Probabilistic Programming

Lecture #12: Hardness of Weakest Precondition Reasoning

Joost-Pieter Katoen

Software Modeling and Verification Chair | RWTH AACHEN UNIVERSITY

RWTH Lecture Series on Probabilistic Programming 2018

Overview

Overview

What we all know about termination

The halting problem
— does a program P terminate on a given input state s? — Σ_1
is semi-decidable.

The universal halting problem
— does a program P terminate on all input states? — Π_2
is undecidable.

Alan Mathison Turing
On computable numbers,
with an application to the Entscheidungsproblem
1937

Aim

Known fact: termination of ordinary programs is undecidable.

Π_2

Our aim is to classify "how undecidable"

(~~positive~~) almost-sure termination is. Π_3

(This is the topic of the next lecture.)

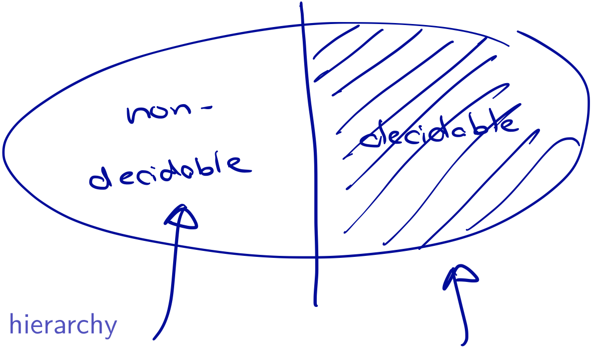
$=$

This lecture: how undecidable is it to compute weakest pre-expectations?

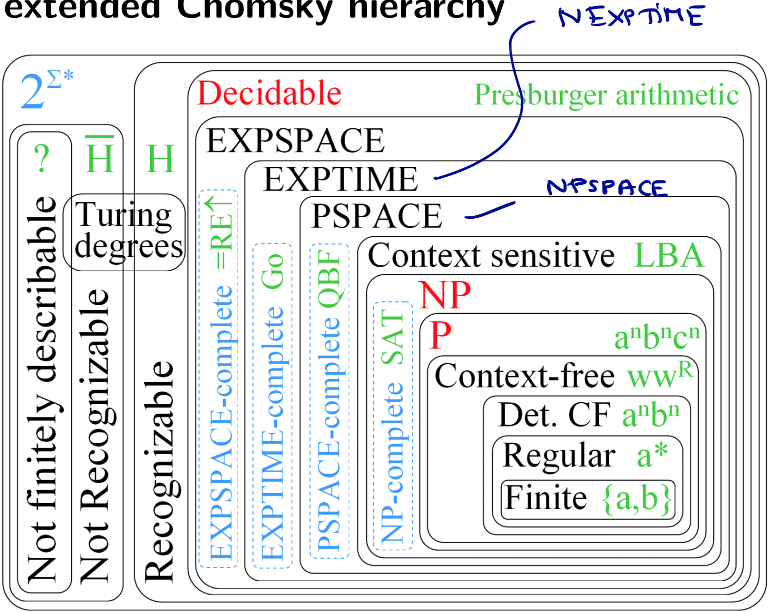
Lower bounds, upper bounds, exactly, or whether they are finite or not.

$wp(P, f) > q$? $wp(P, f) < q$

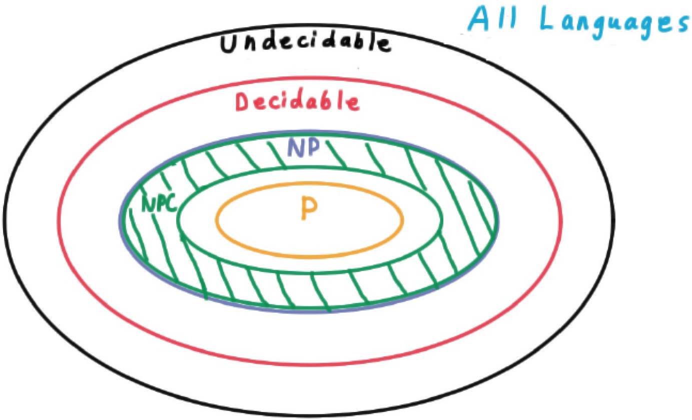
Overview



The extended Chomsky hierarchy

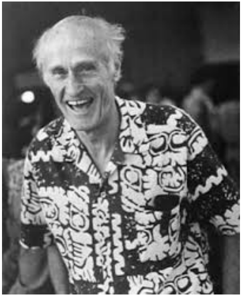


Undecidable versus decidable problems



How can we categorise the undecidable problems?

Kleene and Mostovski



Stephen Kleene (1909–1994)

Andrzej Mostovski (1913–1975)

Decision problems as formulas (1)

Idea: classify sets – ought to model decision problems – based on the complexity of characterising formulas in first-order Peano arithmetic.

Let H be the halting problem. The set H is defined for program P and input state s by:

$$(P, s) \in H \quad \text{iff} \quad \exists k \in \mathbb{N}. \exists s' \in \mathbb{S}. \, P \text{ terminates on input } s \text{ in } k \text{ steps in state } s'$$

or equivalently:

$$(P, s) \in H \quad \text{iff} \quad \underbrace{\exists k \in \mathbb{N}, s' \in \mathbb{S}.}_{\text{one quantifier}} \, P \text{ terminates on input } s \text{ in } k \text{ steps in state } s'$$

$H \in \Sigma_1$ as H can be defined by an existentially quantified formula of one level.

$$\underbrace{\exists \ldots \exists \ldots \exists \ldots}_{\exists} \quad \underbrace{\forall \ldots \forall \ldots}_{\forall}$$

Decision problems as formulas (1)

Idea: classify sets – ought to model decision problems – based on the complexity of characterising formulas in first-order Peano arithmetic.

Let H be the halting problem. The set H is defined for program P and input state s by:

$$(P, s) \in H \quad \text{iff} \quad \exists k \in \mathbb{N}.\, \exists s' \in \mathbb{S}.\, P \text{ terminates on input } s \text{ in } k \text{ steps in state } s'$$

or equivalently:

$$(P, s) \in H \quad \text{iff} \quad \underbrace{\exists k \in \mathbb{N}, s' \in \mathbb{S}.}_{\text{one quantifier}}\, P \text{ terminates on input } s \text{ in } k \text{ steps in state } s'$$

$H \in \Sigma_1$ as H can be defined by an existentially quantified formula of one level.

The level indicates the number of required quantifier alternations.

This is not the number of quantifiers as multiple quantifiers of the same type are contracted into one quantifier.

Decision problems as formulas (2)

Let UH be the universal halting problem. The set UH is defined for program P by:

$$P \in UH \quad \text{iff} \quad \forall s \in \mathbb{S}. \underbrace{(P, s) \in H}.$$

That is:

$$P \in UH \quad \text{iff} \quad \forall s \in \mathbb{S}. \overbrace{\big(\exists k \in \mathbb{N}, s' \in \mathbb{S}.\, P \text{ terminates on input } s \text{ in } k \text{ steps in state } s'\big)}$$

$UH \in \Pi_2$ as UH can be defined by a universally quantified formula of two alternations.

The arithmetical (Kleene-Mostovski) hierarchy

- Class Σ_n is defined as:

$$\Sigma_n = \{A \mid A = \{x \mid \exists y_1 \forall y_2 \exists y_3 \ldots \underbrace{\forall / \exists}_{\text{depending on } n \text{ odd or even}} y_n : (x, y_1, \ldots, y_n) \in R\}\}$$

where R is a decidable relation.

$n = 2$ $\exists y_1 \forall y_2$

$n = 3$ $\exists y_1 \forall y_2 \exists y_3$

The arithmetical (Kleene-Mostovski) hierarchy

- Class Σ_n is defined as:

$$\Sigma_n \;=\; \left\{ A \mid A = \{ x \mid \exists y_1 \forall y_2 \exists y_3 \ldots \forall/\exists y_n : (x, y_1, \ldots, y_n) \in R \} \right\}$$

where R is a decidable relation.

Example: the halting problem H is in Σ_1. It is semi-decidable.

- Class Π_n is defined as:

$$\Pi_n \;=\; \{ A \mid A = \{ x \mid \forall y_1 \exists y_2 \forall y_3 \ldots \forall/\exists y_n : (x, y_1, \ldots, y_n) \in R \} \}$$

where R is a decidable relation.

The arithmetical (Kleene-Mostovski) hierarchy

- Class Σ_n is defined as:

$$\Sigma_n \;=\; \{A \mid A = \{\, x \mid \exists y_1 \forall y_2 \exists y_3 \ldots \forall/\exists y_n : (x, y_1, \ldots, y_n) \in R \,\}\}$$

  where R is a decidable relation.

  Example: the halting problem H is in Σ_1. It is semi-decidable.

- Class Π_n is defined as:

$$\Pi_n \;=\; \{A \mid A = \{\, x \mid \forall y_1 \exists y_2 \forall y_3 \ldots \forall/\exists y_n : (x, y_1, \ldots, y_n) \in R \,\}\}$$

  where R is a decidable relation.

  Example: the universal halting problem UH is in Π_2.

$\underbrace{\forall s \in S.\ \exists}_{2} k \in \mathbb{N}, s', \ldots$

The arithmetical (Kleene-Mostovski) hierarchy

- Class Σ_n is defined as:

  $$\Sigma_n = \{A \mid A = \{ x \mid \exists y_1 \forall y_2 \exists y_3 \ldots \forall/\exists y_n : (x, y_1, \ldots, y_n) \in R \}\}$$

  where R is a decidable relation.

  Example: the halting problem H is in Σ_1. It is semi-decidable.

- Class Π_n is defined as:

  $$\Pi_n = \{A \mid A = \{ x \mid \forall y_1 \exists y_2 \forall y_3 \ldots \forall/\exists y_n : (x, y_1, \ldots, y_n) \in R \}\}$$

  where R is a decidable relation.

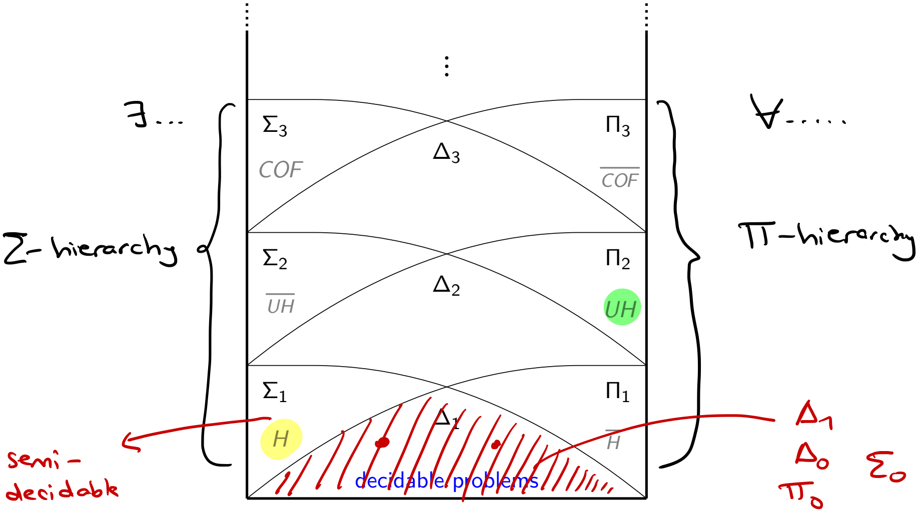
  Example: the universal halting problem UH is in Π_2.

- Let $\Delta_n = \Sigma_n \cap \Pi_n$. Δ_1 is the class of decidable problems.

The arithmetical hierarchy is used to classify the degree of undecidability.

The bigger picture

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



Elementary properties

- Classes Σ_0, Δ_0, Δ_1 and Π_0 coincide: decidable problems
- Classes Σ_n, Π_n and Δ_n are closed under conjunction and disjunction;Δ_n is closed under negation
- The classes Σ_n and Π_n are complementary $\longrightarrow$ negation

  $P \in \Sigma_n \implies \overline{P} \in \Pi_n$
- There is a strict inclusion relation between classes in the hierarchy:

$$\begin{tikzcd}
\Sigma_n \arrow[dr, phantom, "\subset" sloped] & & \Sigma_{n+1} \\
& \Delta_{n+1} \arrow[ur, phantom, "\subset" sloped] \arrow[dr, phantom, "\subset" sloped] & \\
\Pi_n \arrow[ur, phantom, "\subset" sloped] & & \Pi_{n+1}
\end{tikzcd}$$

Reducibility and completeness [Post 1944]

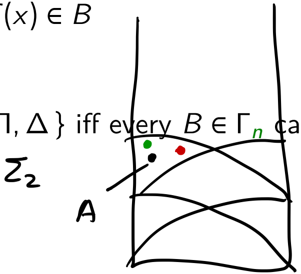
"Problem A is at least as hard as problem B"

- A set A is called arithmetical if $A \in \Gamma_n$ for some $\Gamma \in \{ \Sigma, \Pi, \Delta \}$ and $n \in \mathbb{N}$

- $A \subseteq X$ is reducible to $B \subseteq X$, denoted $A \leq_m B$, iff for some computable function $f : X \to X$ it holds:

$$\forall x \in X. \quad x \in A \quad \text{iff} \quad f(x) \in B$$

- Decision problem A is Γ_n-hard for $\Gamma \in \{ \Sigma, \Pi, \Delta \}$ iff every $B \in \Gamma_n$ can be reduced to A.



Reducibility and completeness [Post 1944]

"Problem A is at least as hard as problem B"

- A set A is called arithmetical if $A \in \Gamma_n$ for some $\Gamma \in \{ \Sigma, \Pi, \Delta \}$ and $n \in \mathbb{N}$
- $A \subseteq X$ is reducible to $B \subseteq X$, denoted $A \leq_m B$, iff for some computable function $f : X \to X$ it holds:

$$\forall x \in X. \quad x \in A \quad \text{iff} \quad f(x) \in B$$

- Decision problem A is Γ_n-hard for $\Gamma \in \{ \Sigma, \Pi, \Delta \}$ iff every $B \in \Gamma_n$ can be reduced to A.
- Decision problem A is Γ_n-complete if $A \in \Gamma_n$ and A is Γ_n-hard.

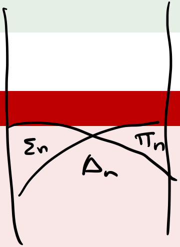
Completeness

Examples

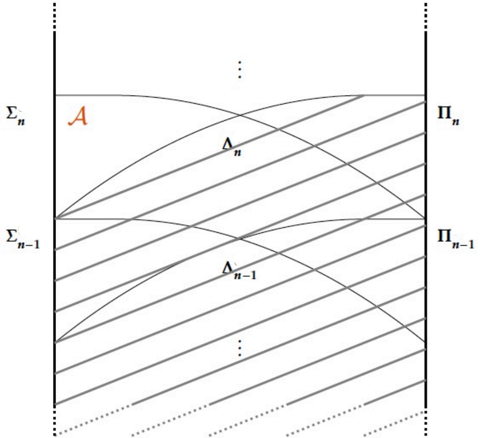
1. The halting problem is Σ_1-complete.
2. The universal halting problem is Π_2-complete.
3. The co-finiteness problem is Σ_3-complete.
4. If problem A is Σ_n-complete, then its complement is Π_n-complete. Analogous for Π_n-complete problems.

Davis' theorem

1. If problem A is Σ_n-complete, then $A \in \Sigma_n \setminus \Pi_n$
2. If problem A is Π_n-complete, then $A \in \Pi_n \setminus \Sigma_n$



Completeness



$\mathcal{A}$

Problem $\mathcal{A}$ is Σ_n-complete and hence sits properly at level n in the hierarchy.

It cannot be placed within the shaded area.

All indications in the previous picture of the arithmetical hierarchy are complete.

Co-finiteness problem

Co-finiteness problem

The co-finiteness problem is defined by:

$$P \in COF \quad \text{iff} \quad \{\, s \in \mathbb{S} \mid (P, s) \in H \,\} \quad \text{is co-finite}$$

It is the problem of deciding whether the set of inputs on which an ordinary program P terminates is co-finite.

The COF-problem is Σ_3 complete.

Overview

Hardness results in a nutshell

Checking lower bounds on expected outcomes is as hard as the halting problem.

$\underbrace{q < wp(P, f)\ ?}$ $\qquad$ H $\qquad$ Σ_1-complete

Hardness results in a nutshell

Checking lower bounds on expected outcomes is as hard as the halting problem.

Checking upper bounds is "more undecidable" than the halting problem.
It is as hard as the complement of the universal halting problem.

Π_2-complete

Σ_2-complete

$wp(P, f) < g$?

Hardness results in a nutshell

Checking lower bounds on expected outcomes is as hard as the halting problem.

Checking upper bounds is "more undecidable" than the halting problem.
It is as hard as the complement of the universal halting problem.

Determining $\underbrace{\text{exact expected outcomes}}$ is as hard as the universal halting problem.

$$wp(P, f) = g \; ?$$

$$\neg\,(wp(P,f) < g) \;\wedge\; \neg\,(wp(P,f) > g)$$

Hardness results in a nutshell

Checking lower bounds on expected outcomes is as hard as the halting problem.

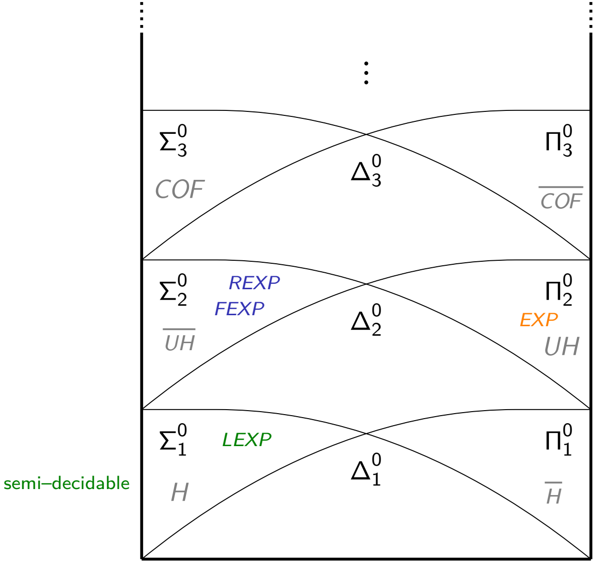
Checking upper bounds is "more undecidable" than the halting problem.
It is as hard as the complement of the universal halting problem.

Determining exact expected outcomes is as hard as the universal halting problem.

Determining whether an expected outcome is finite is as hard as obtaining upper bounds.

$wp(P, f) < \infty$?

Hardness of expected outcomes



Extended program configurations

pGCL $\longrightarrow$ Markov chain

Program configuration

An extended program configuration $\sigma = \langle P, s, n, \theta, q \rangle$ with:

- P is the program left to be executed or, $P = \downarrow$
- $s : Var \rightarrow \mathbb{Q}$ is the variable valuation
- $n \in \overset{\mathbb{N}}{\cancel{Nats}}$ is the number of computation steps the program has executed so far
- $\theta \in \{ L, R \}^*$ the history of all probabilistic choices made so far
- probability $q \in \mathbb{Q} \cap [0, 1]$, the probability of reaching configuration σ if probabilistic choices are resolved according to θ

The initial configuration of program P on input s is $\langle P, s, 0, \varepsilon, 1 \rangle$ where ε denotes the empty history.

The inference rules for pGCL are extended accordingly.

$$(P\,[r]\,Q,\ s) \longrightarrow r \cdot (P, s)$$
$$+\ (1-r) \cdot (Q, s)$$

is now going to be extended as follows

$$(P\,[r]\,Q,\ s,\ n,\ \theta,\ \underline{q})$$

$$\hookrightarrow\ (P,\ s,\ n+1,\ \theta L,\ \underline{q}\cdot r)$$

$\uparrow$ left choice

$$\hookrightarrow\ (Q,\ s,\ n+1,\ \theta R,\ \underline{q}\cdot(1-r))$$

pGCL without observe

+ without randomised assignments.

Distribution over final states

Distribution over final states

The distribution $[\![\, P \,]\!]_s$ of pGCL program P over final states on input s is defined by:

$$[\![\, P \,]\!]_s(t) = \sum_{\sigma \in \Sigma} q \quad \text{wheree} \quad \Sigma = \{\, \sigma = \langle \downarrow, t, n, \theta, q \rangle \mid \langle P, s, 0, \varepsilon, 1 \rangle \rightarrow^{*} \sigma \,\}$$

From now on, a pGCL program has no random assignments and no observe-statements.

Computable approximations of such distributions

1. The (sub-)distribution $[\![P]\!]_s^{=k}$ of pGCL program P over final states on input s after exactly k computation steps is defined by:

$$[\![P]\!]_s^{=k}(t) = \sum_{\sigma \in \Sigma} q \text{ with } \Sigma = \{ \sigma = \langle \downarrow, t, k, \theta, q \rangle \mid \langle P, s, 0, \varepsilon, 1 \rangle \rightarrow^* \sigma \}$$

2. The k-the approximation of the weakest pre-expectation $wp(P, f)$ is defined by:

$$wp(P, f)^{=k}(s) \;=\; \sum_{t \in \Sigma_P} [\![P]\!]_s^{=k}(t) \cdot f(t)$$

3. The computable weakest pre-expectations are defined by:

$$wp(P, f)(s) \;=\; \sum_{k=0}^{\infty} wp(P, f)^{=k}(s)$$

Decision problems on weakest pre-expectations

The decision problems *LEXP*, *REXP* and *EXP*

Let P be a pGCL program, $s \in \mathbb{S}$ a variable valuation, $q \in \mathbb{Q}_{\geq 0}$ and $f : \mathbb{S} \to \mathbb{Q}_{\geq 0}$ a computable function. Then:

$$
\begin{aligned}
(P, s, f, q) \in LEXP \quad &\text{iff} \quad q < wp(P, f)(s) \\
(P, s, f, q) \in REXP \quad &\text{iff} \quad q > wp(P, f)(s) \\
(P, s, f, q) \in EXP \quad &\text{iff} \quad q = wp(P, f)(s)
\end{aligned}
$$

Hardness of computing weakest pre-expectations

$$\begin{aligned} (P, s, f, q) \in LEXP \quad &\text{iff} \quad q < wp(P, f)(s) \\ (P, s, f, q) \in REXP \quad &\text{iff} \quad q > wp(P, f)(s) \\ (P, s, f, q) \in EXP \quad &\text{iff} \quad q = wp(P, f)(s) \end{aligned}$$

1. *LEXP* is Σ_1-complete, i.e., as hard as the halting problem.
2. *REXP* is Σ_2-complete, i.e., strictly harder than *LEXP*.
3. *EXP* is Π_2-complete, i.e., as hard as the universal halting problem.

Proof.

On the black board. □

Claim: LEXP is Σ_1 - complete

$q < wp(P, f)(s)$

① $\mathrm{LEXP} \in \Sigma_1$

$$(P, s, f, q) \in \mathrm{LEXP}$$

iff

$$q < wp(P, f)(s)$$

iff (* definition of wp *)

$$q < \sum_{k=0}^{\infty} wp^{=k}(P, f)(s)$$

iff

$$\underbrace{\exists y .\ q < \sum_{k=0}^{y} wp^{=k}(P, f)(s)}_{\Sigma_1\text{- formula}}$$

② LEXP is Σ_1 - hard.

$$r : H \mapsto \mathrm{LEXP}$$

$r: \quad H \mapsto LEXP$

$\uparrow$

GCL program Q
+ input state s

$$r(Q,s) = (P, s, 1, \tfrac{1}{2})$$

where $P = \texttt{skip}\ [\tfrac{1}{2}]\ Q$

Correctness of the reduction:

a. Q terminates on s. $(Q,s) \in H$.

Then P terminates on s with prob. 1.

As $\frac{1}{2} < 1$, it follows $(P, s, 1, \frac{1}{2}) \in LEXP$

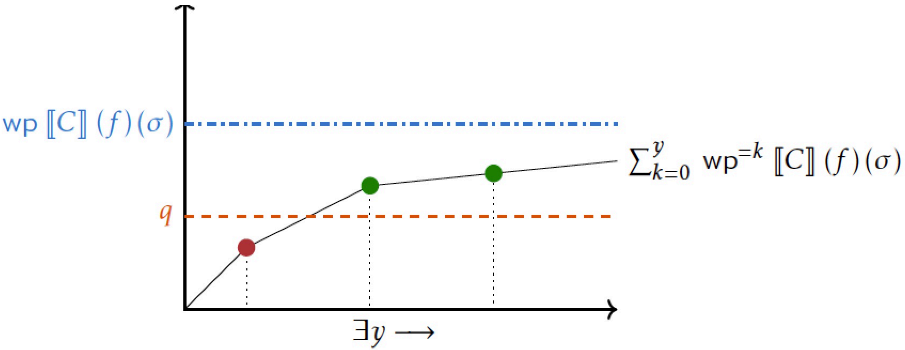
b. Q does not terminate on s. $(Q,s) \notin H$.

P terminates with probability $\frac{1}{2}$

$\frac{1}{2} \not< \frac{1}{2}$, $(P, s, 1, \frac{1}{2}) \notin LEXP$

$\boxtimes$.

Illustration of formula defining *LEXP*



Claim: REXP is Σ_2 - complete

① REXP $\in \Sigma_2$.

$$(P, s, f, q) \in REXP$$

iff (* def. of REXP *)

$$q > wp(P, f)(s)$$

iff (* def. of wp *)

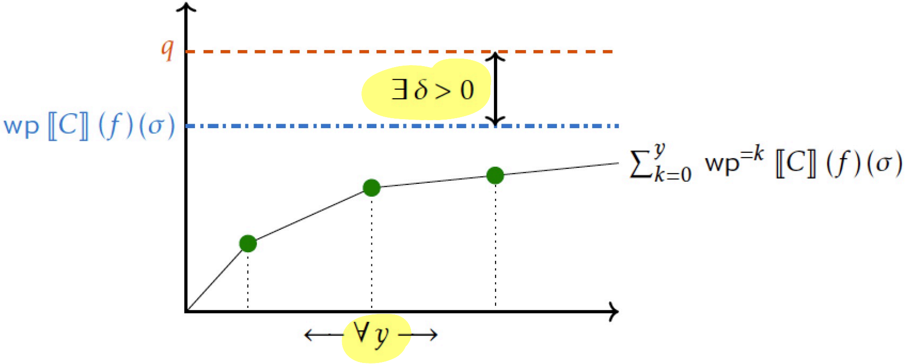
$$q > \sum_{k=0}^{\infty} wp^{=k}(P, f)(s)$$

iff

$$\underbrace{\exists \delta > 0.\ \forall y: \quad q - \delta > \sum_{k=0}^{y} wp^{=k}(P, f)(s)}_{\Sigma_2\text{- formula}}$$

② Σ_2- hard: $r : \overline{UH} \mapsto REXP$

Illustration of formula defining *REXP*



Finiteness of weakest pre-expectations

The finiteness decision problem *FEXP*

Let P be a pGCL program, $s \in \mathbb{S}$ a variable valuation, and $f : \mathbb{S} \to \mathbb{Q}_{\geq 0}$ a computable function. Then:

$$(P, s, f) \in \mathit{FEXP} \quad \text{iff} \quad \mathit{wp}(P, f)(s) < \infty.$$

FEXP is Σ_2-complete, i.e., as hard as the *REXP*-problem.

Complexity landscape of weakest pre-expectations