inputs,

Hardness of Deciding Positive Almost–Sure Termination

Theorem

$\mathcal{PAST}$ is Σ_2^0–complete

Proof: Prove $\mathcal{PAST} \in \Sigma_2^0$ and prove $\overline{\mathcal{UH}} \leq_{\mathsf{m}} \mathcal{PAST}$.

The (Un)intuition for $\overline{\mathcal{UH}} \leq_{\mathsf{m}} \mathcal{PAST}$

Reduction function computes from every ordinary program Q that does *not* terminate on all inputs, a probabilistic program P (depending on Q) and an input η,

Hardness of Deciding Positive Almost–Sure Termination

Theorem

$\mathcal{PAST}$ is Σ_2^0–complete

Proof: Prove $\mathcal{PAST} \in \Sigma_2^0$ and prove $\overline{\mathcal{UH}} \leq_{\mathsf{m}} \mathcal{PAST}$.

The (Un)intuition for $\overline{\mathcal{UH}} \leq_{\mathsf{m}} \mathcal{PAST}$

Reduction function computes from every ordinary program Q that does *not* terminate on all inputs, a probabilistic program P (depending on Q) and an input η, such that P does terminate in an expected finite number of steps on input η.

The Construction for $\overline{\mathcal{UH}} \leq \mathcal{PAST}$

```
continue := 1;
while (continue ≠ 0){
    {continue := 0} [0.5] {continue := 1}
}
```

The Construction for $\overline{\mathcal{UH}} \leq \mathcal{PAST}$

```
continue := 1;
while (continue ≠ 0){
    {continue := 0} [0.5] {continue := 1}
}
```

Expected runtime (integral over the bars):

The Construction for $\overline{\mathcal{UH}} \leq \mathcal{PAST}$

Given ordinary program Q.

The Construction for $\overline{\mathcal{UH}} \leq \mathcal{PAST}$

Given ordinary program Q. Construct probabilistic program P:

The Construction for $\overline{\mathcal{UH}} \leq \mathcal{PAST}$

Given ordinary program Q. Construct probabilistic program P:

```
continue := 1;
flips := 0;
i := 0;
while (continue ≠ 0){

        flips := flips + 1;
        {continue := 0} [0.5] {continue := 1}

}
```

The Construction for $\overline{\mathcal{UH}} \leq \mathcal{PAST}$

Given ordinary program Q. Construct probabilistic program P:

```
continue := 1;
flips := 0;
i := 0;
while (continue ≠ 0){
    Simulate Q for one (further) step on input g_Q(i);

        flips := flips + 1;
        {continue := 0} [0.5] {continue := 1}

}
```

where $g_Q \colon \mathbb{N} \to \mathit{Inputs}_Q$ is an enumeration of all inputs for Q

The Construction for $\overline{\mathcal{UH}} \leq \mathcal{PAST}$

Given ordinary program Q. Construct probabilistic program P:

```
continue := 1;
flips := 0;
i := 0;
while (continue ≠ 0){
    Simulate Q for one (further) step on input g_Q(i);
    if (Q has terminated on input g_Q(i)){


      } else {
        flips := flips + 1;
        {continue := 0} [0.5] {continue := 1}
    }
}
```

where $g_Q \colon \mathbb{N} \to Inputs_Q$ is an enumeration of all inputs for Q

The Construction for $\overline{\mathcal{UH}} \leq \mathcal{PAST}$

Given ordinary program Q. Construct probabilistic program P:

```
continue := 1;
flips := 0;
i := 0;
while (continue ≠ 0){
    Simulate Q for one (further) step on input g_Q(i);
    if (Q has terminated on input g_Q(i)){
        i := i + 1; Reset simulation of Q;

      } else {
        flips := flips + 1;
        {continue := 0} [0.5] {continue := 1}
    }
}
```

where $g_Q \colon \mathbb{N} \to \mathit{Inputs}_Q$ is an enumeration of all inputs for Q

The Construction for $\overline{\mathcal{UH}} \leq \mathcal{PAST}$

Given ordinary program Q. Construct probabilistic program P:

```
continue := 1;
flips := 0;
i := 0;
while (continue ≠ 0){
    Simulate Q for one (further) step on input g_Q(i);
    if (Q has terminated on input g_Q(i)){
        i := i + 1; Reset simulation of Q;

      } else {
        flips := flips + 1;
        {continue := 0} [0.5] {continue := 1}
    }
}
```

where $g_Q \colon \mathbb{N} \to \mathit{Inputs}_Q$ is an enumeration of all inputs for Q

The Construction for $\overline{\mathcal{UH}} \leq \mathcal{PAST}$

Given ordinary program Q. Construct probabilistic program P:

```
continue := 1;
flips := 0;
i := 0;
while (continue ≠ 0){
    Simulate Q for one (further) step on input g_Q(i);
    if (Q has terminated on input g_Q(i)){
        i := i + 1; Reset simulation of Q;
        Take 2^flips (meaningless) steps;
      } else {
        flips := flips + 1;
        {continue := 0} [0.5] {continue := 1}
    }
}
```

where $g_Q \colon \mathbb{N} \to \mathit{Inputs}_Q$ is an enumeration of all inputs for Q

The Construction for $\overline{\mathcal{UH}} \leq \mathcal{PAST}$

Given ordinary program Q. Construct probabilistic program P:

```
continue := 1;
flips := 0;
i := 0;
while (continue ≠ 0){
    Simulate Q for one (further) step on input g_Q(i);
    if (Q has terminated on input g_Q(i)){
        i := i + 1; Reset simulation of Q;
        Cheer!;
      } else {
        flips := flips + 1;
        {continue := 0} [0.5] {continue := 1}
    }
}
```

where $g_Q \colon \mathbb{N} \to \mathit{Inputs}_Q$ is an enumeration of all inputs for Q

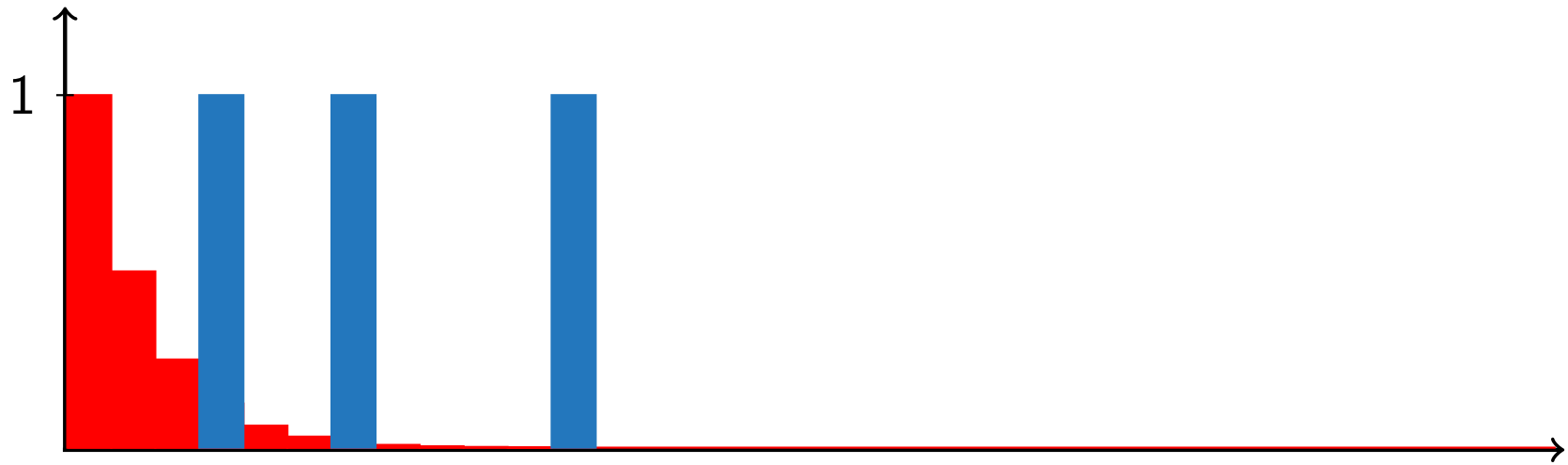
The Construction for $\overline{\mathcal{UH}} \leq \mathcal{PAST}$

Case 1: $Q \in \overline{\mathcal{UH}}$, i.e. there exists smallest i, such that Q does not terminate on i-th input.

The Construction for $\overline{\mathcal{UH}} \leq \mathcal{PAST}$

Case 1: $Q \in \overline{\mathcal{UH}}$, i.e. there exists smallest i, such that Q does not terminate on i-th input.

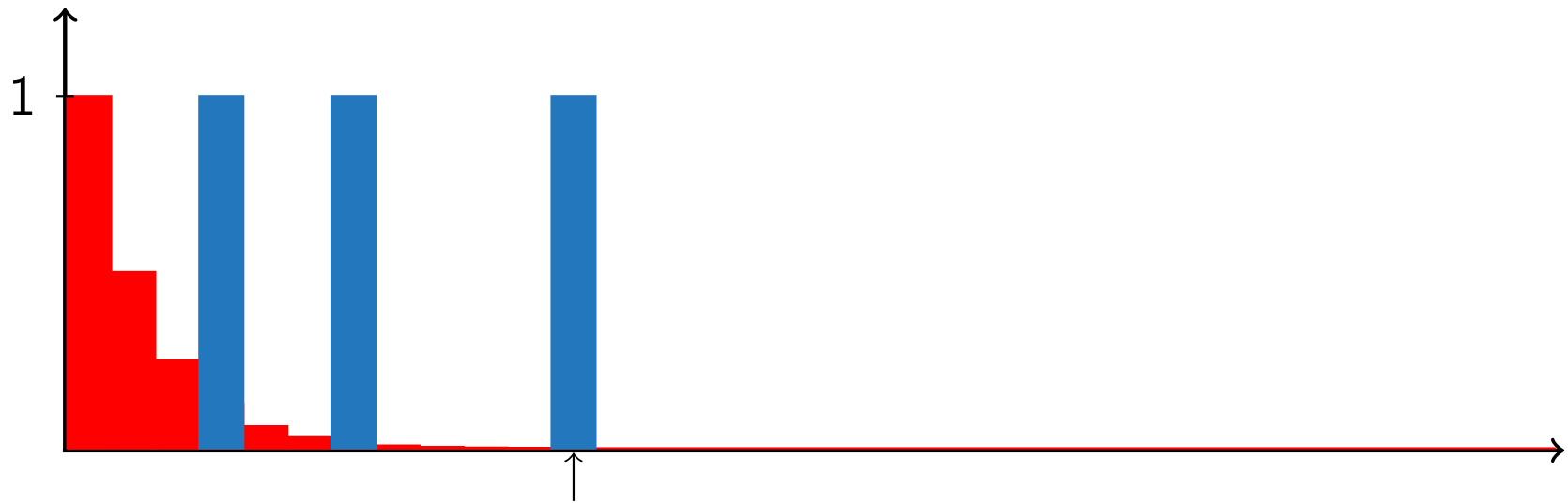
Expected runtime of P (integral over the bars):



The Construction for $\overline{\mathcal{UH}} \leq \mathcal{PAST}$

Case 1: $Q \in \overline{\mathcal{UH}}$, i.e. there exists smallest i, such that Q does not terminate on i-th input.

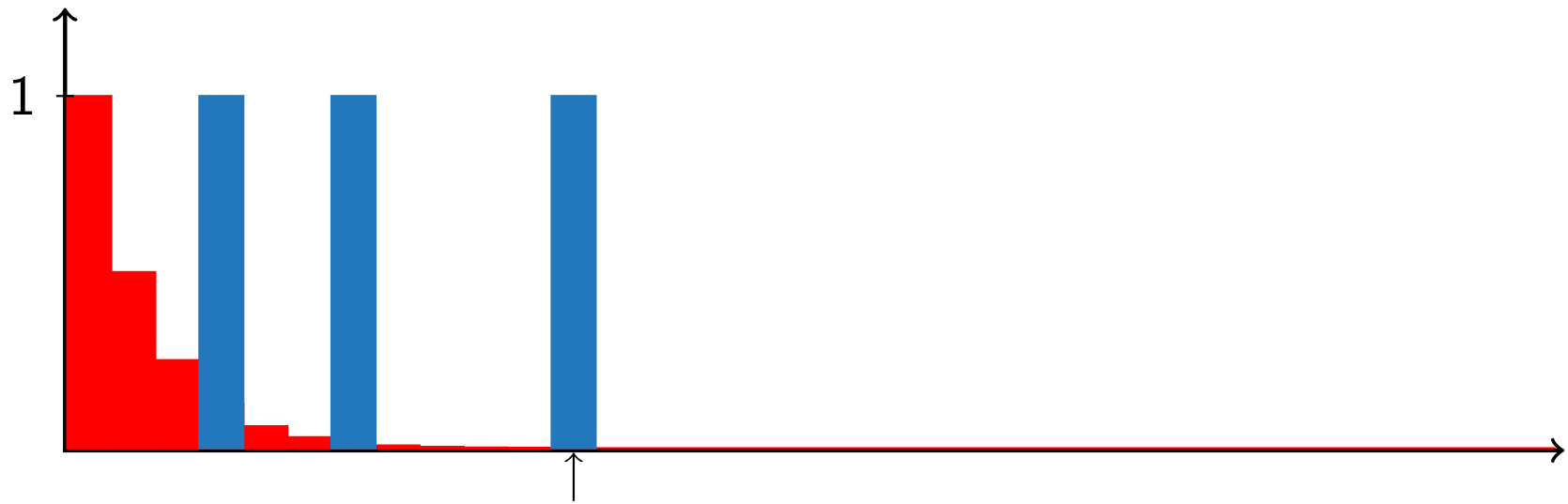
Expected runtime of P (integral over the bars):



The Construction for $\overline{\mathcal{UH}} \leq \mathcal{PAST}$

Case 1: $Q \in \overline{\mathcal{UH}}$, i.e. there exists smallest i, such that Q does not terminate on i-th input.

Expected runtime of P (integral over the bars):



cheering on termination on $(i-1)$-th input

Finite cheering — finite expected runtime!

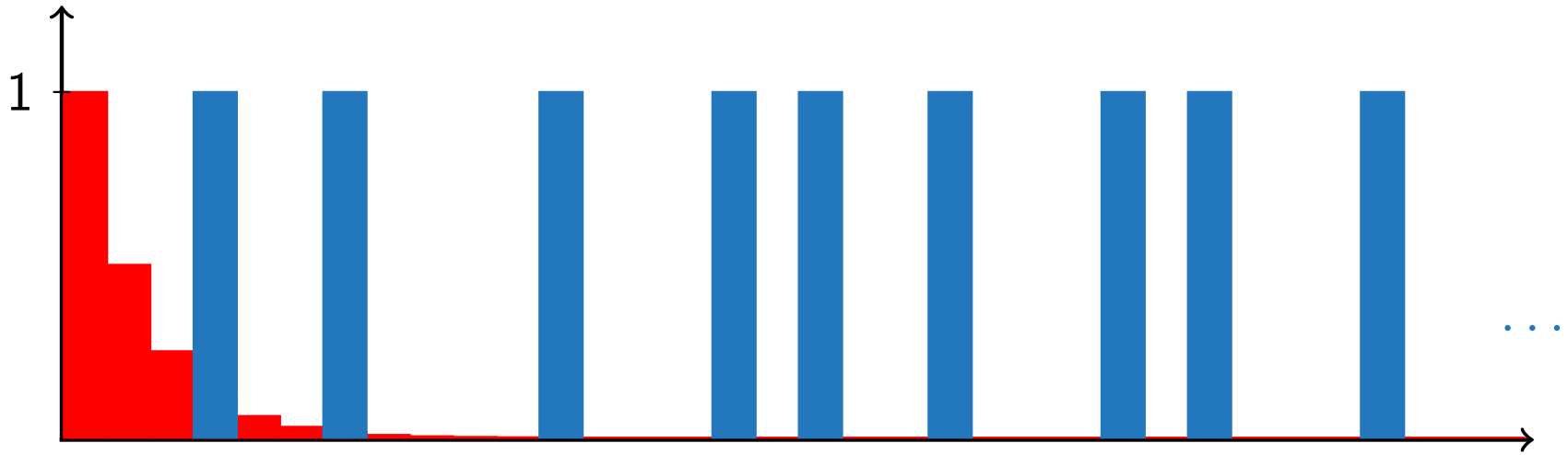
The Construction for $\overline{\mathcal{UH}} \leq \mathcal{PAST}$

Case 2: $Q \notin \overline{\mathcal{UH}}$, i.e. Q terminates on every input.

The Construction for $\overline{\mathcal{UH}} \leq \mathcal{PAST}$

Case 2: $Q \notin \overline{\mathcal{UH}}$, i.e. Q terminates on every input.

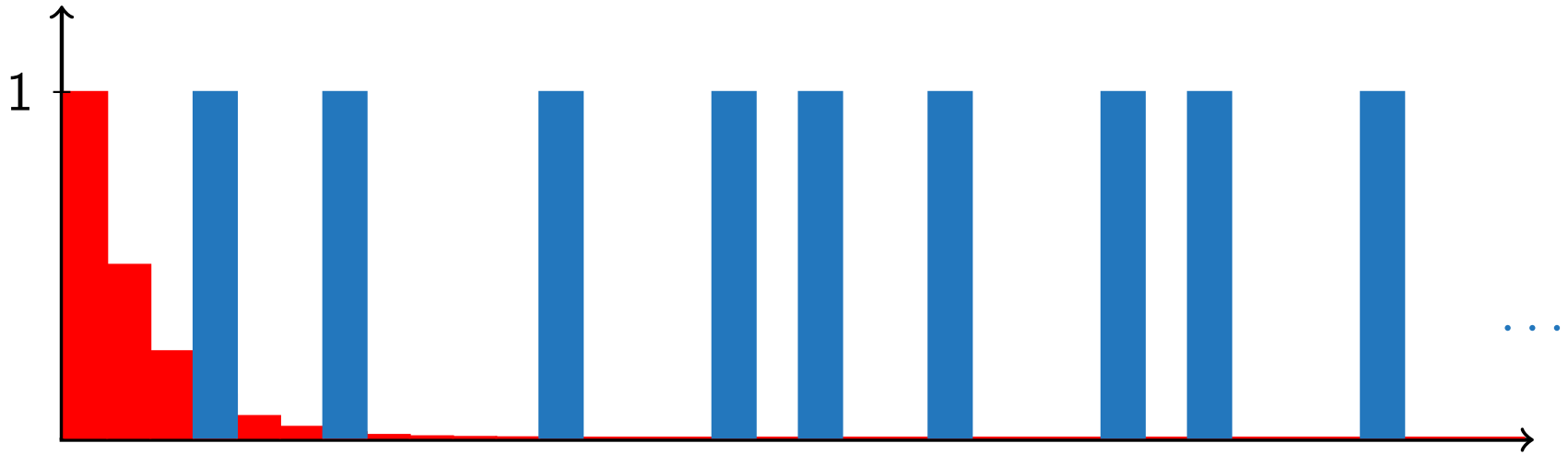
Expected runtime of P (integral over the bars):



The Construction for $\overline{\mathcal{UH}} \leq \mathcal{PAST}$

Case 2: $Q \notin \overline{\mathcal{UH}}$, i.e. Q terminates on every input.

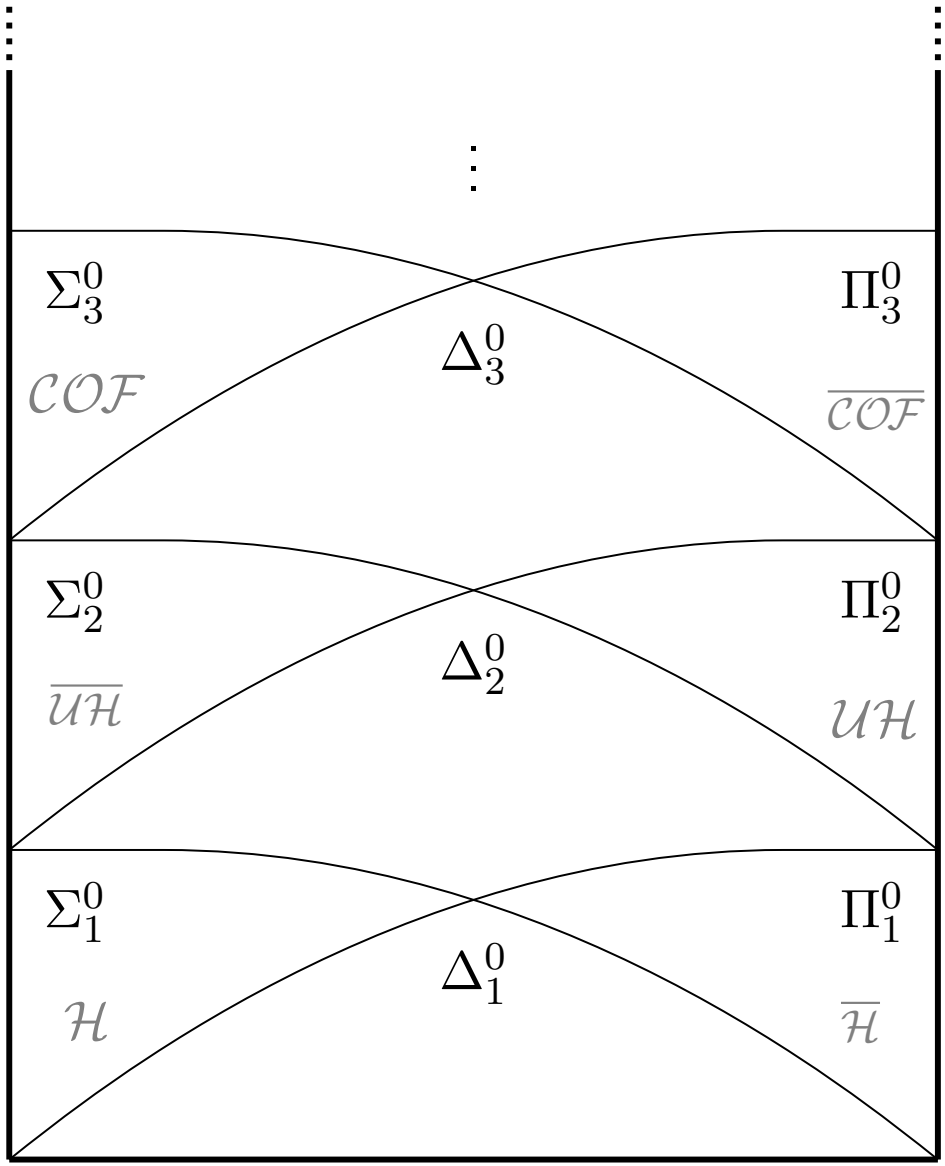
Expected runtime of P (integral over the bars):



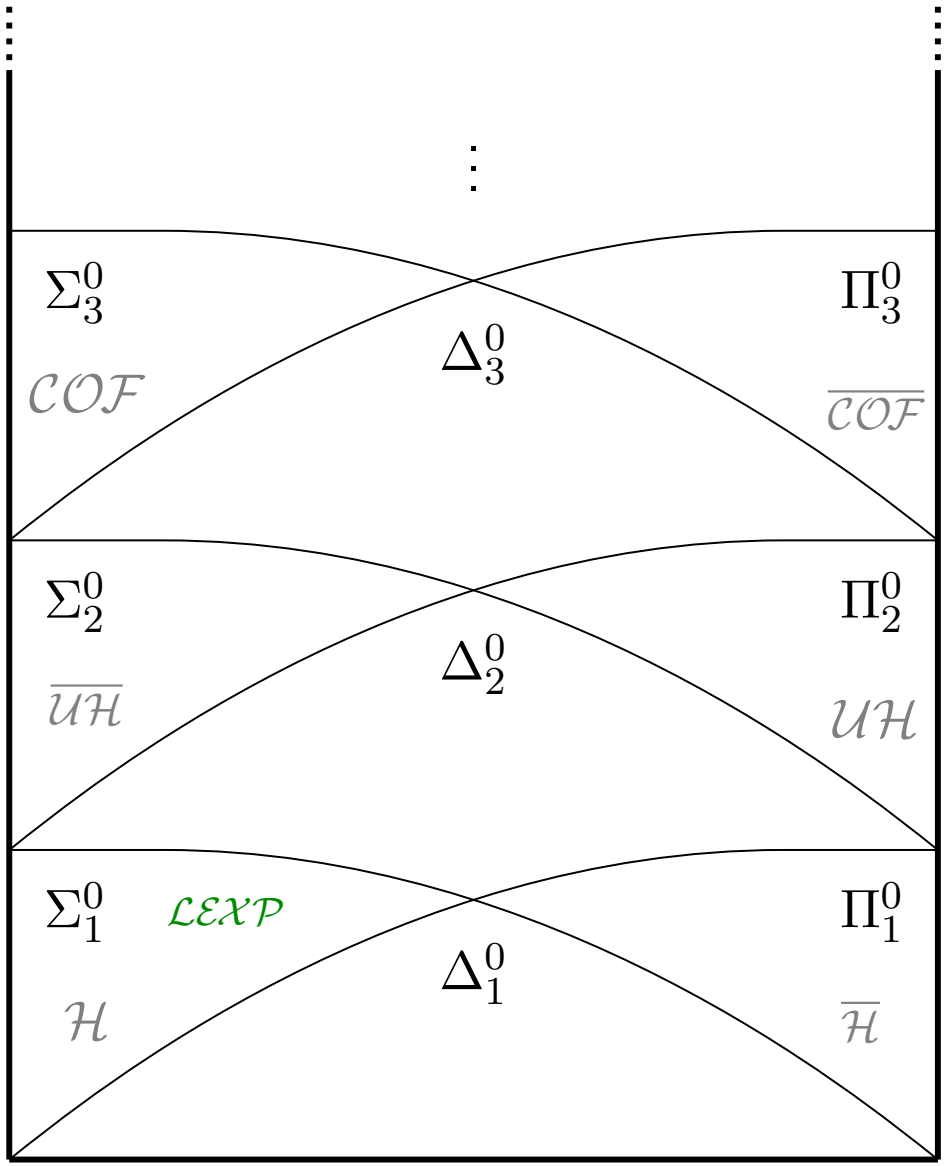
Infinite cheering — infinite expected runtime!

Summary

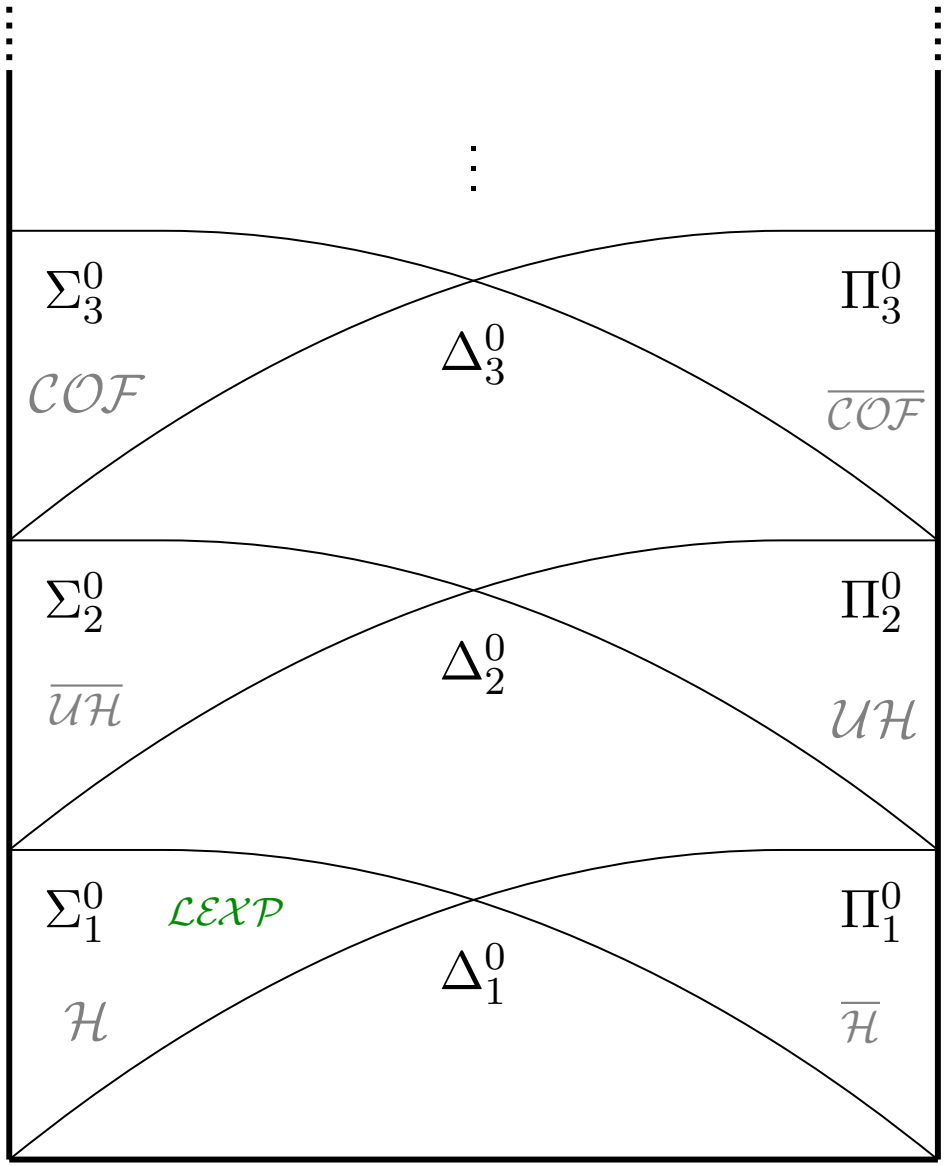
Summary


Σ_3^0
Δ_3^0
Π_3^0
$\mathcal{COF}$
$\overline{\mathcal{COF}}$
Σ_2^0
Δ_2^0
Π_2^0
$\overline{\mathcal{UH}}$
$\mathcal{UH}$
Σ_1^0
Δ_1^0
Π_1^0
$\mathcal{H}$
$\overline{\mathcal{H}}$

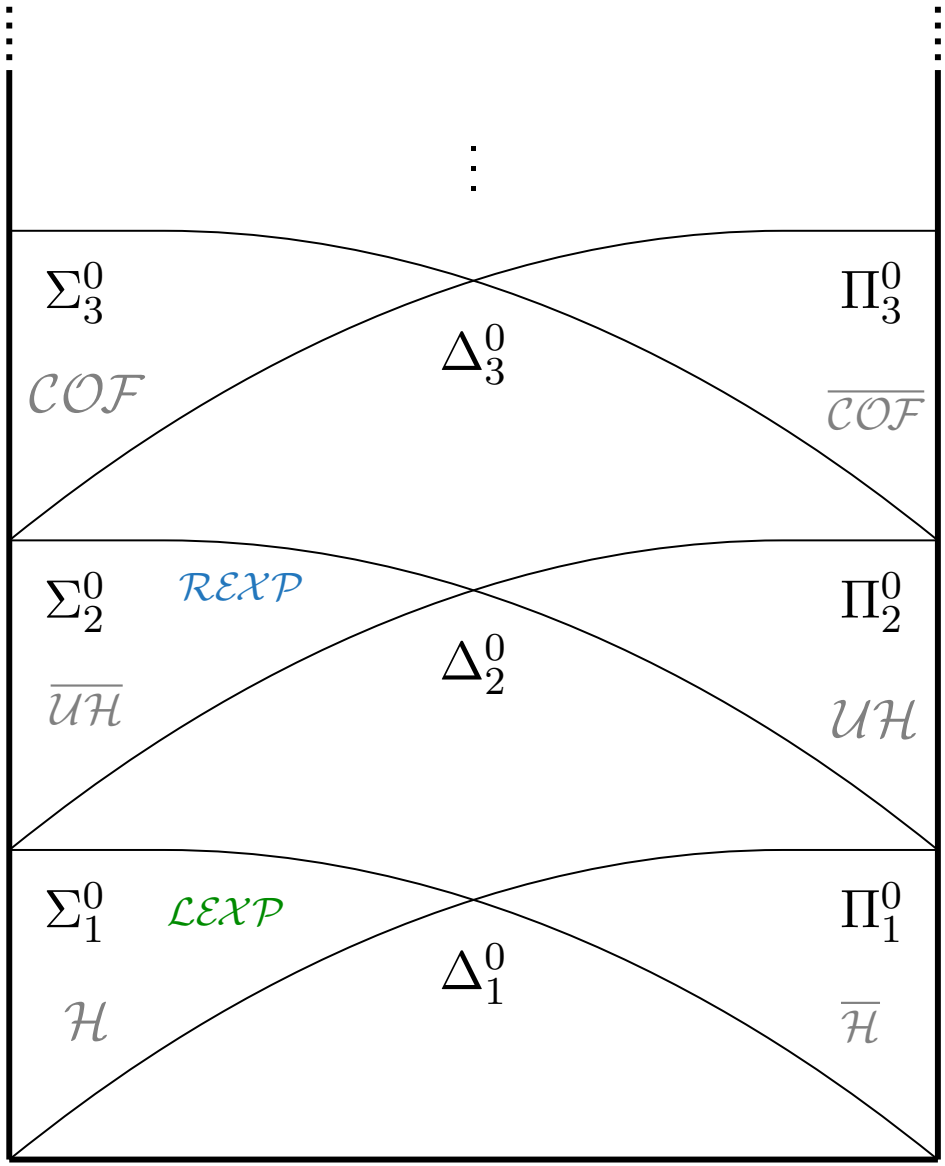
Summary


Σ_3^0
Π_3^0
Δ_3^0
$\mathcal{COF}$
$\overline{\mathcal{COF}}$
Σ_2^0
Π_2^0
Δ_2^0
$\overline{\mathcal{UH}}$
$\mathcal{UH}$
Σ_1^0
$\mathcal{LEXP}$
Π_1^0
Δ_1^0
$\mathcal{H}$
$\overline{\mathcal{H}}$

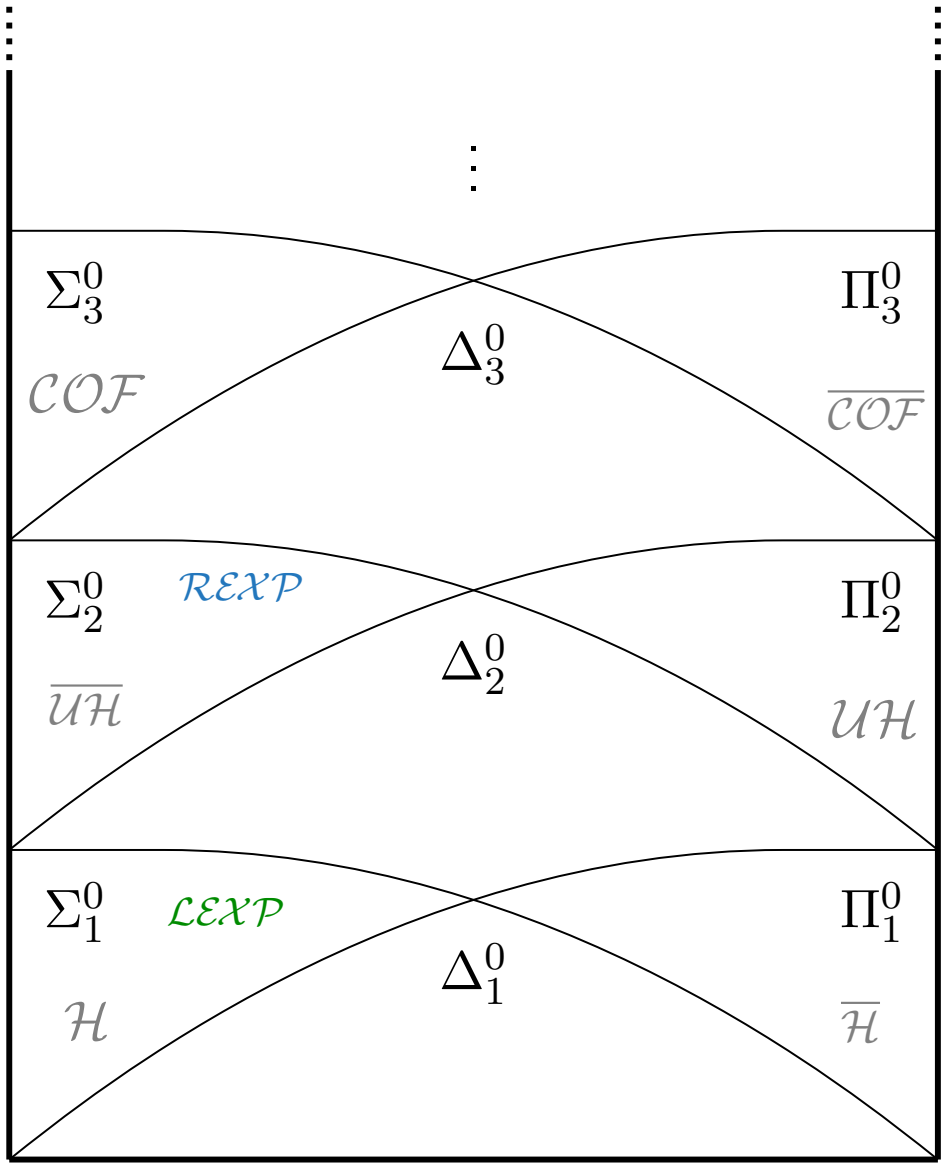
Summary


Σ_3^0
Π_3^0
Δ_3^0
$\mathcal{COF}$
$\overline{\mathcal{COF}}$
Σ_2^0
Π_2^0
Δ_2^0
$\overline{\mathcal{UH}}$
$\mathcal{UH}$
Σ_1^0
$\mathcal{LEXP}$
Π_1^0
Δ_1^0
$\mathcal{H}$
$\overline{\mathcal{H}}$
semi–decidable

Summary


Σ_3^0
Π_3^0
Δ_3^0
$\mathcal{COF}$
$\overline{\mathcal{COF}}$
Σ_2^0
$\mathcal{REXP}$
Π_2^0
Δ_2^0
$\overline{\mathcal{UH}}$
$\mathcal{UH}$
Σ_1^0
$\mathcal{LEXP}$
Π_1^0
Δ_1^0
semi–decidable
$\mathcal{H}$
$\overline{\mathcal{H}}$

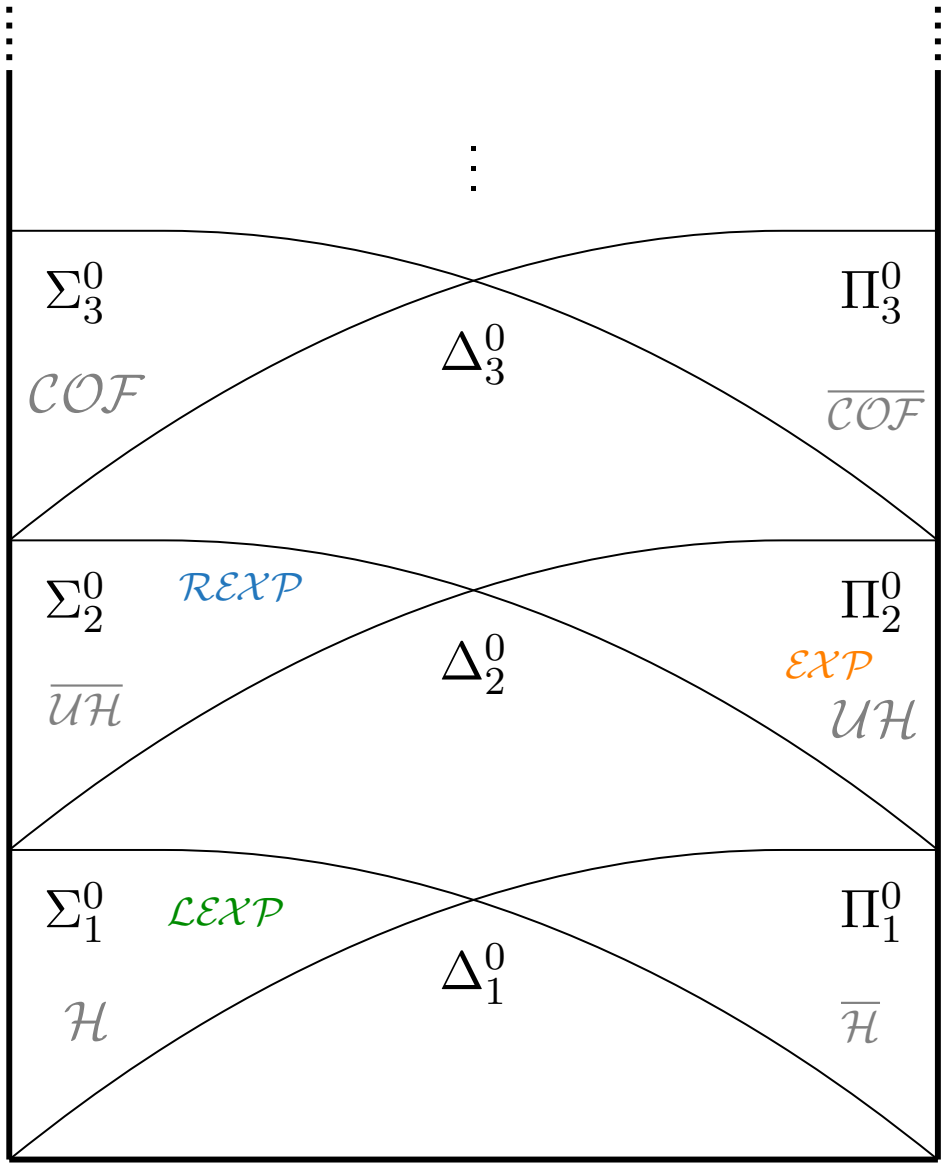
Summary



with access to $\mathcal{H}$–oracle: semi–decidable

semi–decidable

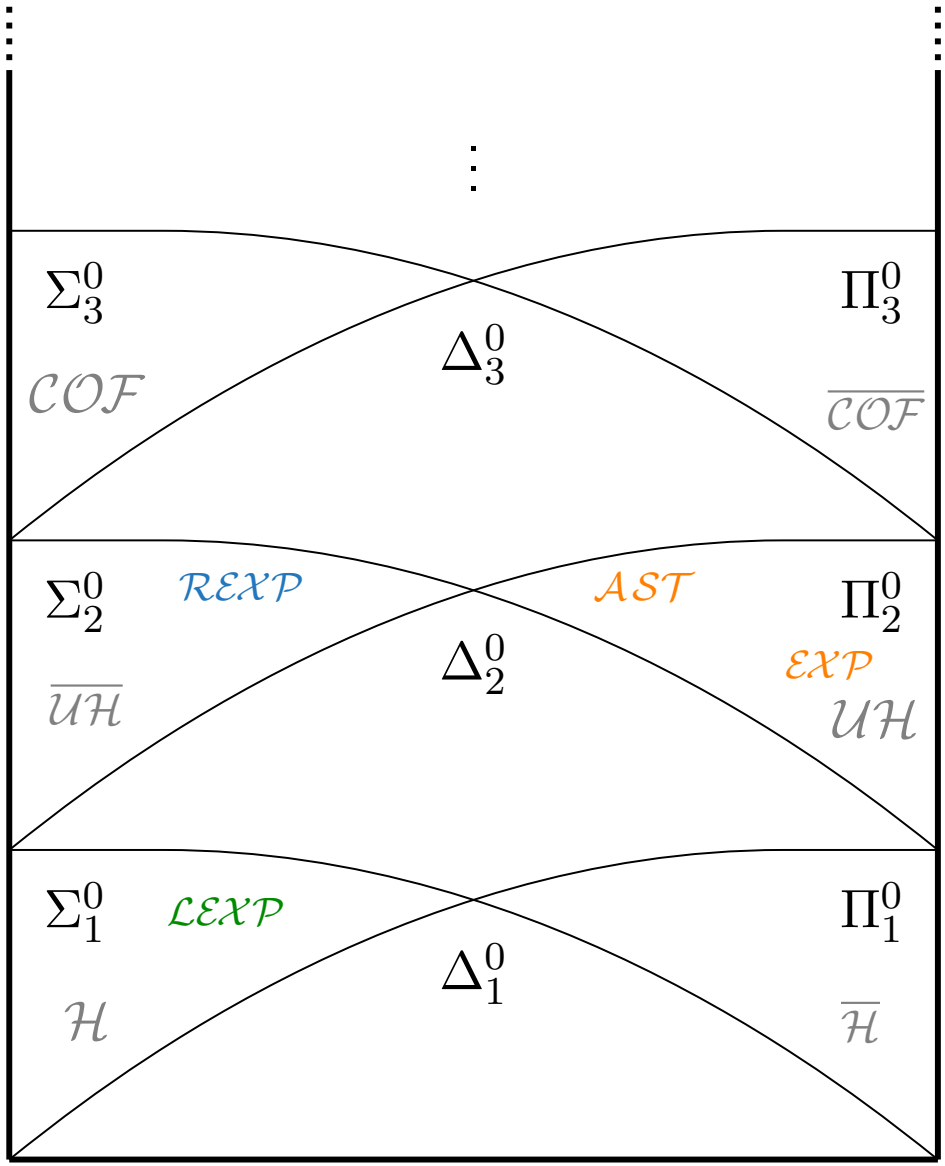
Summary



with access to $\mathcal{H}$–oracle: semi–decidable

semi–decidable

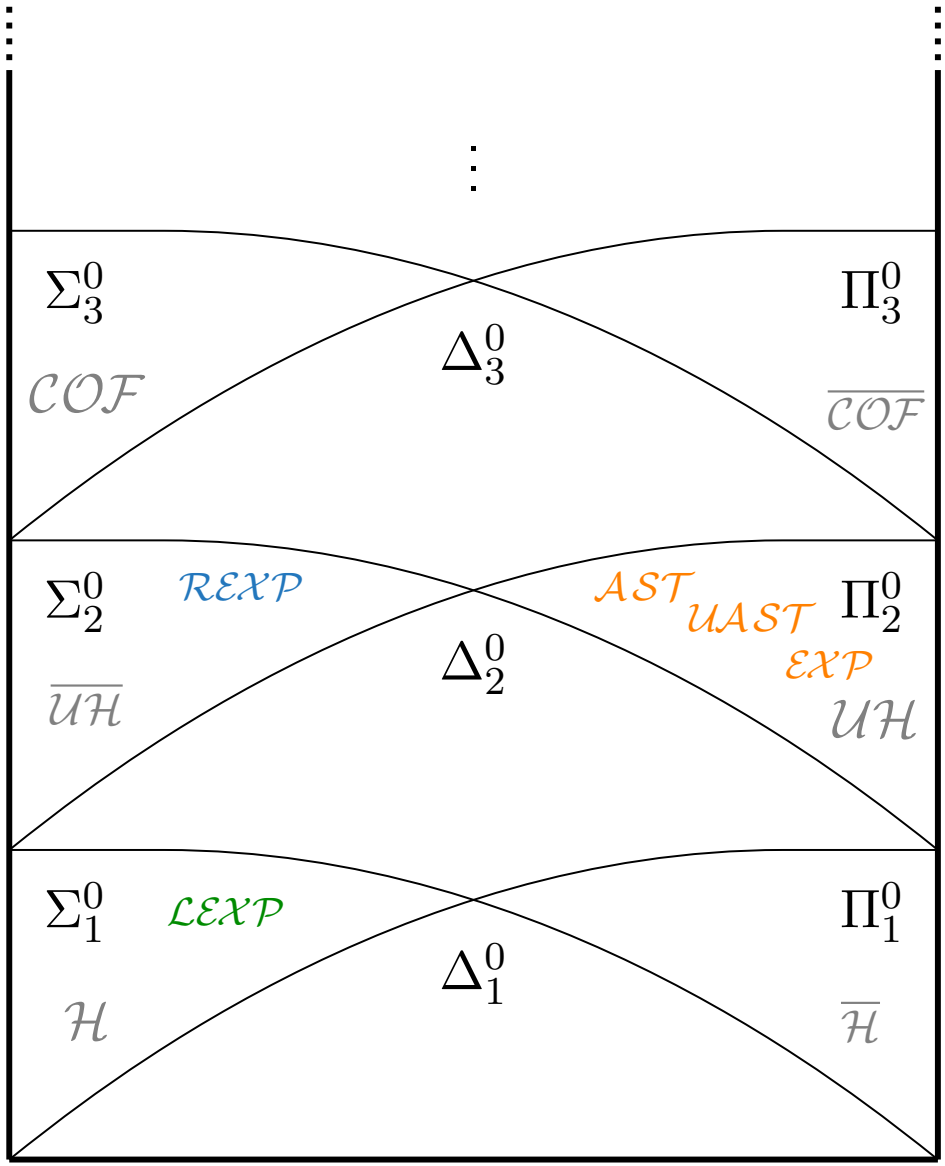
Summary



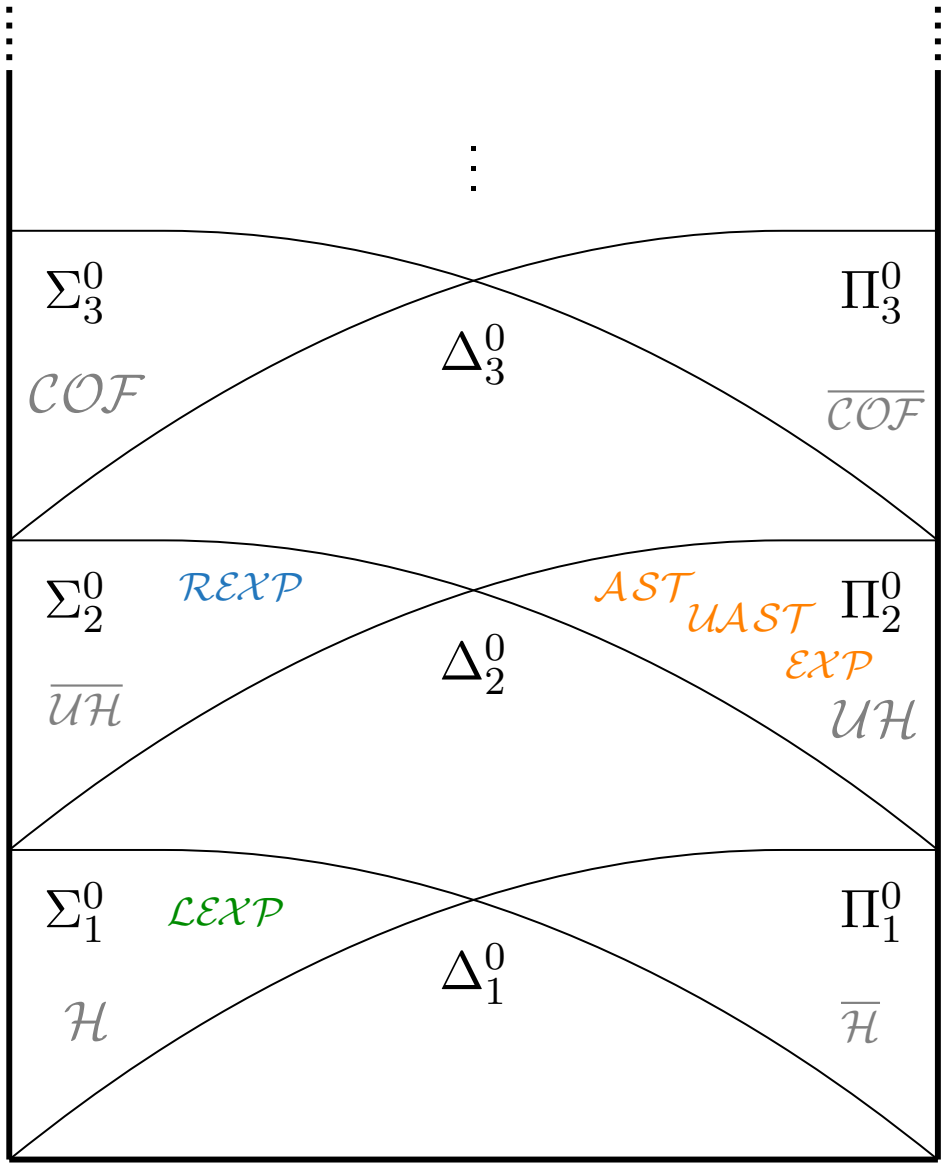
with access to $\mathcal{H}$–oracle: semi–decidable

semi–decidable

Summary


Σ_3^0
Π_3^0
Δ_3^0
$\mathcal{COF}$
$\overline{\mathcal{COF}}$
Σ_2^0
$\mathcal{REXP}$
$\mathcal{AST}$
$\mathcal{UAST}$
Π_2^0
Δ_2^0
$\mathcal{EXP}$
$\overline{\mathcal{UH}}$
$\mathcal{UH}$
with access to $\mathcal{H}$–oracle: semi–decidable
Σ_1^0
$\mathcal{LEXP}$
Π_1^0
Δ_1^0
semi–decidable
$\mathcal{H}$
$\overline{\mathcal{H}}$

Summary

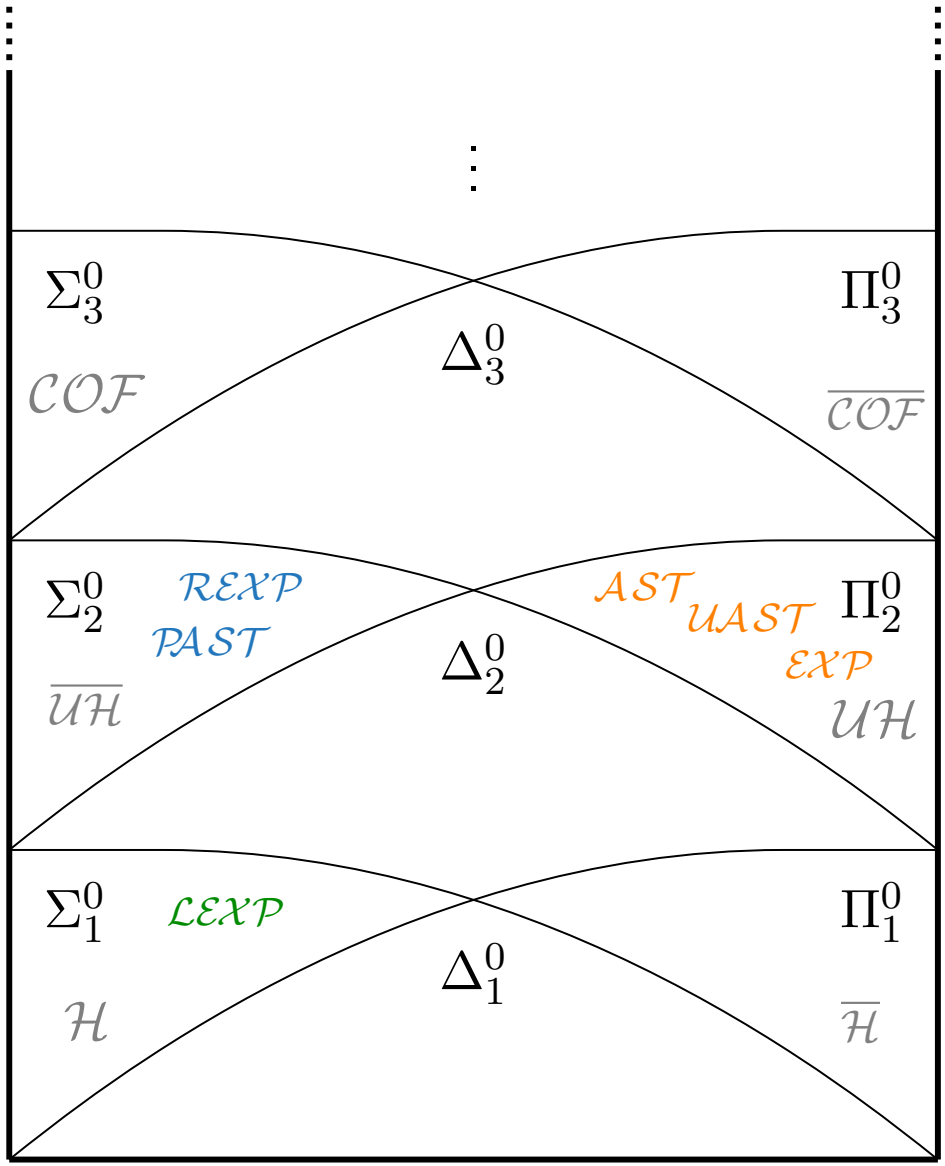


with access to $\mathcal{H}$–oracle: semi–decidable

semi–decidable

not semi–decidable; even with access to $\mathcal{H}$–oracle

Summary

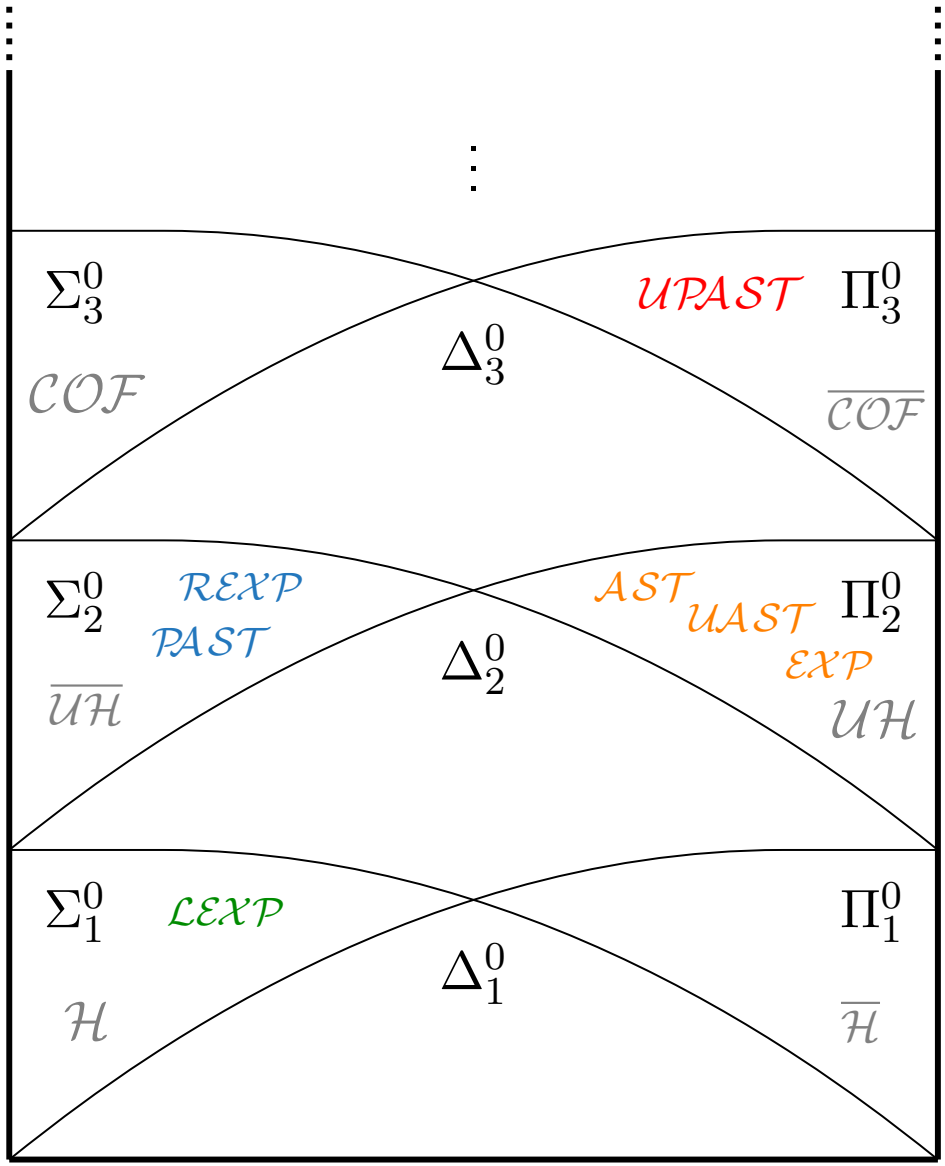


with access to $\mathcal{H}$–oracle: semi–decidable

semi–decidable

not semi–decidable; even with access to $\mathcal{H}$–oracle

Summary

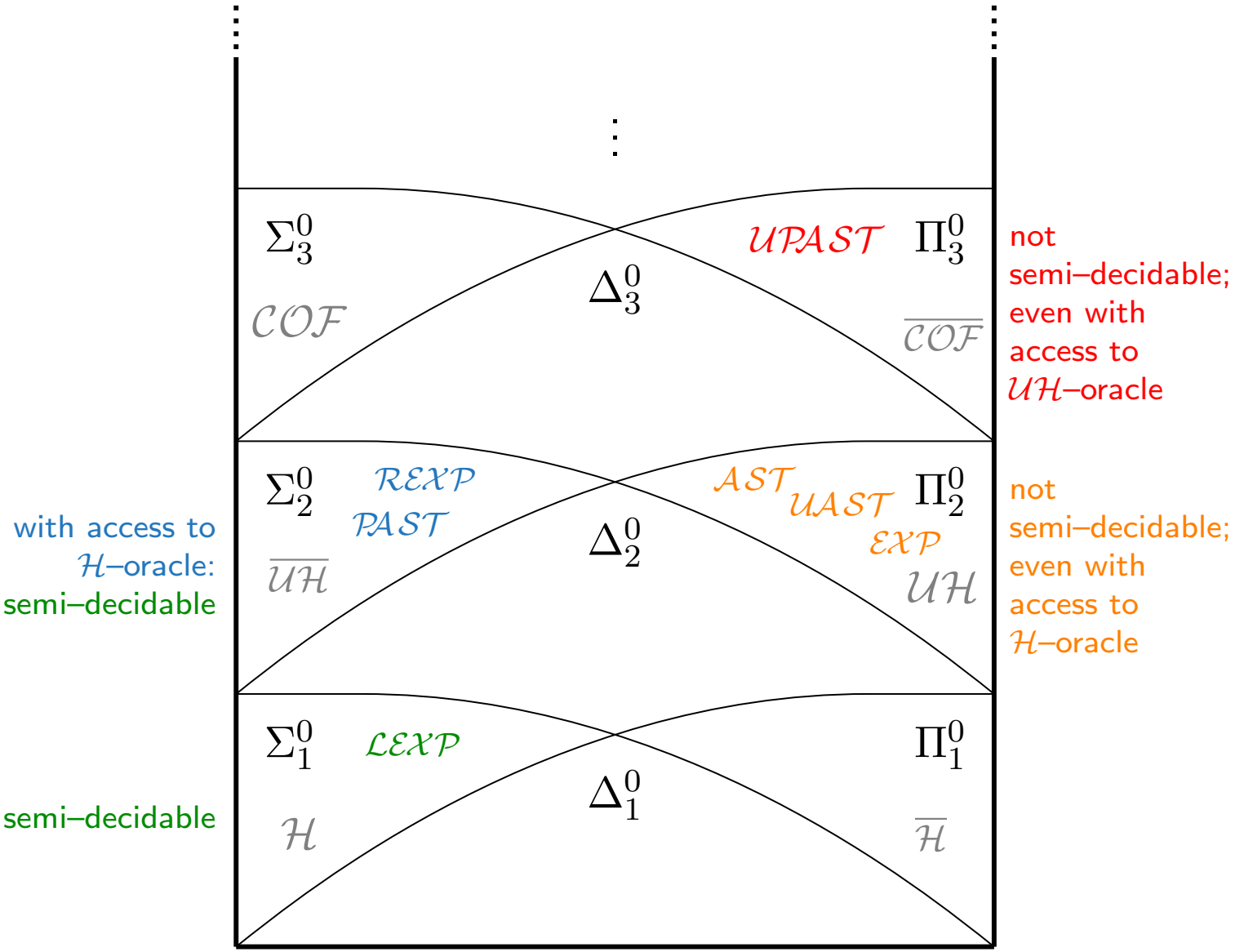


with access to $\mathcal{H}$–oracle: semi–decidable

semi–decidable

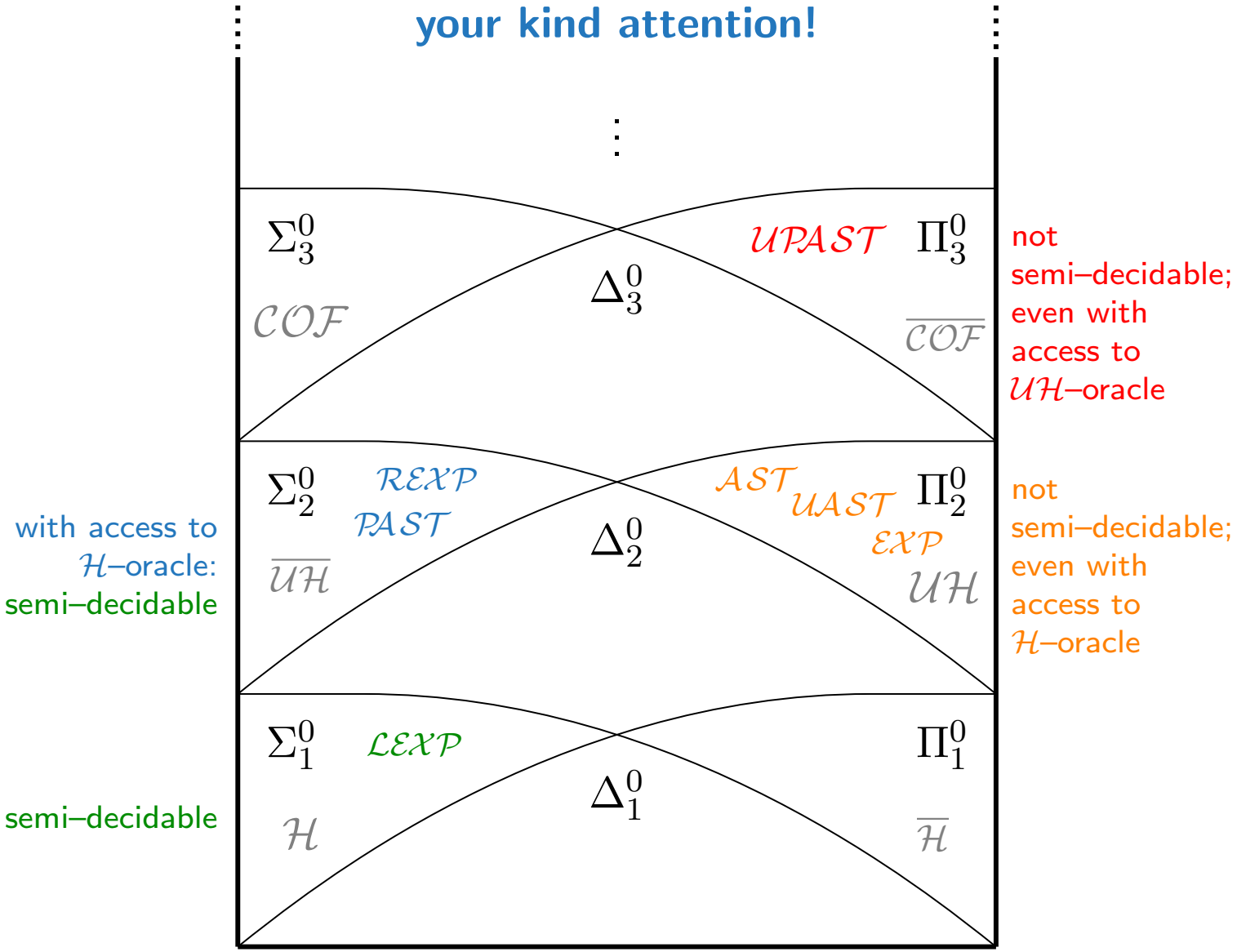
not semi–decidable; even with access to $\mathcal{H}$–oracle

Summary


Σ_3^0
$\mathcal{UPAST}$
Π_3^0
Δ_3^0
$\mathcal{COF}$
$\overline{\mathcal{COF}}$
not semi–decidable; even with access to $\mathcal{UH}$–oracle
Σ_2^0
$\mathcal{REXP}$
$\mathcal{PAST}$
$\mathcal{AST}$
$\mathcal{UAST}$
Π_2^0
$\mathcal{EXP}$
Δ_2^0
$\overline{\mathcal{UH}}$
$\mathcal{UH}$
with access to $\mathcal{H}$–oracle: semi–decidable
not semi–decidable; even with access to $\mathcal{H}$–oracle
Σ_1^0
$\mathcal{LEXP}$
Π_1^0
Δ_1^0
$\mathcal{H}$
$\overline{\mathcal{H}}$
semi–decidable

Summary

Thank you for your kind attention!



Proof of $\mathcal{LEXP} \in \Sigma_1^0$

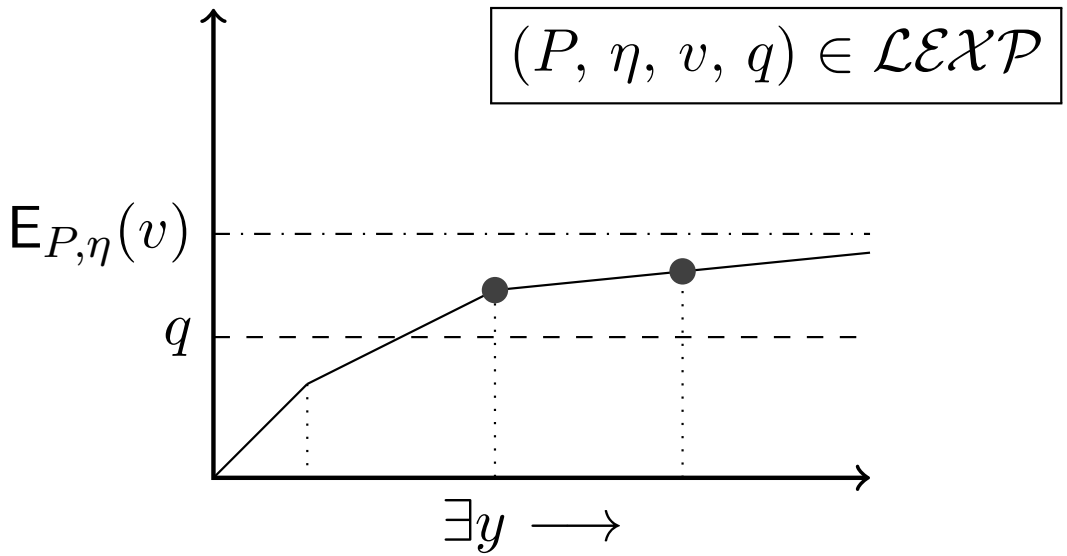
$\mathcal{LEXP}$ is defined by the following formula:

$$\exists y \colon q < \sum_{k=0}^{y} \sum_{w \in \{L,\, R\}^{\leq k}} \wp\Big(\mathsf{T}_k\big(\sigma_{P,\eta},\, w\big),\, v\Big)$$

Proof of $\mathcal{LEXP} \in \Sigma_1^0$

$\mathcal{LEXP}$ is defined by the following formula:

$$\exists y\colon q < \sum_{k=0}^{y} \sum_{w \in \{L,\, R\}^{\leq k}} \wp\Big(\mathsf{T}_k\big(\sigma_{P,\eta},\, w\big),\, v\Big)$$



Hardness of Deciding Universal Positive A.–S. Termination

Hardness of Deciding Universal Positive A.–S. Termination

$\overline{\mathcal{COF}}$ — A Very Hard Problem

The problem $\mathcal{COF}$, defined by

$$\mathcal{COF} \;=\; \left\{ Q \in \text{ordProg} \;\middle|\; \{\eta \mid Q \text{ halts on input } \eta\} \text{ is cofinite} \right\},$$

is Σ_3^0–complete.

Hardness of Deciding Universal Positive A.–S. Termination

$\overline{\mathcal{COF}}$ — A Very Hard Problem

The problem $\mathcal{COF}$, defined by

$$\mathcal{COF} \;=\; \left\{ Q \in \text{ordProg} \;\middle|\; \{\eta \mid Q \text{ halts on input } \eta\} \text{ is cofinite} \right\},$$

is Σ^0_3–complete. The complement $\overline{\mathcal{COF}}$ is Π^0_3–complete.

Hardness of Deciding Universal Positive A.–S. Termination

$\overline{\mathcal{COF}}$ — A Very Hard Problem

The problem $\mathcal{COF}$, defined by

$$\mathcal{COF} \;=\; \left\{ Q \in \text{ordProg} \;\middle|\; \{\eta \mid Q \text{ halts on input } \eta\} \text{ is cofinite} \right\},$$

is Σ_3^0–complete. The complement $\overline{\mathcal{COF}}$ is Π_3^0–complete.

Theorem

$\mathcal{UPAST}$ is Π_3^0–complete.

Hardness of Deciding Universal Positive A.–S. Termination

$\overline{\mathcal{COF}}$ — A Very Hard Problem

The problem $\mathcal{COF}$, defined by

$$\mathcal{COF} \;=\; \left\{ Q \in \text{ordProg} \;\middle|\; \{\eta \mid Q \text{ halts on input } \eta\} \text{ is cofinite} \right\},$$

is Σ_3^0–complete. The complement $\overline{\mathcal{COF}}$ is Π_3^0–complete.

Theorem

$\mathcal{UPAST}$ is Π_3^0–complete.

Proof: $\mathcal{UPAST} \in \Pi_3^0$ as $\mathcal{PAST} \in \Sigma_2^0$ and $\mathcal{UPAST}$ is in universal closure of $\mathcal{PAST}$.

Hardness of Deciding Universal Positive A.–S. Termination

$\overline{\mathcal{COF}}$ — A Very Hard Problem

The problem $\mathcal{COF}$, defined by

$$\mathcal{COF} \;=\; \big\{Q \in \text{ordProg} \;\big|\; \{\eta \mid Q \text{ halts on input } \eta\} \text{ is cofinite}\big\}\;,$$

is Σ_3^0–complete. The complement $\overline{\mathcal{COF}}$ is Π_3^0–complete.

Theorem

$\mathcal{UPAST}$ is Π_3^0–complete.

Proof: $\mathcal{UPAST} \in \Pi_3^0$ as $\mathcal{PAST} \in \Sigma_2^0$ and $\mathcal{UPAST}$ is in universal closure of $\mathcal{PAST}$. $\mathcal{UPAST}$ is Π_3^0–hard, as $\overline{\mathcal{COF}} \leq_{\mathsf{m}} \mathcal{UPAST}$.