Analyzing Expected Outcomes and Almost–Sure Termination of Probabilistic Programs is Hard

Benjamin Kaminski

Rheinisch–Westfälische Technische Hochschule Aachen
Lehrstuhl für Informatik 2

28.5.2014

Motivation

- Probabilistic Programs are like ordinary programs, except:

Motivation

- Probabilistic Programs are like ordinary programs, except:
  - Allow for random choice on how to continue the execution
  - Random choice is done with some specified probability p

Motivation

- Probabilistic Programs are like ordinary programs, except:
  - Allow for random choice on how to continue the execution
  - Random choice is done with some specified probability p
- **Analysis problems** we consider:

Motivation

- Probabilistic Programs are like ordinary programs, except:
  - Allow for random choice on how to continue the execution
  - Random choice is done with some specified probability p
- **Analysis problems** we consider:
  - Determine the value of a variable after program execution

Motivation

- Probabilistic Programs are like ordinary programs, except:
  - Allow for random choice on how to continue the execution
  - Random choice is done with some specified probability p
- **Analysis problems** we consider:
  - Determine the value of a variable after program execution
  - Decide whether the program terminates

Motivation

- Probabilistic Programs are like ordinary programs, except:
  - Allow for random choice on how to continue the execution
  - Random choice is done with some specified probability p
- **Analysis problems** we consider:
  - Determine the value of a variable after program execution
    - Determine expected values
  - Decide whether the program terminates

Motivation

- Probabilistic Programs are like ordinary programs, except:
  - Allow for random choice on how to continue the execution
  - Random choice is done with some specified probability p
- **Analysis problems** we consider:
  - Determine the value of a variable after program execution
    - Determine expected values (expected outcomes)
  - Decide whether the program terminates

Motivation

- Probabilistic Programs are like ordinary programs, except:
  - Allow for random choice on how to continue the execution
  - Random choice is done with some specified probability p
- **Analysis problems** we consider:
  - Determine the value of a variable after program execution
    - Determine expected values (expected outcomes)
  - Decide whether the program terminates
    - Decide almost–sure termination!

Motivation

- Probabilistic Programs are like ordinary programs, except:
  - Allow for random choice on how to continue the execution
  - Random choice is done with some specified probability p
- **Analysis problems** we consider:
  - Determine the value of a variable after program execution
    - Determine expected values (expected outcomes)
  - Decide whether the program terminates
    - Decide almost–sure termination!

How hard is it to solve these analysis problems?

Outline

Outline

Outline

1. The arithmetical hierarchy
2. Probabilistic programs
   - Syntax
   - Semantics
   - Notations

Outline

1. The arithmetical hierarchy
2. Probabilistic programs
   - Syntax
   - Semantics
   - Notations
3. Hardness of determining expected outcomes

Outline

1. The arithmetical hierarchy
2. Probabilistic programs
   - Syntax
   - Semantics
   - Notations
3. Hardness of determining expected outcomes
4. Hardness of deciding almost–sure termination

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

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

- Class Π_n^0 is defined as

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

The Arithmetical Hierarchy

- Class Σ_n^0 is defined as

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

- Class Π_n^0 is defined as

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

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

The Arithmetical Hierarchy Revisited

The Arithmetical Hierarchy Revisited

- Reconsider class Σ_n^0:

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

The Arithmetical Hierarchy Revisited

- Reconsider class Σ_n^0:

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

- For our convenience: domain of the variables shall be $\mathbb{Q}^+$

The Arithmetical Hierarchy Revisited

- Reconsider class Σ_n^0:

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

- For our convenience: domain of the variables shall be $\mathbb{Q}^+$

The Arithmetical Hierarchy Revisited

- Reconsider class Σ_n^0:

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

- For our convenience: domain of the variables shall be $\mathbb{Q}^+$
- Consecutive quantifiers of the same type can be contracted:

The Arithmetical Hierarchy Revisited

- Reconsider class Σ_n^0:

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

- For our convenience: domain of the variables shall be $\mathbb{Q}^+$
- Consecutive quantifiers of the same type can be contracted:

$$A = \{ \vec{x} \mid \exists y_1 \, \exists y_2 \, \exists y_3 \, \forall y_4 \, \forall y_5 : (x, y_1, \ldots, y_5) \in R \}$$

The Arithmetical Hierarchy Revisited

- Reconsider class Σ_n^0:

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

- For our convenience: domain of the variables shall be $\mathbb{Q}^+$
- Consecutive quantifiers of the same type can be contracted:

$$A = \{ \vec{x} \mid \exists y_1 \, \exists y_2 \, \exists y_3 \, \forall y_4 \, \forall y_5 : (x, y_1, \ldots, y_5) \in R \}$$

The Arithmetical Hierarchy Revisited

- Reconsider class Σ_n^0:

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

- For our convenience: domain of the variables shall be $\mathbb{Q}^+$
- Consecutive quantifiers of the same type can be contracted:

$$\begin{aligned} A &= \{ \vec{x} \mid \exists y_1 \, \exists y_2 \, \exists y_3 \, \forall y_4 \, \forall y_5 : (x, y_1, \ldots, y_5) \in R \} \\ &= \{ \vec{x} \mid \exists y_{1,2,3} \, \forall y_{4,5} : (x, y_{1,2,3}, y_{4,5}) \in R' \} \end{aligned}$$

The Arithmetical Hierarchy Revisited

- Reconsider class Σ_n^0:

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

- For our convenience: domain of the variables shall be $\mathbb{Q}^+$
- Consecutive quantifiers of the same type can be contracted:

$$\begin{aligned} A &= \{ \vec{x} \mid \exists y_1 \, \exists y_2 \, \exists y_3 \, \forall y_4 \, \forall y_5 : (x, y_1, \ldots, y_5) \in R \} \\ &= \{ \vec{x} \mid \exists y_{1,2,3} \, \forall y_{4,5} : (x, y_{1,2,3}, y_{4,5}) \in R' \} \\ &\in \Sigma_2^0 \end{aligned}$$

The Arithmetical Hierarchy Revisited

- Reconsider class Σ_n^0:

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

- For our convenience: domain of the variables shall be $\mathbb{Q}^+$
- Consecutive quantifiers of the same type can be contracted:

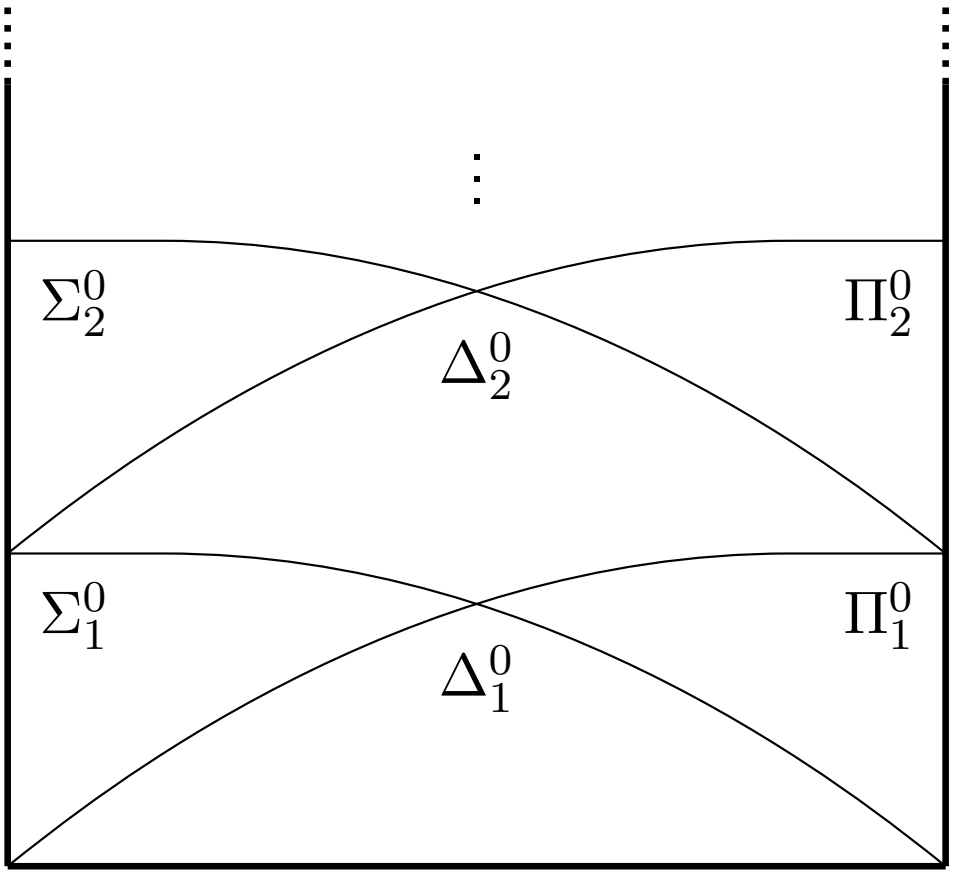
$$\begin{aligned} A &= \{ \vec{x} \mid \exists y_1 \, \exists y_2 \, \exists y_3 \, \forall y_4 \, \forall y_5 : (x, y_1, \ldots, y_5) \in R \} \\ &= \{ \vec{x} \mid \exists y_{1,2,3} \, \forall y_{4,5} : (x, y_{1,2,3}, y_{4,5}) \in R' \} \\ &\in \Sigma_2^0 \end{aligned}$$

- n is really about the number of alternating quantifiers

The Arithmetical Hierarchy — The Bigger Picture

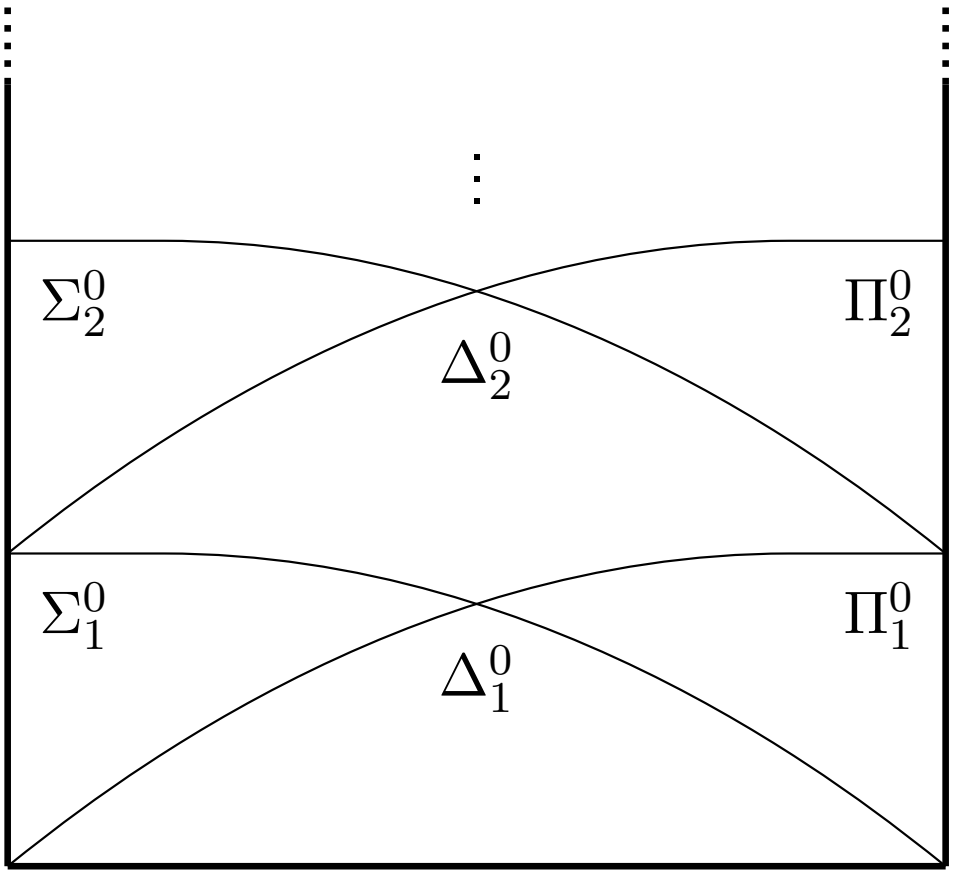
The Arithmetical Hierarchy — The Bigger Picture

- The following inclusion diagram holds:



The Arithmetical Hierarchy — The Bigger Picture

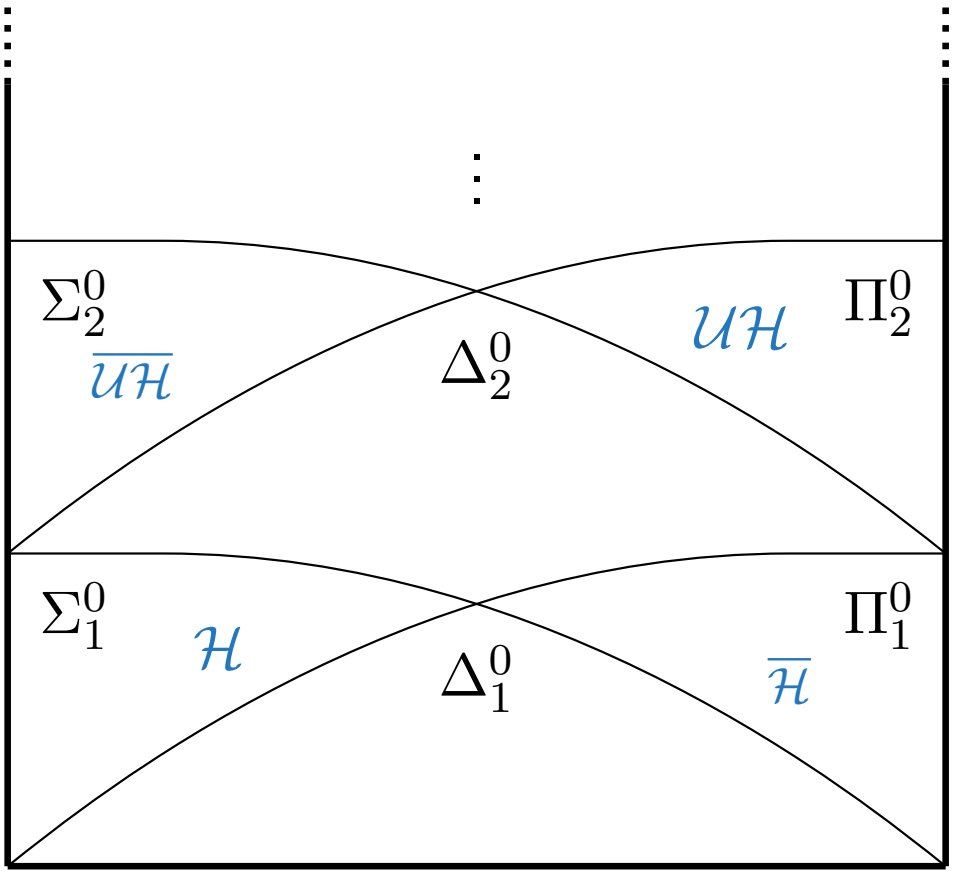
- The following inclusion diagram holds:



- All inclusions are strict

The Arithmetical Hierarchy — The Bigger Picture

- The following inclusion diagram holds:



- All inclusions are strict

Probabilistic Programs — Syntax

- Syntax of probabilistic programs:

Probabilistic Programs — Syntax

- Syntax of probabilistic programs:
  - Assignment: var := $expr$

Probabilistic Programs — Syntax

- Syntax of probabilistic programs:
  - Assignment: var := $expr$
  - Concatenation: P_1 ; P_2

Probabilistic Programs — Syntax

- Syntax of probabilistic programs:
  - Assignment: var `:=` $expr$
  - Concatenation: P_1 `;` P_2
  - While–loop: `WHILE (`$bexpr$`) {`P`}`

Probabilistic Programs — Syntax

- Syntax of probabilistic programs:
  - Assignment: var `:=` $expr$
  - Concatenation: P_1 `;` P_2
  - While–loop: `WHILE (`$bexpr$`) {`P`}`
  - Probabilistic choice: $\{P_1\}\ [p]\ \{P_2\}$, for $p \in [0, 1] \subseteq \mathbb{Q}$

Probabilistic Programs — Syntax

- Syntax of probabilistic programs:
  - Assignment: var `:=` $expr$
  - Concatenation: P_1 `;` P_2
  - While–loop: `WHILE` `(`$bexpr$`)` $\{P\}$
  - Probabilistic choice: $\{P_1\}\ [p]\ \{P_2\}$, for $p \in [0, 1] \subseteq \mathbb{Q}$
- Denote the set of probabilistic programs by Prog.

Probabilistic Programs — Syntax

- Syntax of probabilistic programs:
  - Assignment: var `:=` $expr$
  - Concatenation: P_1 ; P_2
  - While–loop: `WHILE` $(bexpr)$ $\{P\}$
  - Probabilistic choice: $\{P_1\}\ [p]\ \{P_2\}$, for $p \in [0, 1] \subseteq \mathbb{Q}$
- Denote the set of probabilistic programs by Prog.
- Denote the set of ordinary programs (programs that contain no probabilistic choice) by ordProg.

Probabilistic Programs — Semantics

- Set of variable valuations: $\mathbb{V} = \{\eta \mid \eta \colon \mathsf{Var} \to \mathbb{Q}^+\}$

Probabilistic Programs — Semantics

- Set of variable valuations: $\mathbb{V} = \{\eta \mid \eta \colon \mathsf{Var} \to \mathbb{Q}^+\}$
- Set of program states: $\mathbb{S} = \big(\mathsf{Prog} \cup \{\downarrow\}\big) \times \mathbb{V} \times I \times \{L, R\}^*$, for $I = [0, 1] \subseteq \mathbb{Q}^+$

Probabilistic Programs — Semantics

- Set of variable valuations: $\mathbb{V} = \{\eta \mid \eta\colon \mathsf{Var} \to \mathbb{Q}^+\}$
- Set of program states: $\mathbb{S} = \big(\mathsf{Prog} \cup \{\downarrow\}\big) \times \mathbb{V} \times I \times \{L, R\}^*$, for $I = [0, 1] \subseteq \mathbb{Q}^+$
- A closer look at a program state $\sigma = \langle P, \eta, a, \theta \rangle \in \mathbb{S}$:

Probabilistic Programs — Semantics

- Set of variable valuations: $\mathbb{V} = \{\eta \mid \eta\colon \mathsf{Var} \to \mathbb{Q}^+\}$
- Set of program states: $\mathbb{S} = \big(\mathsf{Prog} \cup \{\downarrow\}\big) \times \mathbb{V} \times I \times \{L, R\}^*$, for $I = [0, 1] \subseteq \mathbb{Q}^+$
- A closer look at a program state $\sigma = \langle P, \eta, a, \theta \rangle \in \mathbb{S}$:
  - P is the program remaining to be executed

Probabilistic Programs — Semantics

- Set of variable valuations: $\mathbb{V} = \{\eta \mid \eta\colon \mathsf{Var} \to \mathbb{Q}^+\}$
- Set of program states: $\mathbb{S} = \big(\mathsf{Prog} \cup \{\downarrow\}\big) \times \mathbb{V} \times I \times \{L, R\}^*$, for $I = [0, 1] \subseteq \mathbb{Q}^+$
- A closer look at a program state $\sigma = \langle P, \eta, a, \theta \rangle \in \mathbb{S}$:
  - P is the program remaining to be executed
    - If $P = \downarrow$ then σ is a terminal state

Probabilistic Programs — Semantics

- Set of variable valuations: $\mathbb{V} = \{\eta \mid \eta\colon \mathsf{Var} \to \mathbb{Q}^+\}$
- Set of program states: $\mathbb{S} = \big(\mathsf{Prog} \cup \{\downarrow\}\big) \times \mathbb{V} \times I \times \{L, R\}^*$, for $I = [0, 1] \subseteq \mathbb{Q}^+$
- A closer look at a program state $\sigma = \langle P, \eta, a, \theta \rangle \in \mathbb{S}$:
  - P is the program remaining to be executed
    - If $P = \downarrow$ then σ is a terminal state
  - η is the current variable valuation

Probabilistic Programs — Semantics

- Set of variable valuations: $\mathbb{V} = \{\eta \mid \eta\colon \mathsf{Var} \to \mathbb{Q}^+\}$
- Set of program states: $\mathbb{S} = \big(\mathsf{Prog} \cup \{\downarrow\}\big) \times \mathbb{V} \times I \times \{L, R\}^*$, for $I = [0, 1] \subseteq \mathbb{Q}^+$
- A closer look at a program state $\sigma = \langle P, \eta, a, \theta \rangle \in \mathbb{S}$:
  - P is the program remaining to be executed
    - If $P = \downarrow$ then σ is a terminal state
  - η is the current variable valuation
  - θ is a string over $\{L, R\}$ that encodes which probabilistic choices were made to reach σ

Probabilistic Programs — Semantics

- Set of variable valuations: $\mathbb{V} = \{\eta \mid \eta\colon \mathsf{Var} \to \mathbb{Q}^+\}$
- Set of program states: $\mathbb{S} = \big(\mathsf{Prog} \cup \{\downarrow\}\big) \times \mathbb{V} \times I \times \{L, R\}^*$, for $I = [0, 1] \subseteq \mathbb{Q}^+$
- A closer look at a program state $\sigma = \langle P, \eta, a, \theta \rangle \in \mathbb{S}$:
  - P is the program remaining to be executed
    - If $P = \downarrow$ then σ is a terminal state
  - η is the current variable valuation
  - θ is a string over $\{L, R\}$ that encodes which probabilistic choices were made to reach σ
  - a is the probability that those choices were made

Probabilistic Programs — Semantics continued

- **Semantics of probabilistic programs**:

Probabilistic Programs — Semantics continued

- **Semantics of probabilistic programs**: smallest relation $\vdash \subseteq \mathbb{S} \times \mathbb{S}$ which satisfies the following inference rules:

Probabilistic Programs — Semantics continued

- **Semantics of probabilistic programs**: smallest relation $\vdash \subseteq \mathbb{S} \times \mathbb{S}$ which satisfies the following inference rules:

$$(\text{assign})\ \frac{}{\langle v \texttt{ := } e, \eta, a, \theta \rangle \vdash \langle \downarrow, \eta[v \mapsto \max\{[\![e]\!]_\eta, 0\}], a, \theta \rangle}$$

Probabilistic Programs — Semantics continued

- **Semantics of probabilistic programs**: smallest relation $\vdash \subseteq \mathbb{S} \times \mathbb{S}$ which satisfies the following inference rules:

$$(\text{assign})\ \frac{}{\langle v \texttt{ := } e, \eta, a, \theta \rangle \vdash \langle \downarrow, \eta[v \mapsto \max\{[\![e]\!]_\eta, 0\}], a, \theta \rangle}$$

Probabilistic Programs — Semantics continued

- **Semantics of probabilistic programs**: smallest relation $\vdash \subseteq \mathbb{S} \times \mathbb{S}$ which satisfies the following inference rules:

$$\text{(assign)}\ \frac{}{\langle v := e, \eta, a, \theta\rangle \vdash \langle \downarrow, \eta[v \mapsto \max\{[\![e]\!]_\eta, 0\}], a, \theta\rangle}$$

$$\text{(concat1)}\ \frac{\langle P_1, \eta, a, \theta\rangle \vdash \langle P_1', \eta', a', \theta'\rangle}{\langle P_1 ; P_2, \eta, a, \theta\rangle \vdash \langle P_1' ; P_2, \eta', a', \theta'\rangle}$$

$$\text{(concat2)}\ \frac{}{\langle \downarrow ; P_2, \eta, a, \theta\rangle \vdash \langle P_2, \eta, a, \theta\rangle}$$

Probabilistic Programs — Semantics continued

- **Semantics of probabilistic programs**: smallest relation $\vdash \subseteq \mathbb{S} \times \mathbb{S}$ which satisfies the following inference rules:

$$\text{(assign)}\ \frac{}{\langle v \texttt{ := } e,\ \eta,\ a,\ \theta\rangle \ \vdash\ \langle \downarrow,\ \eta[v \mapsto \max\{[\![e]\!]_\eta,\ 0\}],\ a,\ \theta\rangle}$$

$$\text{(concat1)}\ \frac{\langle P_1,\ \eta,\ a,\ \theta\rangle \ \vdash\ \langle P_1',\ \eta',\ a',\ \theta'\rangle}{\langle P_1 \texttt{;}\ P_2,\ \eta,\ a,\ \theta\rangle \ \vdash\ \langle P_1' \texttt{;}\ P_2,\ \eta',\ a',\ \theta'\rangle}$$

$$\text{(concat2)}\ \frac{}{\langle \downarrow \texttt{;}\ P_2,\ \eta,\ a,\ \theta\rangle \ \vdash\ \langle P_2,\ \eta,\ a,\ \theta\rangle}$$

$$\text{(while1)}\ \frac{[\![b]\!]_\eta = \mathsf{True}}{\langle \texttt{WHILE (}b\texttt{) \{}P\texttt{\},}\ \eta,\ a,\ \theta\rangle \ \vdash\ \langle P \texttt{;}\ \texttt{WHILE (}b\texttt{) \{}P\texttt{\},}\ \eta,\ a,\ \theta\rangle}$$

$$\text{(while2)}\ \frac{[\![b]\!]_\eta = \mathsf{False}}{\langle \texttt{WHILE (}b\texttt{) \{}P\texttt{\},}\ \eta,\ a,\ \theta\rangle \ \vdash\ \langle \downarrow,\ \eta,\ a,\ \theta\rangle}$$

Probabilistic Programs — Semantics continued

$$(\text{prob1})\ \frac{}{\langle \{P_1\}\ [p]\ \{P_2\},\ \eta,\ a,\ \theta \rangle\ \vdash\ \langle P_1,\ \eta,\ a \cdot p,\ \theta \cdot L \rangle}$$

Probabilistic Programs — Semantics continued

$$(\text{prob1})\ \frac{}{\langle \{P_1\}\ [p]\ \{P_2\},\ \eta,\ a,\ \theta \rangle\ \vdash\ \langle P_1,\ \eta,\ a \cdot p,\ \theta \cdot L \rangle}$$

Probabilistic Programs — Semantics continued

$$(\text{prob1})\ \frac{}{\langle \{P_1\}\ [p]\ \{P_2\},\ \eta,\ a,\ \theta \rangle\ \vdash\ \langle P_1,\ \eta,\ a \cdot p,\ \theta \cdot L \rangle}$$

$$(\text{prob2})\ \frac{}{\langle \{P_1\}\ [p]\ \{P_2\},\ \eta,\ a,\ \theta \rangle\ \vdash\ \langle P_2,\ \eta,\ a \cdot (1-p),\ \theta \cdot R \rangle}$$

Probabilistic Programs — Semantics continued

$$(\text{prob1})\ \frac{}{\langle \{P_1\}\ [p]\ \{P_2\},\ \eta,\ a,\ \theta \rangle\ \vdash\ \langle P_1,\ \eta,\ a \cdot p,\ \theta \cdot L \rangle}$$

$$(\text{prob2})\ \frac{}{\langle \{P_1\}\ [p]\ \{P_2\},\ \eta,\ a,\ \theta \rangle\ \vdash\ \langle P_2,\ \eta,\ a \cdot (1-p),\ \theta \cdot R \rangle}$$

Probabilistic Programs — Semantics continued

$$\text{(prob1)}\ \frac{}{\langle \{P_1\}\ [p]\ \{P_2\},\ \eta,\ a,\ \theta\rangle\ \vdash\ \langle P_1,\ \eta,\ a \cdot p,\ \theta \cdot L\rangle}$$

$$\text{(prob2)}\ \frac{}{\langle \{P_1\}\ [p]\ \{P_2\},\ \eta,\ a,\ \theta\rangle\ \vdash\ \langle P_2,\ \eta,\ a \cdot (1-p),\ \theta \cdot R\rangle}$$

- Use $\sigma \vdash^k \tau$ in the usual sense

Probabilistic Programs — Semantics continued

$$\text{(prob1)}\ \frac{}{\langle \{P_1\}\ [p]\ \{P_2\},\ \eta,\ a,\ \theta\rangle\ \vdash\ \langle P_1,\ \eta,\ a \cdot p,\ \theta \cdot L\rangle}$$

$$\text{(prob2)}\ \frac{}{\langle \{P_1\}\ [p]\ \{P_2\},\ \eta,\ a,\ \theta\rangle\ \vdash\ \langle P_2,\ \eta,\ a \cdot (1-p),\ \theta \cdot R\rangle}$$

- Use $\sigma \vdash^k \tau$ in the usual sense
- Write $\sigma \vdash_{\text{(name)}} \tau$

Probabilistic Programs — Semantics continued

$$\text{(prob1)}\ \frac{}{\langle \{P_1\}\ [p]\ \{P_2\},\ \eta,\ a,\ \theta\rangle\ \vdash\ \langle P_1,\ \eta,\ a \cdot p,\ \theta \cdot L\rangle}$$

$$\text{(prob2)}\ \frac{}{\langle \{P_1\}\ [p]\ \{P_2\},\ \eta,\ a,\ \theta\rangle\ \vdash\ \langle P_2,\ \eta,\ a \cdot (1-p),\ \theta \cdot R\rangle}$$

- Use $\sigma \vdash^k \tau$ in the usual sense
- Write $\sigma \vdash_{\text{(name)}} \tau$ if τ is inferred by the use of the (name)–rule (for name $\in \{\text{assign, concat1, } \ldots\}$)

State Successors — The Classical Case

- Let $\sigma = \langle P, \eta, a, \theta \rangle$

State Successors — The Classical Case

- Let $\sigma = \langle P, \eta, a, \theta \rangle$
- Assume next instruction in P is a classical instruction

State Successors — The Classical Case

- Let $\sigma = \langle P, \eta, a, \theta \rangle$
- Assume next instruction in P is a classical instruction

Ex. function T that computes the (unique) successor of σ:

State Successors — The Classical Case

- Let $\sigma = \langle P, \eta, a, \theta \rangle$
- Assume next instruction in P is a classical instruction

Ex. function T that computes the (unique) successor of σ:

$$\mathsf{T}(\sigma) = \begin{cases} \tau, & \text{if } \sigma \vdash \tau \\ \top, & \text{else} \end{cases}$$

State Successors — The Classical Case

- Let $\sigma = \langle P, \eta, a, \theta \rangle$
- Assume next instruction in P is a classical instruction

Ex. function T that computes the (unique) successor of σ:

$$\mathsf{T}(\sigma) = \begin{cases} \tau, & \text{if } \sigma \vdash \tau \\ \top, & \text{else (i.e. if } \sigma = \langle \downarrow, \eta, a, \theta \rangle) \end{cases}$$

State Successors — The Classical Case

- Let $\sigma = \langle P, \eta, a, \theta \rangle$
- Assume next instruction in P is a classical instruction

Ex. function T that computes the (unique) successor of σ:

$$\mathsf{T}(\sigma) = \begin{cases} \tau, & \text{if } \sigma \vdash \tau \\ \top, & \text{else (i.e. if } \sigma = \langle \downarrow, \eta, a, \theta \rangle) \end{cases}$$

How do we deal with probabilistic choice?

State Successors — The Classical Case

- Let $\sigma = \langle P, \eta, a, \theta \rangle$
- Assume next instruction in P is a classical instruction

Ex. function T that computes the (unique) successor of σ:

$$\mathsf{T}(\sigma) = \begin{cases} \tau, & \text{if } \sigma \vdash \tau \\ \top, & \text{else (i.e. if } \sigma = \langle \downarrow, \eta, a, \theta \rangle) \end{cases}$$

How do we deal with probabilistic choice?

Basically, we simply tell T how to resolve it!

State Successors — The Probabilistic Case

- Let $\sigma = \langle P, \eta, a, \theta \rangle$

State Successors — The Probabilistic Case

- Let $\sigma = \langle P, \eta, a, \theta \rangle$
- Assume next instruction in P is a probabilistic choice

State Successors — The Probabilistic Case

- Let $\sigma = \langle P, \eta, a, \theta \rangle$
- Assume next instruction in P is a probabilistic choice
- σ has two successors according to $\vdash$

State Successors — The Probabilistic Case

- Let $\sigma = \langle P, \eta, a, \theta \rangle$
- Assume next instruction in P is a probabilistic choice
- σ has two successors according to $\vdash$
- Provide a symbol $s \in \{L, R\}$ resolving the probabilistic

State Successors — The Probabilistic Case

- Let $\sigma = \langle P, \eta, a, \theta \rangle$
- Assume next instruction in P is a probabilistic choice
- σ has two successors according to $\vdash$
- Provide a symbol $s \in \{L, R\}$ resolving the probabilistic

There exists a function T_{prob} that computes the successor of σ:

State Successors — The Probabilistic Case

- Let $\sigma = \langle P, \eta, a, \theta \rangle$
- Assume next instruction in P is a probabilistic choice
- σ has two successors according to $\vdash$
- Provide a symbol $s \in \{L, R\}$ resolving the probabilistic

There exists a function T_{prob} that computes the successor of σ:

$$\mathsf{T}_{prob}(\sigma, s) = \begin{cases} \tau_L, & \text{if } s = L \text{ and } \sigma \vdash_{\text{(prob1)}} \tau_L \\ \tau_R, & \text{if } s = R \text{ and } \sigma \vdash_{\text{(prob2)}} \tau_R \end{cases}$$

State Successors — The Probabilistic Case

- Let $\sigma = \langle P, \eta, a, \theta \rangle$
- Assume next instruction in P is a probabilistic choice
- σ has two successors according to $\vdash$
- Provide a symbol $s \in \{L, R\}$ resolving the probabilistic

There exists a function T_{prob} that computes the successor of σ:

$$\mathsf{T}_{prob}(\sigma, s) = \begin{cases} \tau_L, & \text{if } s = L \text{ and } \sigma \vdash_{\text{(prob1)}} \tau_L \\ \tau_R, & \text{if } s = R \text{ and } \sigma \vdash_{\text{(prob2)}} \tau_R \end{cases}$$

Provided s, the successor of σ is unique!

k–th State Successors — The Combined Case

k–th State Successors — The Combined Case

There exists a computable function T^*_{prob}, such that:

k–th State Successors — The Combined Case

There exists a computable function T^*_{prob}, such that:

$$\mathsf{T}^*_{prob}(\sigma, k, w) = \begin{cases} \tau, & \text{if } \sigma = \langle P, \eta, a, \theta \rangle \vdash^k \langle P', \eta', a', \theta \cdot w \rangle = \tau \\ \top, & \text{else} \end{cases}$$

k–th State Successors — The Combined Case

There exists a computable function T^*_{prob}, such that:

$$\mathsf{T}^*_{prob}(\sigma, k, w) = \begin{cases} \tau, & \text{if } \sigma = \langle P, \eta, a, \theta \rangle \vdash^k \langle P', \eta', a', \theta \cdot w \rangle = \tau \\ \top, & \text{else} \end{cases}$$

- T^*_{prob} returns a successor state τ, if

k–th State Successors — The Combined Case

There exists a computable function T^*_{prob}, such that:

$$\mathsf{T}^*_{prob}(\sigma, k, w) = \begin{cases} \tau, & \text{if } \sigma = \langle P, \eta, a, \theta \rangle \vdash^k \langle P', \eta', a', \theta \cdot w \rangle = \tau \\ \top, & \text{else} \end{cases}$$

- T^*_{prob} returns a successor state τ, if
  - $\sigma \vdash^k \tau$

k–th State Successors — The Combined Case

There exists a computable function T^*_{prob}, such that:

$$\mathsf{T}^*_{prob}(\sigma, k, w) = \begin{cases} \tau, & \text{if } \sigma = \langle P, \eta, a, \theta \rangle \vdash^k \langle P', \eta', a', \theta \cdot w \rangle = \tau \\ \top, & \text{else} \end{cases}$$

- T^*_{prob} returns a successor state τ, if
  - $\sigma \vdash^k \tau$
  - **Exactly** $|w|$ probabilistic choices occur

k–th State Successors — The Combined Case

There exists a computable function T^*_{prob}, such that:

$$\mathsf{T}^*_{prob}(\sigma, k, w) = \begin{cases} \tau, & \text{if } \sigma = \langle P, \eta, a, \theta \rangle \vdash^k \langle P', \eta', a', \theta \cdot w \rangle = \tau \\ \top, & \text{else} \end{cases}$$

- T^*_{prob} returns a successor state τ, if
  - $\sigma \vdash^k \tau$
  - **Exactly** $|w|$ probabilistic choices occur
  - The probabilistic choices are resolved according to w

k–th State Successors — The Combined Case

There exists a computable function T^*_{prob}, such that:

$$\mathsf{T}^*_{prob}(\sigma, k, w) = \begin{cases} \tau, & \text{if } \sigma = \langle P, \eta, a, \theta \rangle \vdash^k \langle P', \eta', a', \theta \cdot w \rangle = \tau \\ \top, & \text{else} \end{cases}$$

- T^*_{prob} returns a successor state τ, if
  - $\sigma \vdash^k \tau$
  - **Exactly** $|w|$ probabilistic choices occur
  - The probabilistic choices are resolved according to w
- Otherwise, T^*_{prob} returns $\top$

Some Helper Functions

Some Helper Functions

Extracting Probabilities and Variable Values of Terminal States

$$\alpha(\sigma) = \begin{cases} a, & \text{if } \sigma = \langle \downarrow, \eta, a, \theta \rangle \\ 0, & \text{otherwise} \end{cases}$$

Some Helper Functions

Extracting Probabilities and Variable Values of Terminal States

$$\alpha(\sigma) = \begin{cases} a, & \text{if } \sigma = \langle \downarrow, \eta, a, \theta \rangle \\ 0, & \text{otherwise} \end{cases}$$

$$\wp(\sigma, v) = \begin{cases} \eta(v) \cdot a, & \text{if } \sigma = \langle \downarrow, \eta, a, \theta \rangle \\ 0, & \text{otherwise} \end{cases}$$

Some Helper Functions

Extracting Probabilities and Variable Values of Terminal States

$$\alpha(\sigma) = \begin{cases} a, & \text{if } \sigma = \langle \downarrow, \eta, a, \theta \rangle \\ 0, & \text{otherwise} \end{cases}$$

$$\wp(\sigma, v) = \begin{cases} \eta(v) \cdot a, & \text{if } \sigma = \langle \downarrow, \eta, a, \theta \rangle \\ 0, & \text{otherwise} \end{cases}$$

Computable Enumeration of all $w \in \{L, R\}^*$

There exists a computable bijection $h \colon \mathbb{N} \to \{L, R\}^*$.

Expected Outcomes

The Expected Outcome $\mathsf{E}_P(v)$

The expected outcome of variable v after executing program P:

$$\mathsf{E}_P(v) \;:=\; \sum_{i\in\mathbb{N}} \sum_{k\in\mathbb{N}} \wp\Big(\mathsf{T}^*_{prob}\big(\langle P,\, \eta_0,\, 1,\, \varepsilon\rangle,\, k,\, h(i)\big),\, v\Big)$$

Expected Outcomes

The Expected Outcome $\mathsf{E}_P(v)$

The expected outcome of variable v after executing program P:

$$\mathsf{E}_P(v) \;:=\; \sum_{i \in \mathbb{N}} \sum_{k \in \mathbb{N}} \wp\Big(\mathsf{T}^*_{prob}\big(\langle P,\, \eta_0,\, 1,\, \varepsilon\rangle,\, k,\, h(i)\big),\, v\Big)$$

- Start in initial state $\langle P,\, \eta_0,\, 1,\, \varepsilon\rangle$

Expected Outcomes

The Expected Outcome $\mathsf{E}_P(v)$

The expected outcome of variable v after executing program P:

$$\mathsf{E}_P(v) \;:=\; \sum_{i \in \mathbb{N}} \sum_{k \in \mathbb{N}} \wp\Big(\mathsf{T}^*_{prob}\big(\langle P,\, \eta_0,\, 1,\, \varepsilon\rangle,\, k,\, h(i)\big),\, v\Big)$$

- Start in initial state $\langle P,\, \eta_0,\, 1,\, \varepsilon\rangle$
- Allow P any finite number of $k \in \mathbb{N}$ steps to terminate

Expected Outcomes

The Expected Outcome $\mathsf{E}_P(v)$

The expected outcome of variable v after executing program P:

$$\mathsf{E}_P(v) \;:=\; \sum_{i \in \mathbb{N}} \sum_{k \in \mathbb{N}} \wp\Big(\mathsf{T}^*_{prob}\big(\langle P,\, \eta_0,\, 1,\, \varepsilon\rangle,\, k,\, h(i)\big),\, v\Big)$$

- Start in initial state $\langle P,\, \eta_0,\, 1,\, \varepsilon\rangle$
- Allow P any finite number of $k \in \mathbb{N}$ steps to terminate
- Sum over all possible resolutions of the probabilistic choice

Expected Outcomes

The Expected Outcome $\mathsf{E}_P(v)$

The expected outcome of variable v after executing program P:

$$\mathsf{E}_P(v) \;:=\; \sum_{i\in\mathbb{N}} \sum_{k\in\mathbb{N}} \wp\Big(\mathsf{T}^*_{prob}\big(\langle P,\, \eta_0,\, 1,\, \varepsilon\rangle,\, k,\, h(i)\big),\, v\Big)$$

- Start in initial state $\langle P,\, \eta_0,\, 1,\, \varepsilon\rangle$
- Allow P any finite number of $k \in \mathbb{N}$ steps to terminate
- Sum over all possible resolutions of the probabilistic choice
- For each terminal state, take the probability times the value of v as the summand

Termination Probabilities

The Probability $\mathsf{Pr}_P(\downarrow)$ that P terminates

The probability that P terminates:

$$\mathsf{Pr}_P(\downarrow) \;:=\; \sum_{i \in \mathbb{N}} \sum_{k \in \mathbb{N}} \alpha\Big(\mathsf{T}^*_{prob}\big(\langle P,\, \eta_0,\, 1,\, \varepsilon\rangle,\, k,\, h(i)\big)\Big)$$

Termination Probabilities

The Probability $\mathsf{Pr}_P(\downarrow)$ that P terminates

The probability that P terminates:

$$\mathsf{Pr}_P(\downarrow) := \sum_{i \in \mathbb{N}} \sum_{k \in \mathbb{N}} \alpha\Big(\mathsf{T}^*_{prob}\big(\langle P, \eta_0, 1, \varepsilon\rangle, k, h(i)\big)\Big)$$

- Start in initial state $\langle P, \eta_0, 1, \varepsilon\rangle$

Termination Probabilities

The Probability $\mathsf{Pr}_P(\downarrow)$ that P terminates

The probability that P terminates:

$$\mathsf{Pr}_P(\downarrow) \;:=\; \sum_{i \in \mathbb{N}} \sum_{k \in \mathbb{N}} \alpha\Big(\mathsf{T}^*_{prob}\big(\langle P,\, \eta_0,\, 1,\, \varepsilon\rangle,\, k,\, h(i)\big)\Big)$$

- Start in initial state $\langle P,\, \eta_0,\, 1,\, \varepsilon\rangle$
- Allow P any finite number of $k \in \mathbb{N}$ steps to terminate

Termination Probabilities

The Probability $\mathsf{Pr}_P(\downarrow)$ that P terminates

The probability that P terminates:

$$\mathsf{Pr}_P(\downarrow) := \sum_{i \in \mathbb{N}} \sum_{k \in \mathbb{N}} \alpha\Big(\mathsf{T}^*_{prob}\big(\langle P, \eta_0, 1, \varepsilon\rangle, k, h(i)\big)\Big)$$

- Start in initial state $\langle P, \eta_0, 1, \varepsilon\rangle$
- Allow P any finite number of $k \in \mathbb{N}$ steps to terminate
- Sum over all possible resolutions of the probabilistic choice

Termination Probabilities

The Probability $\mathsf{Pr}_P(\downarrow)$ that P terminates

The probability that P terminates:

$$\mathsf{Pr}_P(\downarrow) := \sum_{i \in \mathbb{N}} \sum_{k \in \mathbb{N}} \alpha\Big(\mathsf{T}^*_{prob}\big(\langle P, \eta_0, 1, \varepsilon\rangle, k, h(i)\big)\Big)$$

- Start in initial state $\langle P, \eta_0, 1, \varepsilon\rangle$
- Allow P any finite number of $k \in \mathbb{N}$ steps to terminate
- Sum over all possible resolutions of the probabilistic choice
- For each terminal state, take the probability as the summand

Decision Problems We Analyzed

Decision Problems We Analyzed

Almost–Sure Termination $\mathcal{AST}$

$$P \in \mathcal{AST} \;:\Longleftrightarrow\; \mathsf{Pr}_P(\downarrow) = 1$$

Decision Problems We Analyzed

Almost–Sure Termination $\mathcal{AST}$

$$P \in \mathcal{AST} \;:\Longleftrightarrow\; \mathsf{Pr}_P(\downarrow) = 1$$

Lower and Upper Bounds, and Exact Expected Outcomes

$$\begin{aligned}(P, v, q) \in \mathcal{LEXP} &:\Longleftrightarrow q < \mathsf{E}_P(v)\\ (P, v, q) \in \mathcal{UEXP} &:\Longleftrightarrow q > \mathsf{E}_P(v)\\ (P, v, q) \in \mathcal{EXP} &:\Longleftrightarrow q = \mathsf{E}_P(v)\end{aligned}$$

Hardness of Deciding $\mathcal{LEXP}$

Recursive Enumerability of $\mathcal{LEXP}$

Recursive Enumerability of $\mathcal{LEXP}$

Lemma

$\mathcal{LEXP} \in \Sigma_1^0$, thus $\mathcal{LEXP}$ is recursively enumerable

Recursive Enumerability of $\mathcal{LEXP}$

Lemma

$\mathcal{LEXP} \in \Sigma_1^0$, thus $\mathcal{LEXP}$ is recursively enumerable

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

Recursive Enumerability of $\mathcal{LEXP}$

Lemma

$\mathcal{LEXP} \in \Sigma_1^0$, thus $\mathcal{LEXP}$ is recursively enumerable

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

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

$$\exists y_1 \exists y_2 \colon q < \sum_{i=1}^{y_1} \sum_{k=1}^{y_2} \wp\Big(\mathsf{T}^*_{prob}\big(\langle P,\, \eta_0,\, 1,\, \varepsilon\rangle,\, k,\, h(i)\big),\, v\Big)$$

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

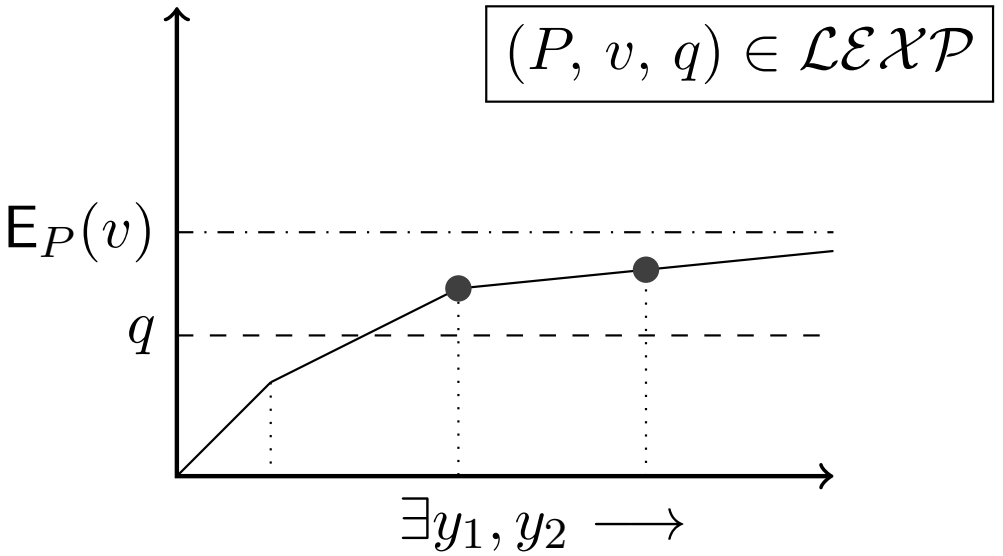
$\mathcal{LEXP}$ is defined by the following formula:

$$\exists y_1 \exists y_2 \colon q < \sum_{i=1}^{y_1} \sum_{k=1}^{y_2} \wp\Big(\mathsf{T}^*_{prob}\big(\langle P,\, \eta_0,\, 1,\, \varepsilon\rangle,\, k,\, h(i)\big),\, v\Big)$$

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

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

$$\exists y_1 \exists y_2 \colon q < \sum_{i=1}^{y_1} \sum_{k=1}^{y_2} \wp\Big(\mathsf{T}^*_{prob}\big(\langle P,\, \eta_0,\, 1,\, \varepsilon\rangle,\, k,\, h(i)\big),\, v\Big)$$



Upper Bound for Hardness of Deciding $\mathcal{UEXP}$

Upper Bound for Hardness of Deciding $\mathcal{UEXP}$

Upper Bound for Hardness of Deciding $\mathcal{UEXP}$

Lemma

$\mathcal{UEXP} \in \Sigma_2^0$

Upper Bound for Hardness of Deciding $\mathcal{UEXP}$

Lemma

$\mathcal{UEXP} \in \Sigma_2^0$

Proof of $\mathcal{UEXP} \in \Sigma_2^0$

Upper Bound for Hardness of Deciding $\mathcal{UEXP}$

Lemma

$\mathcal{UEXP} \in \Sigma_2^0$

Proof of $\mathcal{UEXP} \in \Sigma_2^0$

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

$$\exists \delta > 0 \, \forall y_1 \, \forall y_2 : q - \delta > \sum_{i=1}^{y_1} \sum_{k=1}^{y_2} \wp\Big(\mathsf{T}^*_{prob}\big(\langle P, \eta_0, 1, \varepsilon \rangle, k, h(i)\big), v \Big)$$

Proof of $\mathcal{UEXP} \in \Sigma_2^0$

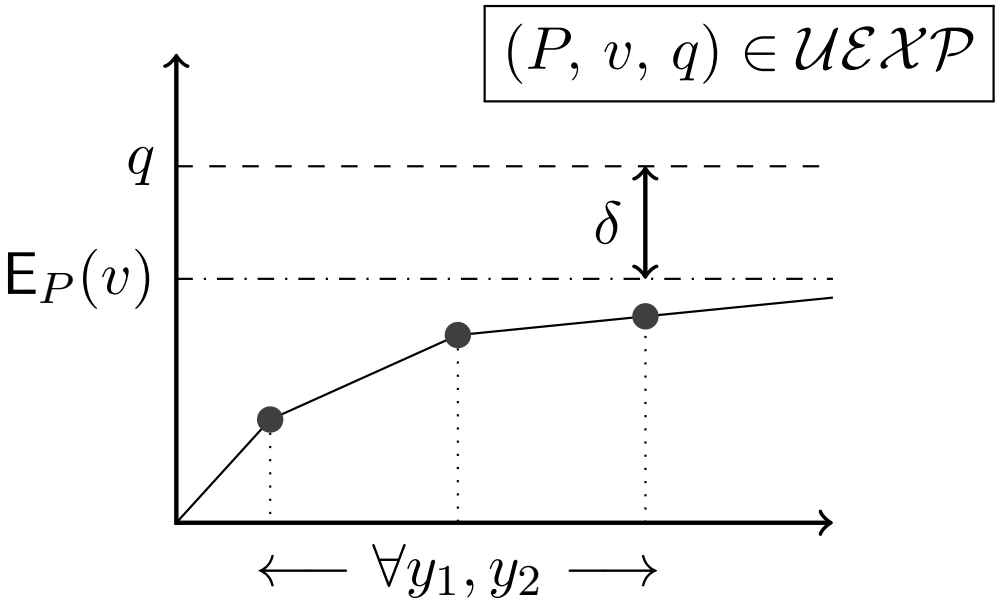
$\mathcal{UEXP}$ is defined by the following formula:

$$\exists \delta > 0 \, \forall y_1 \, \forall y_2 : q - \delta > \sum_{i=1}^{y_1} \sum_{k=1}^{y_2} \wp\Big(\mathsf{T}^*_{prob}\big(\langle P, \eta_0, 1, \varepsilon \rangle, k, h(i)\big), v \Big)$$

Proof of $\mathcal{UEXP} \in \Sigma_2^0$

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

$$\exists \delta > 0 \, \forall y_1 \, \forall y_2 : q - \delta > \sum_{i=1}^{y_1} \sum_{k=1}^{y_2} \wp\Big(\mathsf{T}^*_{prob}\big(\langle P,\, \eta_0,\, 1,\, \varepsilon \rangle,\, k,\, h(i)\big),\, v \Big)$$



Actual Hardness of Deciding $\mathcal{UEXP}$ and $\mathcal{EXP}$

The Universal Halting Problem $\mathcal{UH}$

The Universal Halting Problem $\mathcal{UH}$

The universal halting problem $\mathcal{UH} \subset$ ordProg:

$$P \in \mathcal{UH} \;:\Longleftrightarrow\; \forall \eta\, \exists k\, \exists \eta' \colon \langle P, \eta, 1, \varepsilon \rangle \vdash^{k} \langle \downarrow, \eta', 1, \varepsilon \rangle$$

The Universal Halting Problem $\mathcal{UH}$

The universal halting problem $\mathcal{UH} \subset \mathsf{ordProg}$:

$$P \in \mathcal{UH} \;:\Longleftrightarrow\; \forall \eta\, \exists k\, \exists \eta' \colon \langle P, \eta, 1, \varepsilon \rangle \vdash^{k} \langle \downarrow, \eta', 1, \varepsilon \rangle$$

The complement of the universal halting problem $\overline{\mathcal{UH}}$:

$$\overline{\mathcal{UH}} \;:=\; \mathsf{ordProg} \setminus \mathcal{UH}\ .$$

The Universal Halting Problem $\mathcal{UH}$

The universal halting problem $\mathcal{UH} \subset$ ordProg:

$$P \in \mathcal{UH} \;:\Longleftrightarrow\; \forall \eta\, \exists k\, \exists \eta' \colon \langle P, \eta, 1, \varepsilon \rangle \vdash^k \langle \downarrow, \eta', 1, \varepsilon \rangle$$

The complement of the universal halting problem $\overline{\mathcal{UH}}$:

$$\overline{\mathcal{UH}} \;:=\; \text{ordProg} \setminus \mathcal{UH}\ .$$

Completenesses of $\mathcal{UH}$ and $\overline{\mathcal{UH}}$

$\mathcal{UH}$ is Π_2^0–complete and $\overline{\mathcal{UH}}$ is Σ_2^0–complete.

Hardness of Deciding $\mathcal{UEXP}$ and $\mathcal{EXP}$

Hardness of Deciding $\mathcal{UEXP}$ and $\mathcal{EXP}$

Lemma

$\mathcal{UEXP}$ is Σ_2^0–complete.

Hardness of Deciding $\mathcal{UEXP}$ and $\mathcal{EXP}$

Lemma

$\mathcal{UEXP}$ is Σ_2^0–complete.

Proof: Recall $\mathcal{UEXP} \in \Sigma_2^0$ and prove $\overline{\mathcal{UH}} \leq \mathcal{UEXP}$.

Hardness of Deciding $\mathcal{UEXP}$ and $\mathcal{EXP}$

Lemma

$\mathcal{UEXP}$ is Σ_2^0–complete.

Proof: Recall $\mathcal{UEXP} \in \Sigma_2^0$ and prove $\overline{\mathcal{UH}} \leq \mathcal{UEXP}$. Omitted.

Hardness of Deciding $\mathcal{UEXP}$ and $\mathcal{EXP}$

Lemma

$\mathcal{UEXP}$ is Σ_2^0–complete.

Proof: Recall $\mathcal{UEXP} \in \Sigma_2^0$ and prove $\overline{\mathcal{UH}} \leq \mathcal{UEXP}$. Omitted.

Theorem

$\mathcal{EXP}$ is Π_2^0–complete.

Hardness of Deciding $\mathcal{UEXP}$ and $\mathcal{EXP}$

Lemma

$\mathcal{UEXP}$ is Σ_2^0–complete.

Proof: Recall $\mathcal{UEXP} \in \Sigma_2^0$ and prove $\overline{\mathcal{UH}} \leq \mathcal{UEXP}$. Omitted.

Theorem

$\mathcal{EXP}$ is Π_2^0–complete.

Proof: Prove $\mathcal{EXP} \in \Pi_2^0$ and prove $\mathcal{UH} \leq \mathcal{EXP}$.

Hardness of Deciding $\mathcal{UEXP}$ and $\mathcal{EXP}$

Lemma

$\mathcal{UEXP}$ is Σ_2^0–complete.

Proof: Recall $\mathcal{UEXP} \in \Sigma_2^0$ and prove $\overline{\mathcal{UH}} \leq \mathcal{UEXP}$. Omitted.

Theorem

$\mathcal{EXP}$ is Π_2^0–complete.

Proof: Prove $\mathcal{EXP} \in \Pi_2^0$ and prove $\mathcal{UH} \leq \mathcal{EXP}$. Omitted.

Hardness of Deciding $\mathcal{AST}$

Hardness of Deciding Almost–Sure Termination

Hardness of Deciding Almost–Sure Termination

Lemma

$\mathcal{AST} \in \Pi_2^0$

Hardness of Deciding Almost–Sure Termination

Lemma

$\mathcal{AST} \in \Pi_2^0$

Proof: Prove $\mathcal{AST} \leq \mathcal{EXP}$.

Hardness of Deciding Almost–Sure Termination

Lemma

$\mathcal{AST} \in \Pi_2^0$

Proof: Prove $\mathcal{AST} \leq \mathcal{EXP}$. Omitted.

Hardness of Deciding Almost–Sure Termination

Lemma

$\mathcal{AST} \in \Pi_2^0$

Proof: Prove $\mathcal{AST} \leq \mathcal{EXP}$. Omitted.

Theorem

$\mathcal{AST}$ is Π_2^0–complete.

Hardness of Deciding Almost–Sure Termination

Lemma

$\mathcal{AST} \in \Pi_2^0$

Proof: Prove $\mathcal{AST} \leq \mathcal{EXP}$. Omitted.

Theorem

$\mathcal{AST}$ is Π_2^0–complete.

Proof: Recall $\mathcal{AST} \in \Pi_2^0$ and prove $\mathcal{UH} \leq \mathcal{AST}$.

Hardness of Deciding Almost–Sure Termination

Lemma

$\mathcal{AST} \in \Pi_2^0$

Proof: Prove $\mathcal{AST} \leq \mathcal{EXP}$. Omitted.

Theorem

$\mathcal{AST}$ is Π_2^0–complete.

Proof: Recall $\mathcal{AST} \in \Pi_2^0$ and prove $\mathcal{UH} \leq \mathcal{AST}$. **Not omitted!**

Proof of $\mathcal{UH} \leq \mathcal{AST}$

Proof obligation for $\mathcal{UH} \leq \mathcal{AST}$

Proof of $\mathcal{UH} \leq \mathcal{AST}$

Proof obligation for $\mathcal{UH} \leq \mathcal{AST}$

Find a computable function f, such that

$$Q \in \mathcal{UH} \iff f(Q) \in \mathcal{AST}$$

Proof of $\mathcal{UH} \leq \mathcal{AST}$

Proof obligation for $\mathcal{UH} \leq \mathcal{AST}$

Find a computable function f, such that

$$Q \in \mathcal{UH} \iff f(Q) \in \mathcal{AST}$$

First observation

There is a computable enumeration $g_Q \colon \mathbb{N} \to \{\mathsf{Var} \to \mathbb{Q}\}$ enumerating all possible variable valuations subject to

$$\forall i \in \mathbb{N}\ \forall v \in \mathsf{Var}\colon \Big[\big[g_Q(i)\big](v) \neq 0\Big] \implies \Big[v \text{ occurs in } Q\Big]$$

Proof of $\mathcal{UH} \leq \mathcal{AST}$

Candidate for f

Proof of $\mathcal{UH} \leq \mathcal{AST}$

Candidate for f

$f(Q)$ returns the following probabilistic program P:

Proof of $\mathcal{UH} \leq \mathcal{AST}$

Candidate for f

$f(Q)$ returns the following probabilistic program P:

i := 0; {continue := 0} [0.5] {continue := 1};
while (continue $\neq$ 0){
    i := i + 1;
    {continue := 0} [0.5] {continue := 1}
};
TQ

Proof of $\mathcal{UH} \leq \mathcal{AST}$

Candidate for f

$f(Q)$ returns the following probabilistic program P:

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

where TQ is a program that simulates Q on input $g_Q(i)$.

Partial Correctness

Partial Correctness

Probabilistic Program P Returned by $f(Q)$

i := 0; {continue := 0} [0.5] {continue := 1};
while (continue $\neq$ 0){
    i := i + 1;
    {continue := 0} [0.5] {continue := 1}
};
TQ

Partial Correctness

Probabilistic Program P Returned by $f(Q)$

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

- While–loop establishes geometric distribution on i

Partial Correctness

Probabilistic Program P Returned by $f(Q)$

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

- While–loop establishes geometric distribution on i — hence, by $g_Q(i)$, a geometric distribution on all possible inputs for Q

Partial Correctness

Probabilistic Program P Returned by $f(Q)$

i := 0; {continue := 0} [0.5] {continue := 1};
while (continue $\neq$ 0){
    i := i + 1;
    {continue := 0} [0.5] {continue := 1}
};
TQ

- While–loop establishes geometric distribution on i — hence, by $g_Q(i)$, a geometric distribution on all possible inputs for Q
- Then program P terminates with probability $\sum_{k\in\mathbb{N}} \frac{1}{2^k} = 1$ iff the simulation of Q terminates on every possible input $g_Q(i)$

Total Correctness

Total Correctness

- Code for g_Q is computable
- Code for simulation of Q on a given input is computable

Total Correctness

- Code for g_Q is computable
- Code for simulation of Q on a given input is computable
- So in total, program code for P is computable

Total Correctness

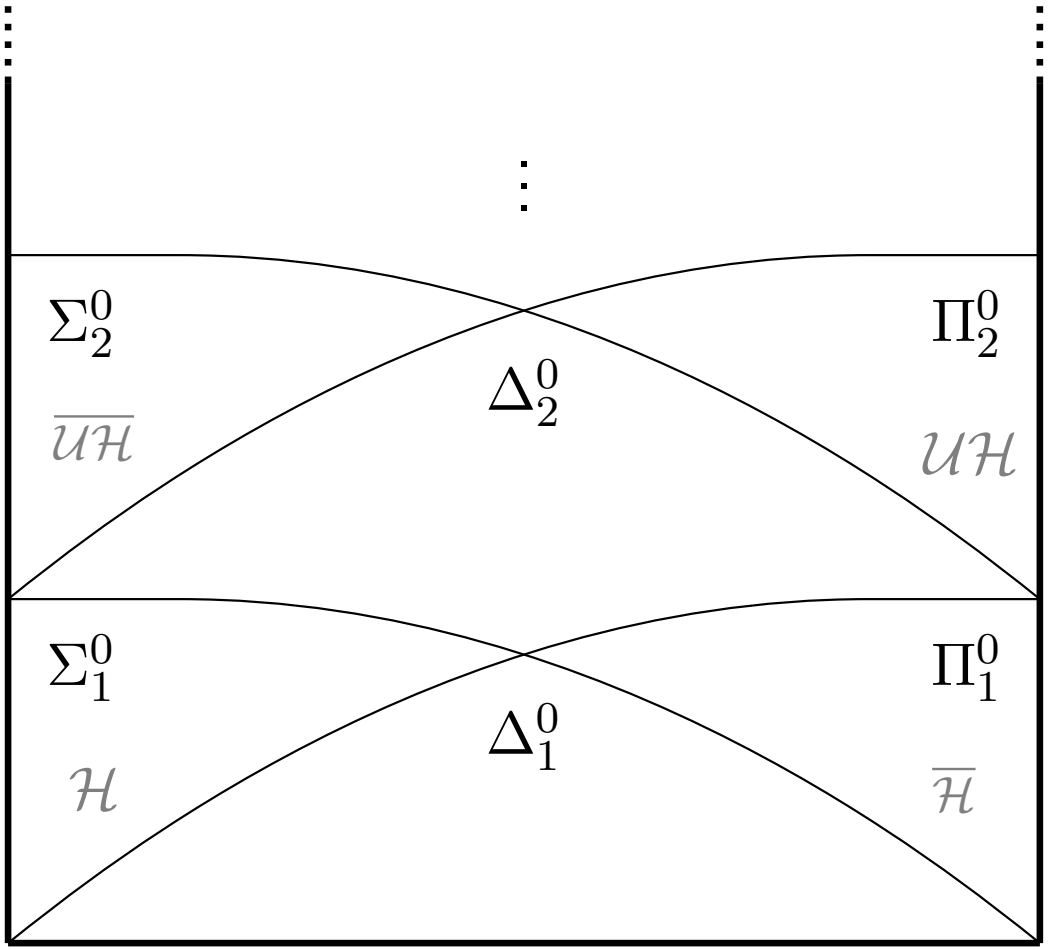
- Code for g_Q is computable
- Code for simulation of Q on a given input is computable
- So in total, program code for P is computable
- All of the above computations terminate

Total Correctness

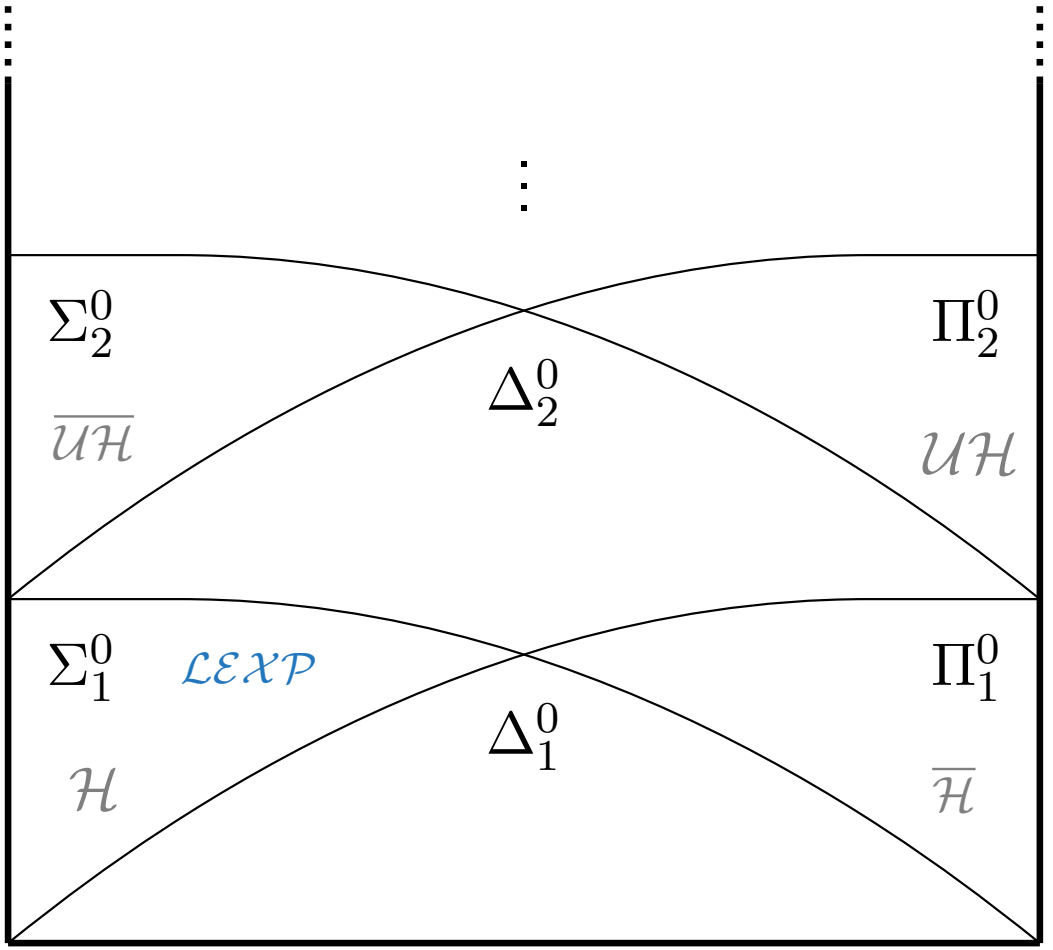
- Code for g_Q is computable
- Code for simulation of Q on a given input is computable
- So in total, program code for P is computable
- All of the above computations terminate

Summary

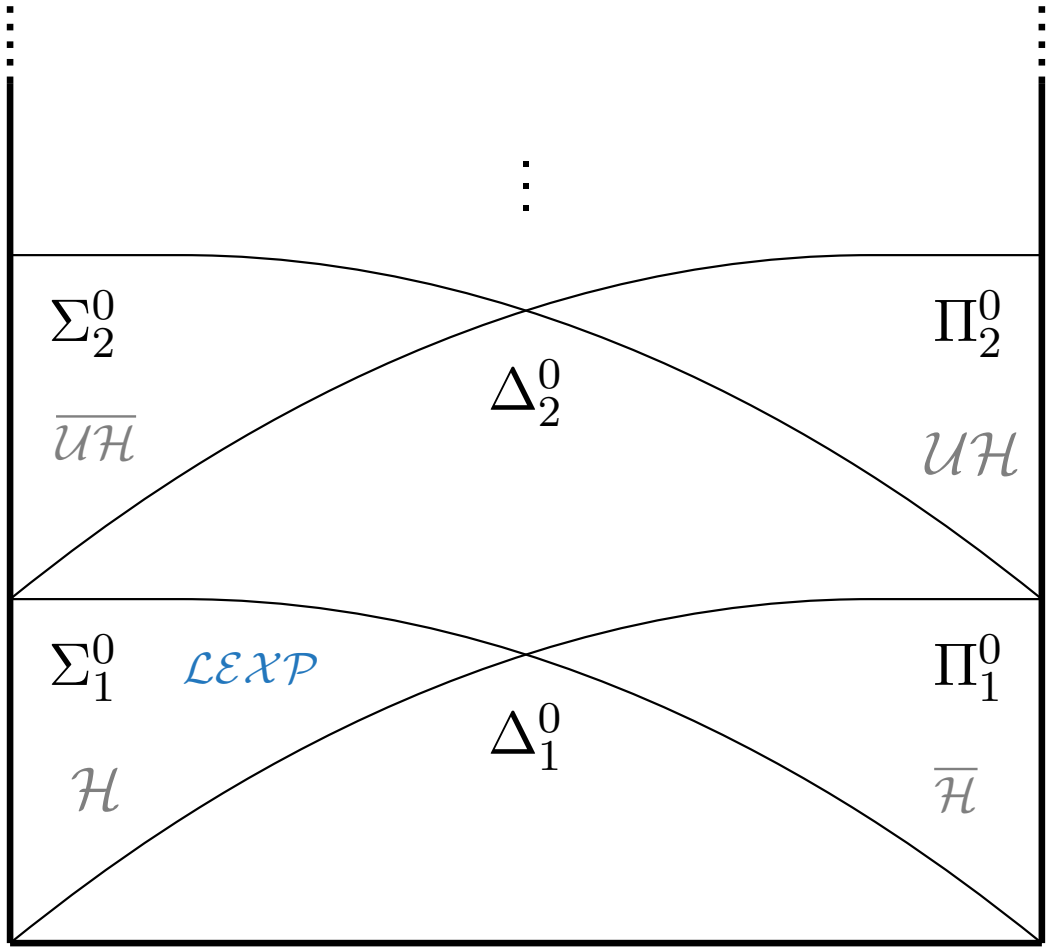
Summary


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

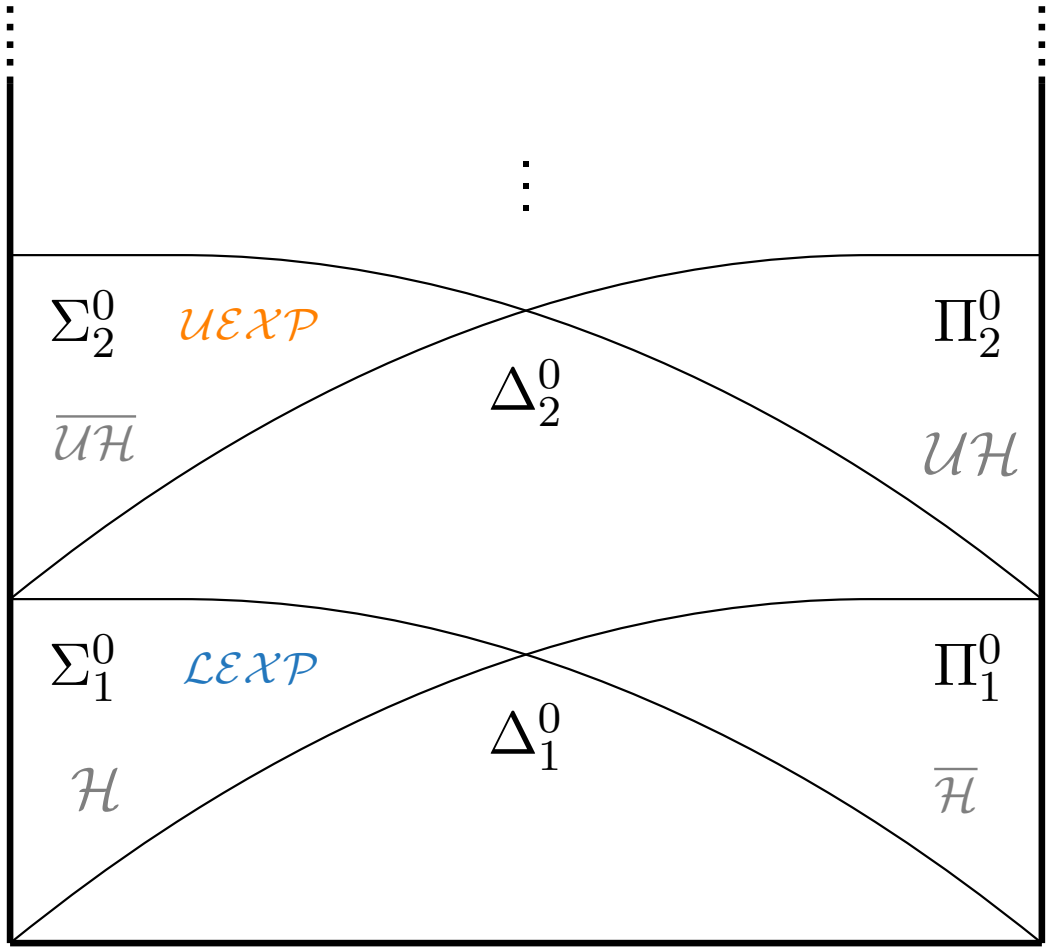
Summary


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
$\mathcal{H}$
$\overline{\mathcal{H}}$

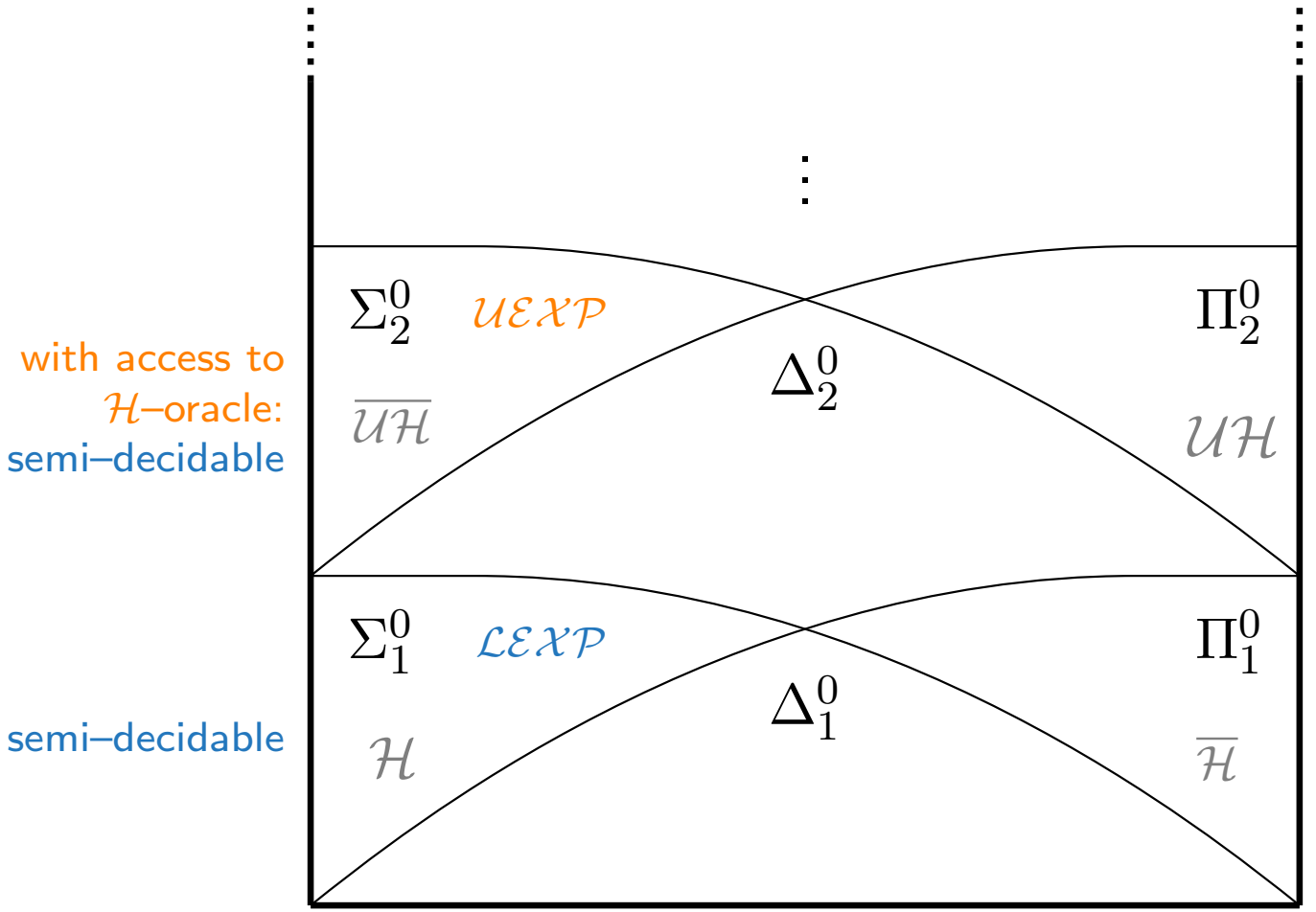
Summary


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

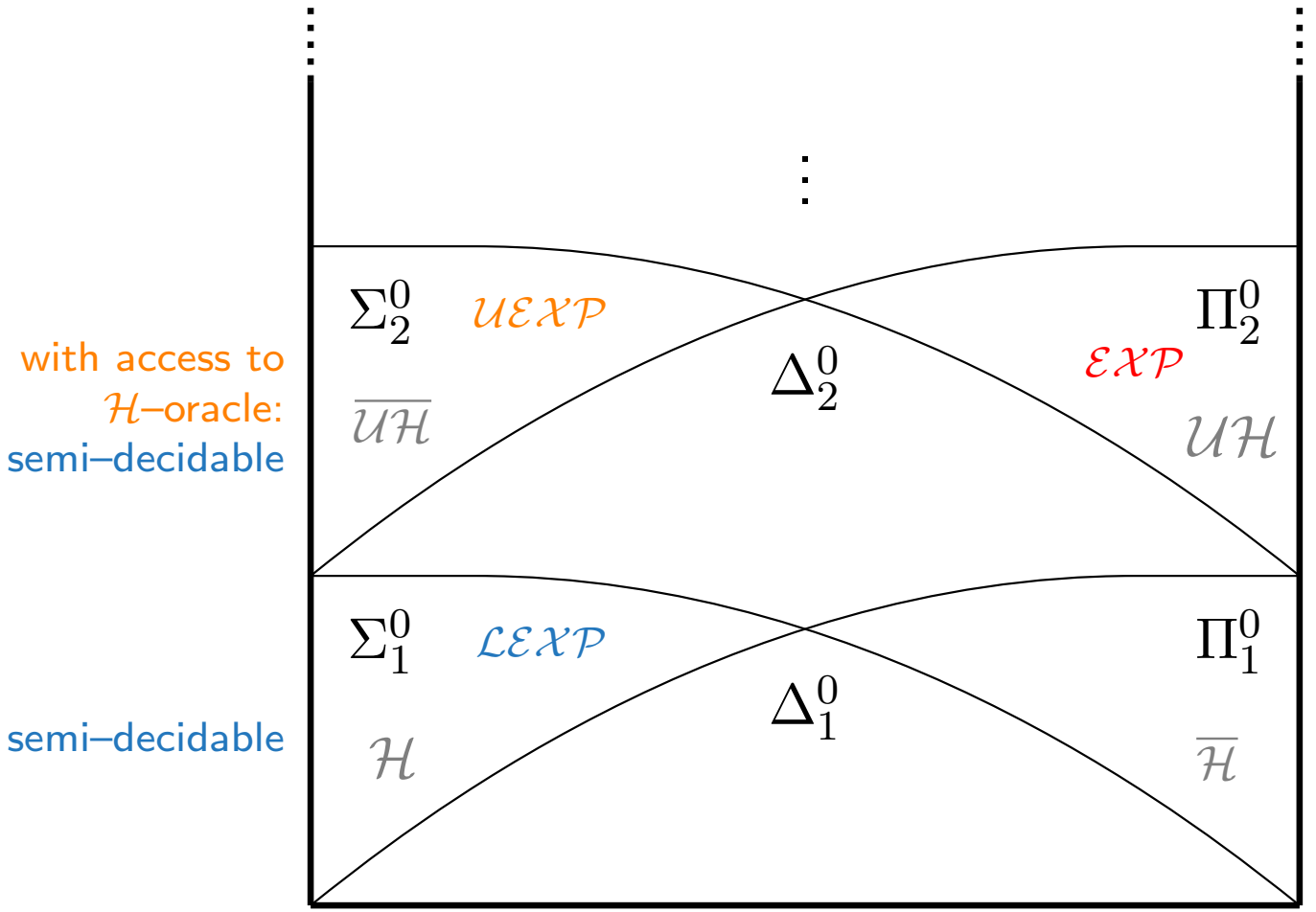
Summary


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

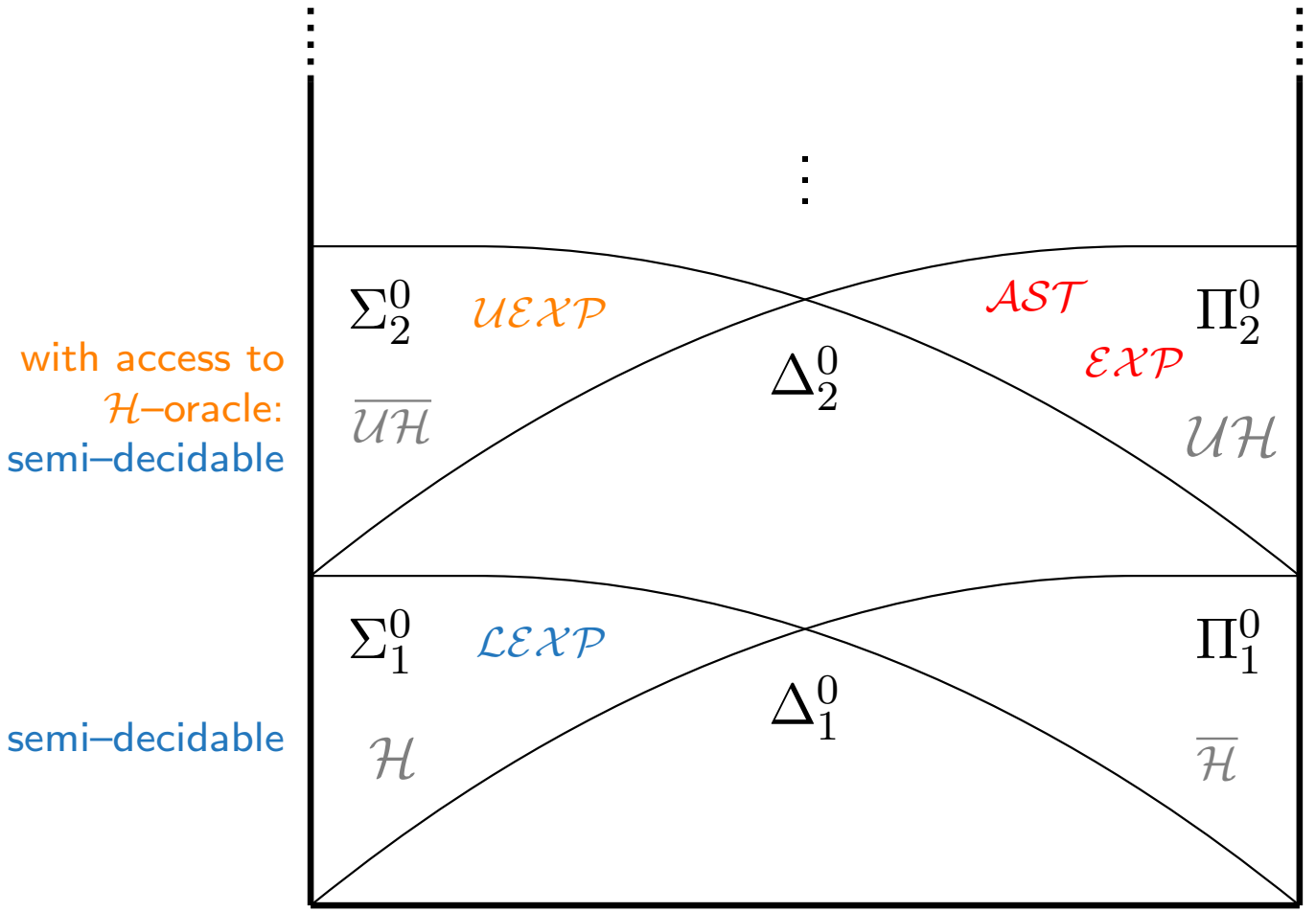
Summary


with access to
$\mathcal{H}$–oracle:
semi–decidable
semi–decidable
Σ_2^0
$\mathcal{UEXP}$
Π_2^0
Δ_2^0
$\overline{\mathcal{UH}}$
$\mathcal{UH}$
Σ_1^0
$\mathcal{LEXP}$
Π_1^0
Δ_1^0
$\mathcal{H}$
$\overline{\mathcal{H}}$

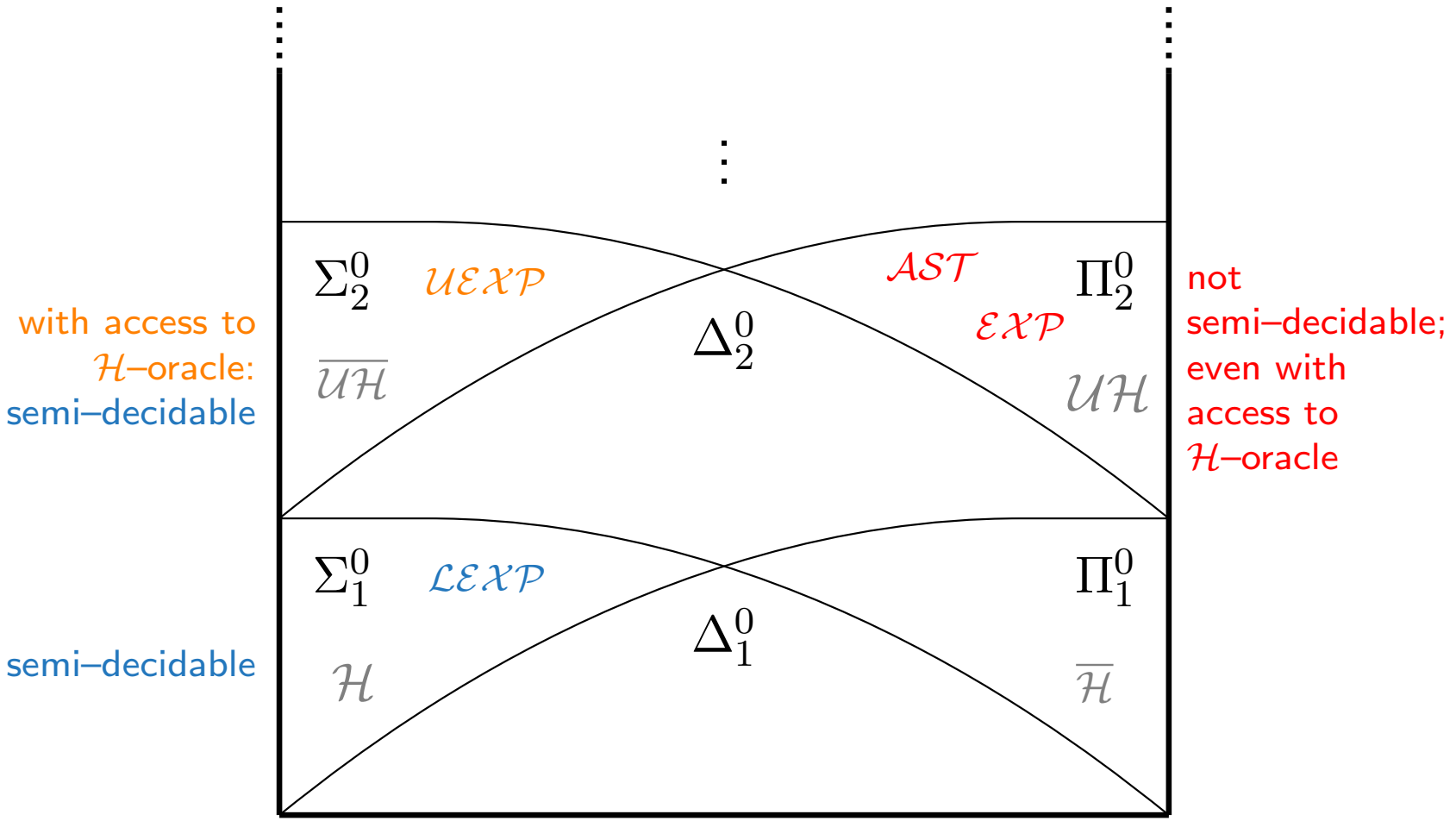
Summary


with access to
$\mathcal{H}$–oracle:
semi–decidable
semi–decidable
Σ_2^0
$\mathcal{UEXP}$
Δ_2^0
Π_2^0
$\mathcal{EXP}$
$\overline{\mathcal{UH}}$
$\mathcal{UH}$
Σ_1^0
$\mathcal{LEXP}$
Δ_1^0
Π_1^0
$\mathcal{H}$
$\overline{\mathcal{H}}$

Summary


with access to
$\mathcal{H}$–oracle:
semi–decidable
semi–decidable
Σ_2^0
$\mathcal{UEXP}$
$\mathcal{AST}$
Π_2^0
$\mathcal{EXP}$
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


with access to $\mathcal{H}$–oracle: semi–decidable
semi–decidable
not semi–decidable; even with access to $\mathcal{H}$–oracle
Σ_2^0
$\mathcal{UEXP}$
$\mathcal{AST}$
Π_2^0
$\mathcal{EXP}$
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

Thank you for your kind attention :-)