Post's Problem and The Priority Method in Synthetic Computability

Haoyi Zeng

Advisors: Yannick Forster and Dominik Kirst Supervisor: Prof. Gert Smolka Programming Systems Lab

Date:16.05.24







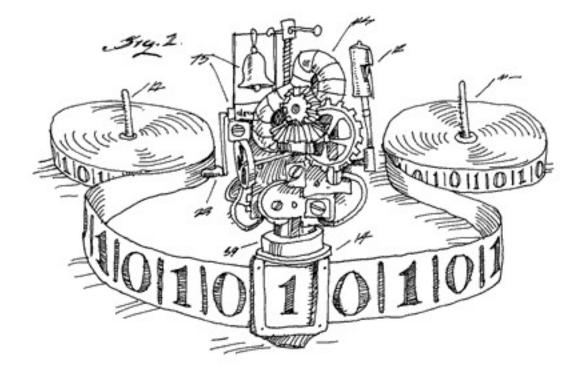
Synthetic Computability

P is **Decidable**: $\exists f: X \to \mathbb{B}$. $P x \leftrightarrow f x = \text{tt} \land f \text{ is computable}$

What is computable ?

Turing Machine

 λ -Calculus



fix $F := \lambda x$. fix' fix' F x $fix' := \lambda f, F. F (\lambda x. f f F x)$



Synthetic Computability

Synthetic Computability

A predicate $P: X \to \mathbb{P}$ is

Decidable

Semi-decidable

"Does a Turing machine halt on a given input?"



$\exists f: X \to \mathbb{B} . P x \leftrightarrow f x = \mathsf{tt}$ $\exists f: X \to \mathbb{N} \to \mathbb{B} . P x \leftrightarrow \exists n . f x n = \mathsf{tt}$

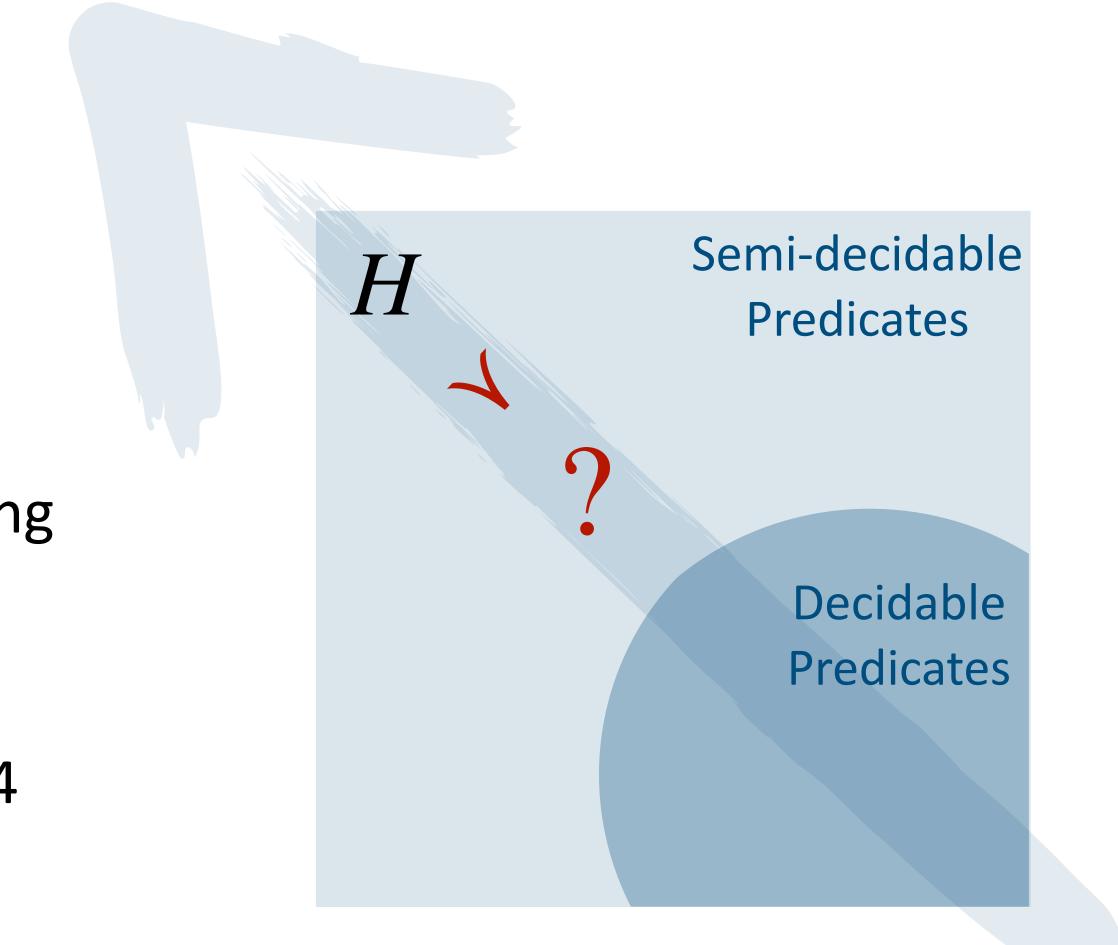
Halting Problem H

 $H x \leftrightarrow x$ -th partial function halts on x

Post's Problem

"Is there an undecidable, semi-decidable predicate that is strictly easier than the Halting problem?"

- Post, 1944

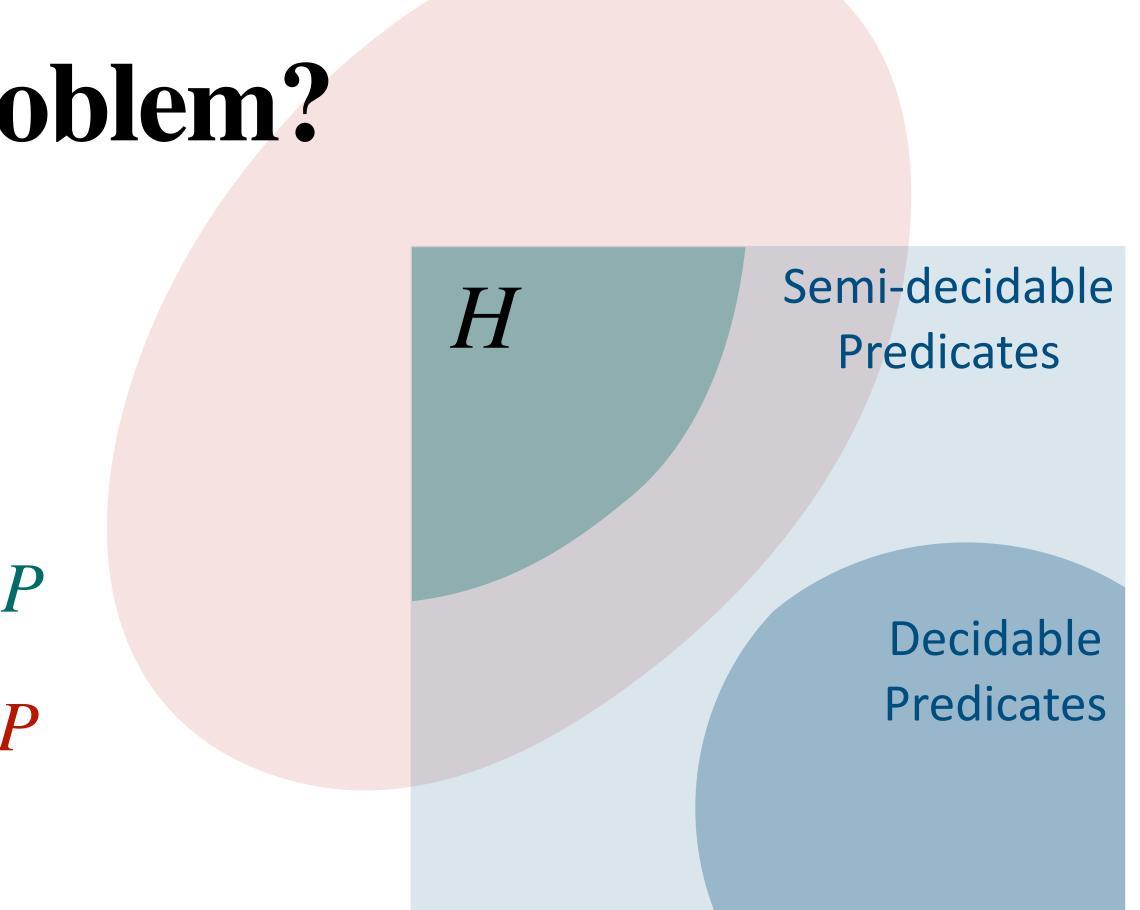


Easier than Halting Problem?

H is reducible to P

Many-one Reduction: $H \leq_m P$ Turing Reduction: $H \leq_T P$

Consider reductions in the most general sense, i.e., Turing reduction, which is also the problem Post left open in his paper.



Solutions to Post's Problem

Finite extension method [Post 1944]

Simple Predicate

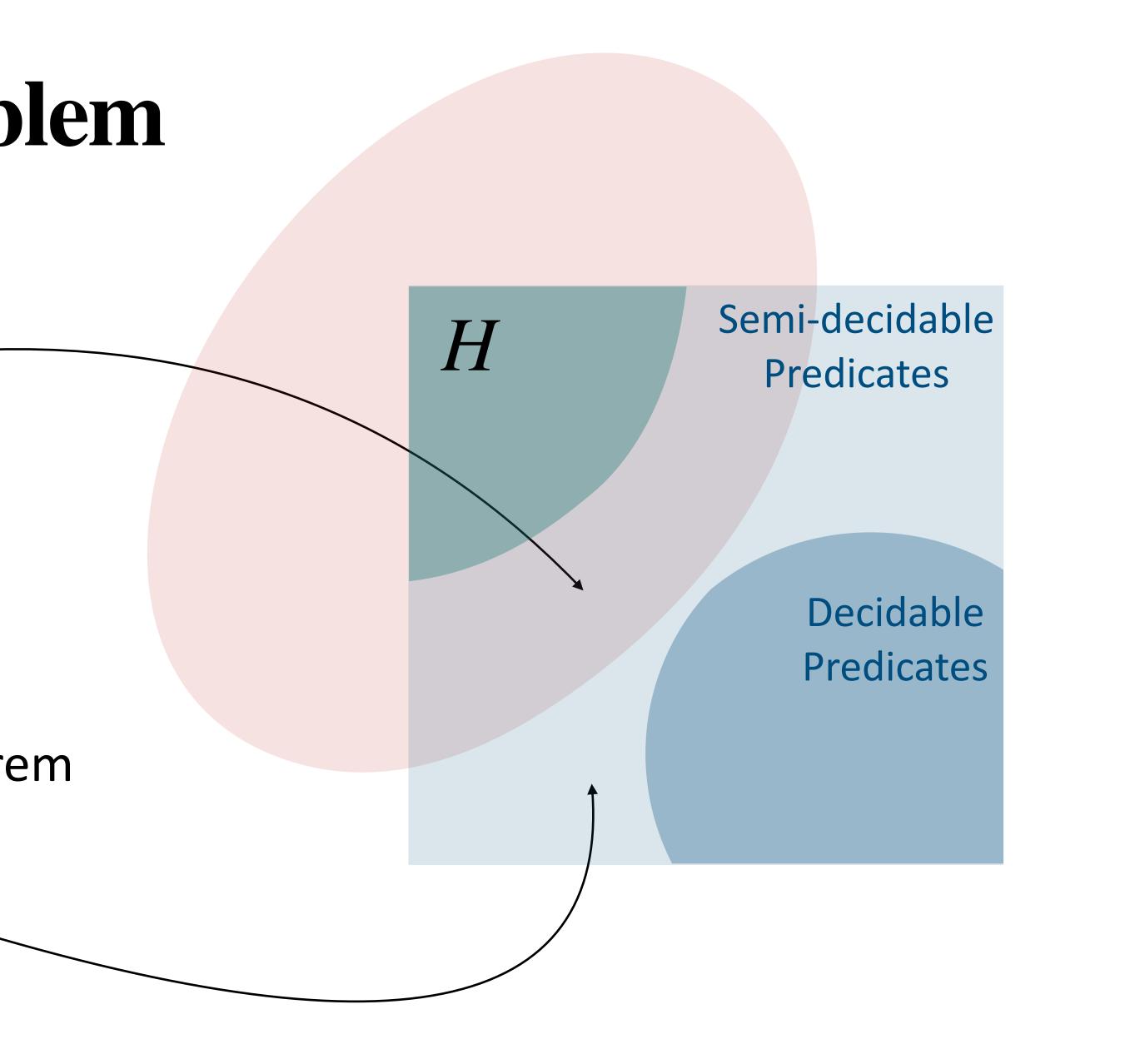
... After 12 years

The Priority Method Friedberg–Muchnik Theorem

[Mučnik 1956] [Friedberg 1957]

Low Simple Predicate

[Lerman & Soare 1980] [Soare 1999]



7

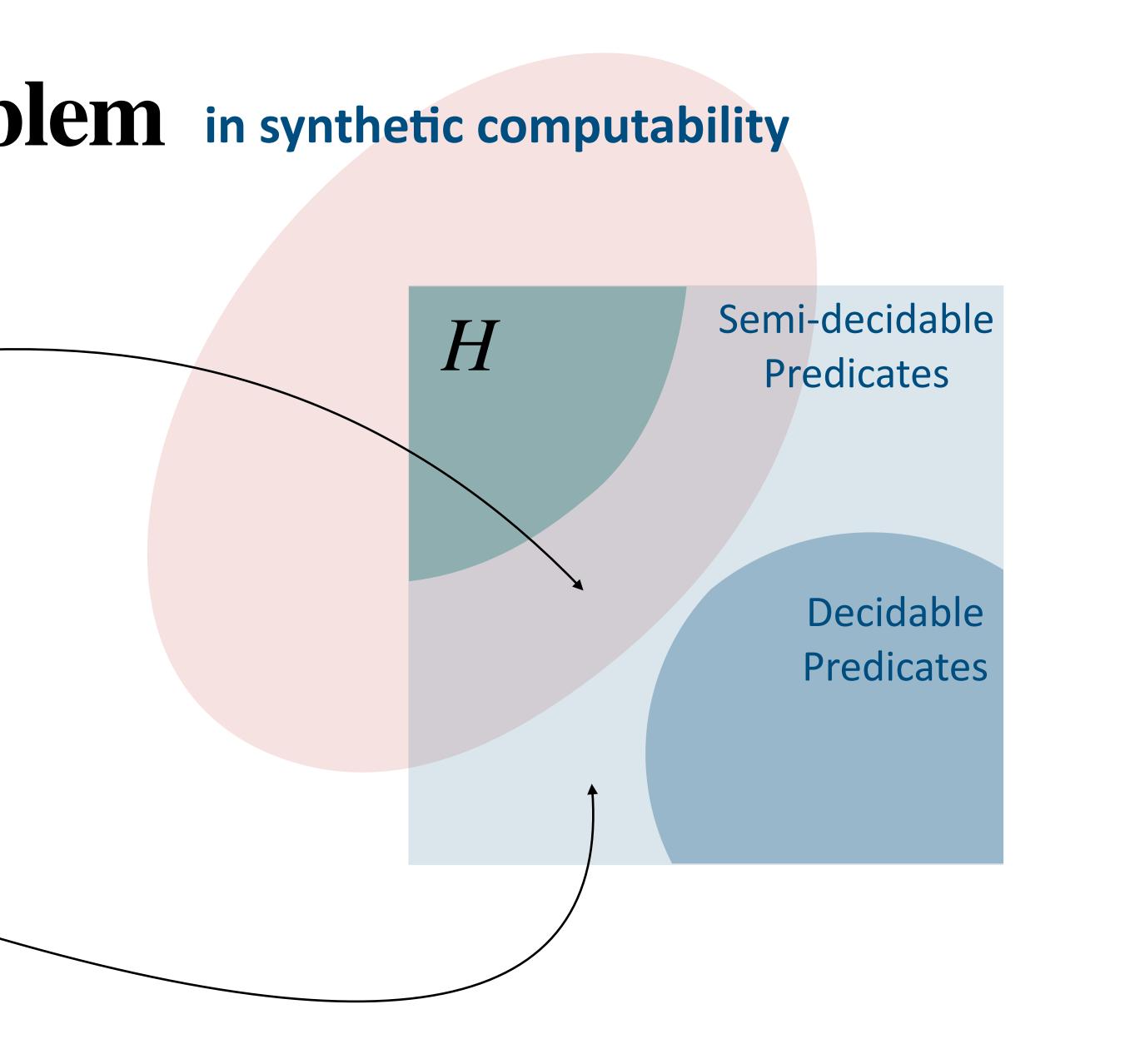
Solutions to Post's Problem in synthetic computability

in synthetic computability [Forster & Jahn 2023]

Simple Predicate

Another 12 years ?

Low Simple Predicate



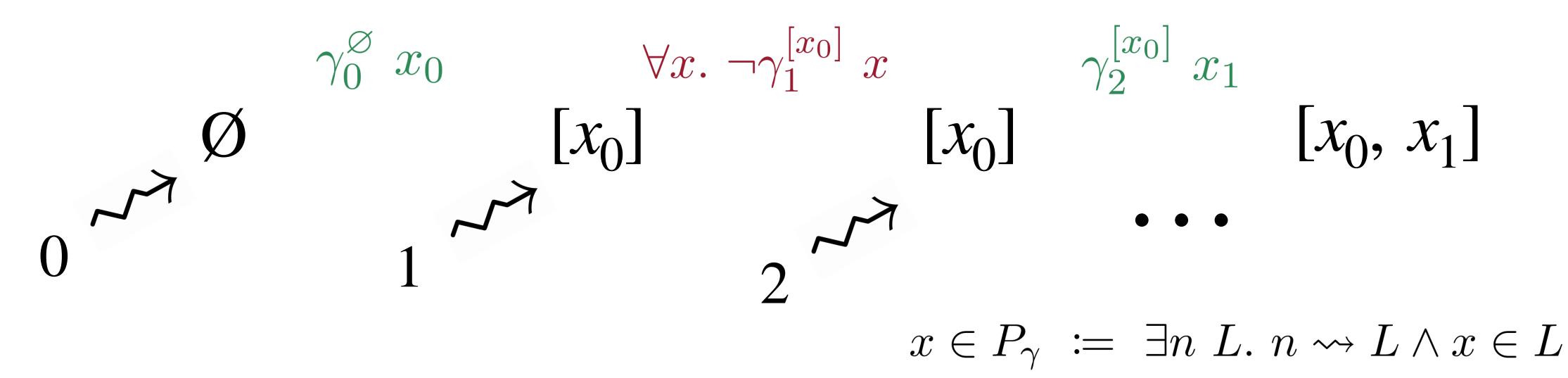
The Priority Method

The Priority Method

 $\gamma: \mathbb{N}^* \to \mathbb{N} \to \mathbb{N} \to \mathfrak{P}$

 $\frac{n \rightsquigarrow L \quad \gamma_n^L x}{0 \rightsquigarrow [\]} \qquad \frac{n \rightsquigarrow L \quad \gamma_n^L x}{n+1 \rightsquigarrow x :: L} \qquad \frac{n \rightsquigarrow L \quad \forall x. \neg \gamma_n^L x}{n+1 \rightsquigarrow L}$

The Priority Method



Let γ be an extension

 γ is computable and unique





Simple Extension

$$\pi_n^L e \ x \ \coloneqq \ x \in \mathcal{W}_e[n] \wedge \omega_n^L(e) < x$$

$$\Pi_n^L e \ \coloneqq \ L \cap \mathcal{W}_e[n] = \varnothing \wedge \exists x. \ \pi_n^L e \ x$$

$$\gamma_n^L x \ \coloneqq \ \exists e. \ e < n \land \mathcal{L} \ e. \ \Pi_n^L \land \mathcal{L} \ x. \ \pi_n^L \ e$$

$$P_1 \prec P_2 \prec P_3 \prec$$

Let ω be a wall

 ω is greater than 2e and convergent

$\omega:\mathbb{N}\to\mathbb{N}^*\to\mathbb{N}\to\mathbb{N}$

$P_4 \prec P_5 \prec P_6 \bullet \bullet \bullet$

P_{γ} is Simple

Low Wall

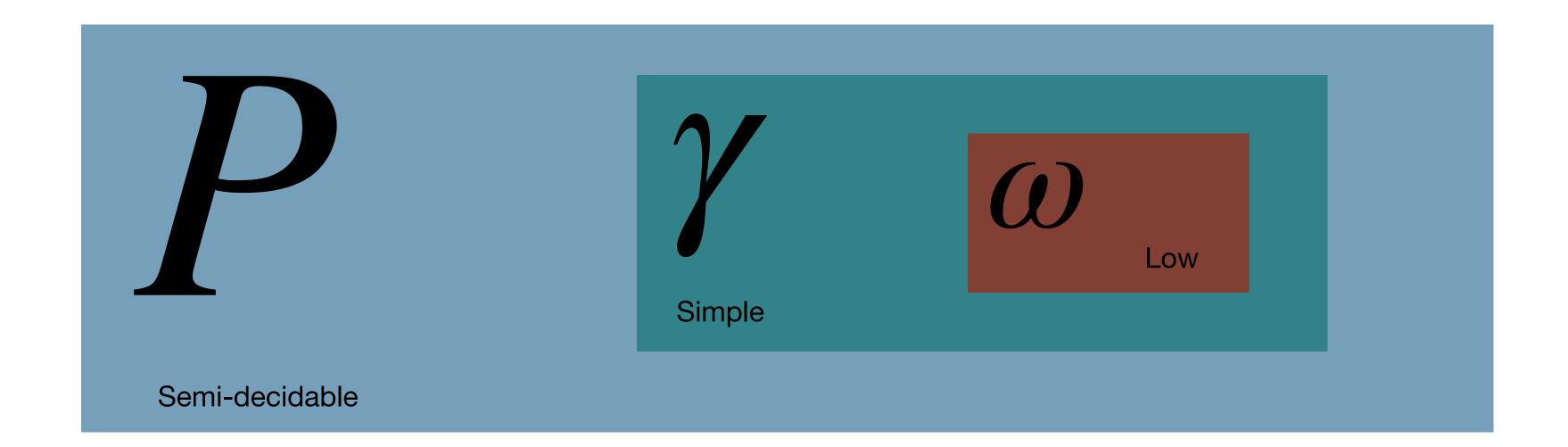
By the Limit Lemma

$\omega_n^L(e) := \max(2 \cdot e, \varphi_e^{x \in L}(e)[n]) \qquad N_e := \exists^\infty s. \ \Phi_e^A(e)[s] \downarrow \to \Phi_e^A(e) \downarrow$

$P_1 \prec N_1 \prec P_2 \prec N_2 \prec P_3 \prec N_3 \bullet \bullet \bullet$

 P_{γ_ω} is Low

Low Simple Predicate

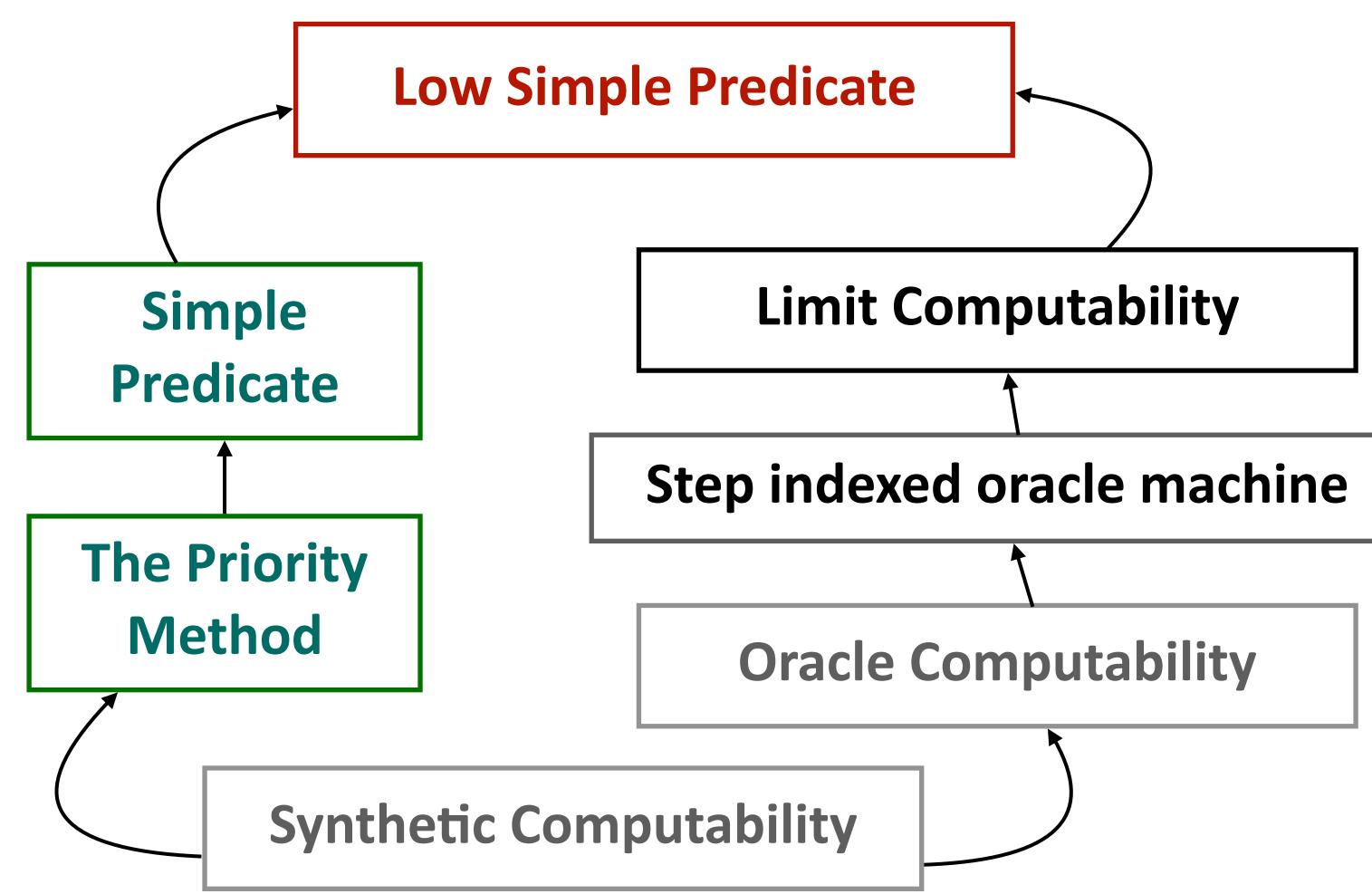




$P_{\gamma_{\omega}}$ is low simple predicate

Formalisation in Coq

Lines of code^{*}: ~ 900 Lines of code: ~ 1000 Lines of code: ~ 250



*: a technical lemma is missing

Goals (now)

We aim to show the following theorems and constructions in synthetic computability:

- Definition of limit computable
- Limit lemma
- The priority method
- Definition of low simple predicate
- Existence of low simple predicate
- Friedberg-Muchnik Theorem
- Constructive analysis







Goals (now)

We aim to show the following theorems and constructions in synthetic computability:

- Definition of limit computable
- Limit lemma
- The priority method
- Definition of low simple predicate
- Existence of low simple predicate
- Friedberg-Muchnik Theorem
- Constructive analysis





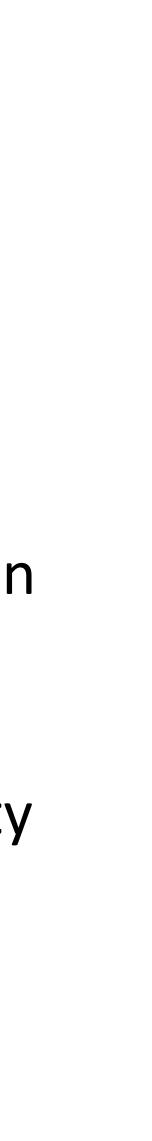


References

[Forster, Kirst & Mück 2023] Yannick Forster, Dominik Kirst and Niklas Mück. Oracle computability and Turing reducibility in the calculus of inductive constructions. *APLAS 2023*

[Post 1944] Post, E. L. Recursively enumerable sets of positive integers and their decision problems.

[Forster & Jahn 2023] Yannick Forster and Felix Jahn. Constructive and Synthetic Reducibility Degrees: Post's Problem for Many-one and Truth-table Reducibility in Coq. CSL 2023



References

[Friedberg 1957] Richard M. Friedberg. Two recursively enumerable sets of incomparable degrees of unsolvability (solution of Post's problem, 1944). Proceedings of the National Academy of Sciences of the United States of America. Vol. 43, no. 2. pp. 236–238.

[Lerman & Soare 1980] LERMAN, M., AND SOARE, R. d-simple sets, small sets, and degree classes. Pacific Journal of Mathematics 87, 1 (1980), 135–155.

[Soare 1999] SOARE, R. Recursively enumerable sets and degrees: A study of computable functions and computably generated sets. Springer Science & Business Media, 1999.



References

[Mučnik 1956] Mučnik Albert Abramovich On the unsolvability of the problem of reducibility in the theory of algorithms. Doklady Akademii Nauk SSSR. 108: 194–197.

[Forster, Kirst & Mück 2024] Yannick Forster, Dominik Kirst and Niklas Mück. The kleene-post and post's theorem in the calculus of inductive constructions CSL

[Shoenfield 1959] Shoenfield, J. R. On degrees of unsolvability. Annals of mathematics 69, 3 (1959), 644-653.

[Gold 1965] Gold, E. M. Limiting recursion. The Journal of Symbolic Logic 30, 1 (1965), 28-48







Appendix A

Oracle Computable

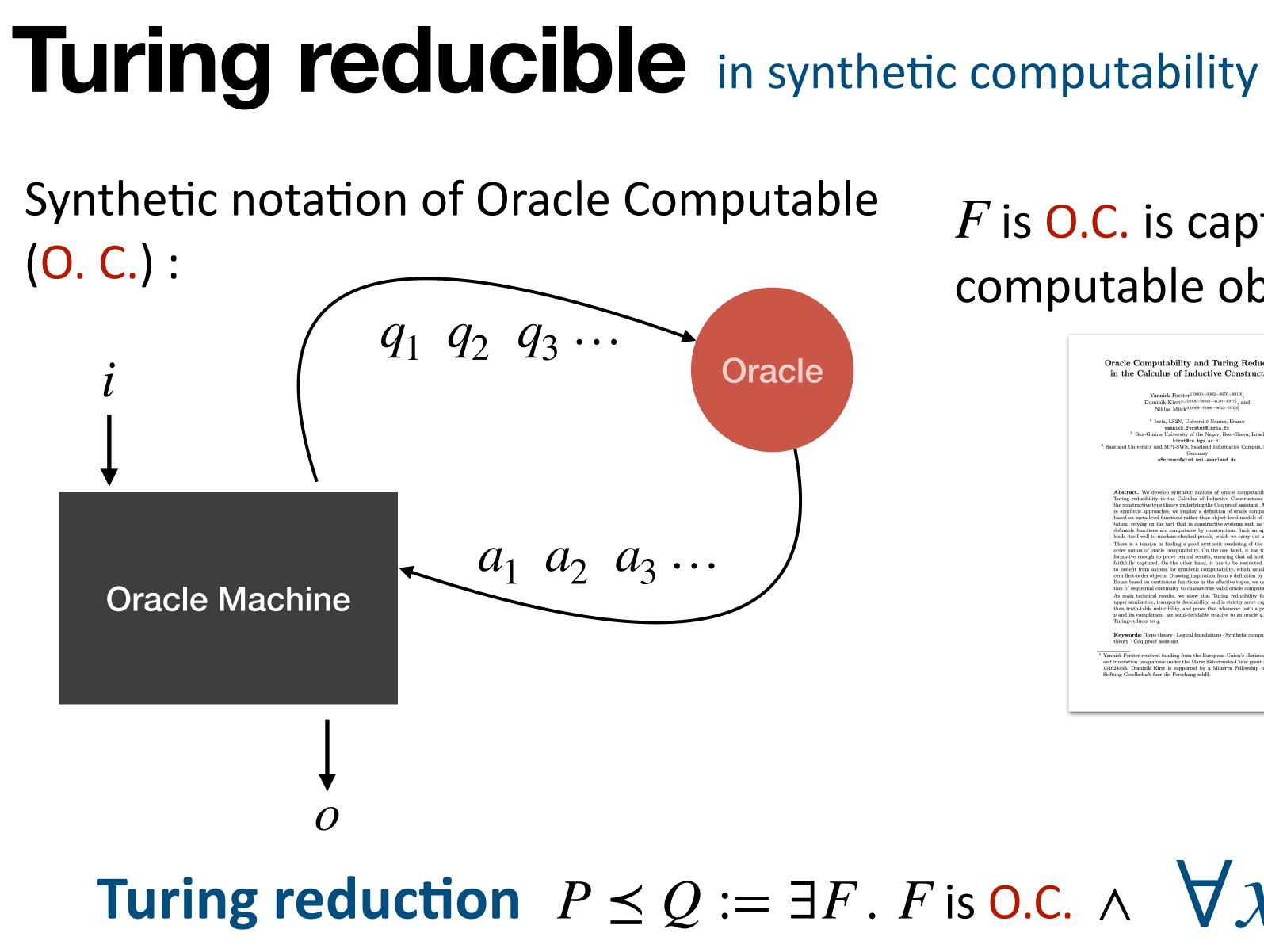
Based on a notion of computability of functionals $F : (Q \to A \to \mathbb{P}) \to (I \to Q \to \mathbb{P})$, The argument $R : Q \to A \to \mathbb{P}$ is to be read as the oracle relating questions q : Q to answers a : A, i : I is the input to the computation, and o : O is the output, such an F is considered (oracle)-computable if there is an underlying computation tree $\tau : I \to A^* \to (Q + O)$:

$$\forall R \ x \ b \ . F \ R \ x \ b \iff \exists qs \ as \ . \ \tau \ x; R \vdash qs; as \land \tau \ x \ as \triangleright \text{ out } b$$

where the interrogation relation $\sigma; R \vdash qs; as$ is inductively defined for $\sigma : A^* \rightharpoonup Q + O$ as:

$$\sigma ; R \vdash qs; as \quad \sigma \ as \triangleright ask \ q \quad R(q, a)$$
$$\sigma ; R \vdash qs @ [q]; as @ [a]$$

$$\sigma ; R \vdash []; []$$



F is O.C. is capturing by some underlying computable object [Forster, Kirst & Mück 2023]

Properties of Turing Reducibility

Oracle Computability and Turing Reducibility in the Calculus of Inductive Constructions*

Yannick Forster 1[0000-0002-8676-9819], Dominik Kirst
2,3[0000-0003-4126-6975], an Niklas Mück
3[0009-0006-9622-0762]

Inria, LS2N, Université Nantes, France yannick.forster@inria.fr

Ben-Gurion University of the Negev, Beer-Sheva, Israe kirst@cs.bgu.ac.il ty and MPI-SWS, Saarland Informatics s8nimuec@stud.uni-saarland.de

Abstract. We develop synthetic notions of oracle computability an Furing reducibility in the Calculus of Inductive Constructions (CIC), the constructive type theory underlying the Coq proof assistant. As usual a synthetic approaches, we employ a definition of oracle con s rather than object-level models of co ation, relying on the fact that in constructive systems such as CIC al finable functions are computable by construction. Such an app ends itself well to machine-checked proofs, which we carry out in Coq ere is a tension in finding a good synthetic rendering of the der notion of oracle computability. On the one hand, it has to As main technical results, we show that Turing reducibility forms an

n truth-table reducibility, and prove that whenever both a pr and its complement are semi-decidable relative to an oracle q, then p

Keywords: Type theory · Logical foundations · Synthetic com ory · Coq proof assistan

Yannick Forster received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101024493. Dominik Kirst is supported by a Minerva Fellowship of the Minerva Stiftung Gesellschaft fuer die Forschung mbH.

inue with similarly standard properties of Turing reducibility. Again is are concise but precise. As a preparation, we first note that Turing lity can be characterised without the relational layer. Lemma 20. $p \preceq_{\tau} q$ if and only if there is τ such that for all x and b we have $\hat{p}xb \leftrightarrow \exists qsas. \tau x ; q \vdash qs ; as \land \tau x as \lor out b.$ Now to begin, we show that Turing reducibility is a preorde Theorem 21. Turing reducibility is reflexive and transitive Proof. Reflexivity follows directly by the identity functional being compu-In fact, Turing reducibility is an upper semilat **Theorem 22.** Let $p: X \to \mathbb{P}$ and $q: Y \to \mathbb{P}$. Then there is a lowest upper bound $n+a: X+Y \to \mathbb{P}$ w.r.t. \leq_{T^*} . Let (p+q) We define oracle computability by observing that a terminating computation f(x) = f(x) + f(x) +

 $\begin{array}{l} p+q \colon X+Y \rightarrow \mathbb{P} \ w.r.t. \ \preceq_{\mathsf{T}} \colon Let \ (p\\ p+q \ is \ the \ join \ of \ p \ and \ q \ w.r.t\\ p \preceq_{\mathsf{T}} r \ and \ q \preceq_{\mathsf{T}} r \ then \ p+q \preceq_{\mathsf{T}} r. \end{array}$ Proof. The first two claims follow et F_1 reduce p to r and be compute r₂. Define

 $FR\,z\,o:=\begin{cases}F_1R\,x\,o & \text{if }z=\text{inl }x\\F_2R\,x\,o & \text{if }z=\text{inr }y\end{cases}$ τ computes F, and F reduces p + q t We continue by establishing the non-relativised notion of decid

with oracles has a sequential form: in any step of the sequence, the oracle compu tation can ask a question to the oracle, return an output, or diverge. Informally we can enforce such sequential behaviour by requiring that every terminatin computation FBio can be described by (finite, possibly empty) lists as: Q^* and $as: \hat{A}^*$ such that from the input *i* the output *o* is eventually obtained after finite sequence of steps, during which the questions in qs are asked to the oracle one-by-one, yielding corresponding answers in as. This computat tional data ca be captured by a partial³ function of type $I \rightarrow A^* \rightarrow Q + O$, called the (computed by $A^* \rightarrow Q + O$). tation) tree of F, that on some input and list of previous answers either return the next question to the oracle, returns the final output, or diverges. So more formally, we call $F: (Q \to A \to \mathbb{P}) \to (I \to O \to \mathbb{P})$ an (oracle-)computable functional if there is a tree $\tau{:}\,I{\rightarrow}A^*{\rightharpoonup}Q+O$ such that

 $\forall R \, i \, o. \ FR \, i \, o \ \leftrightarrow \ \exists qs \ as. \ \tau i \ ; R \vdash qs \ ; as \ \land \ \tau \, i \ as \triangleright \mathsf{out} \ o$ with the interrogation relation $\sigma; R \vdash qs; as$ being defined inductively by

 $\sigma ; R \vdash qs ; as \qquad \sigma as \triangleright \mathsf{ask} \ q \qquad Rqa$

 $\overline{\sigma: R \vdash []; []}$ σ ; $R \vdash qs \# [q]$; as # [a]

where A^* is the type of lists over $a,\,l\!+\!l'$ is list concatenation, where we use the suggestive shorthands ask q and out o for the respective injections into the sum type Q + O, and where $\sigma: A^* \rightarrow Q + O$ denotes a tree at a fixed input *i*. To provide some further intuition and visualise the usage of the word "tree we discuss the following example functional in more detail

> $F : (\mathbb{N} \to \mathbb{B} \to \mathbb{P}) \to (\mathbb{N} \to \mathbb{B} \to \mathbb{P})$ $FRio := o = true \land \forall a < i, Ra true$

Turing reduction $P \leq Q := \exists F. F \text{ is O.C. } \land \forall X \cdot P x \leftrightarrow F \hat{Q} x \text{ tt}$ $\neg P x \leftrightarrow F \hat{Q} x \text{ ff}$

Appendix B Step index function

We insert this oracle O into our Turing machine by fixing a n, and subsequently run τ . Based on this effectively computable oracle, we can define a total function Φ as follows:

$$\Phi_{\tau}^{O(n)} x \, i \, j := \begin{cases} \lceil \text{out } o \rceil & \text{if } (\tau \, x \, []) \rightsquigarrow_{j} \text{out } o \\ \lceil \text{ask } q \rceil & \text{if } (\tau \, x \, []) \rightsquigarrow_{j} \text{ask } q \text{ and } i = 0 \\ \Phi_{\tau @ [\chi_{O} \, n \, q]}^{O(n)} x \, i' \, j & \text{if } (\tau \, x \, []) \rightsquigarrow_{j} \text{ask } q \text{ and } i = S \, i' \\ \text{none} & \text{otherwise} \end{cases}$$

Given that *P* is Turing reducible to \emptyset , we obtain the computable tree τ . Building upon the step-index function described above, we define the following function:

$$\chi_P(s,x) \coloneqq \begin{cases} b & \text{if } \Phi_{\tau}^K(x)[s] = \lceil b \rceil \\ \text{tt } & \text{otherwise} \end{cases}$$

Low Simple Predicate 1

Simple predicate: If a predicate P is simple, then P is semi-decidable and $\neg (P \leq \emptyset)$

Turing jump of *P*: $P' x \leftrightarrow x$ -th oracle machine with oracle P halts on x

undecidable predicate

Low Simple Predicate 2

Low Simple predicate: $\emptyset \prec P \prec K$, where $P \prec Q := P \leq Q \land \neg (P \leq Q)$



- **Low predicate**: A predicate P is low, if the Turing jump of P is reducible to K $P' \leq K \Rightarrow \neg (K \leq P)$

 - A positive solution to Post's Problem
 - Showing a predicate is reducible to K is difficult!

Use Function

Let
$$k = \varphi_e^A(e)[n]$$

 $\Phi_e^p(e)[n] = \ulcorner \star \urcorner \to \forall q. \ q \equiv_k p[n] \to \Xi_e \ \hat{q} \ e$
 $\varphi_e^p(x)[n] = k \to p[n] \equiv_k p[n+1] \to \varphi_e^p(x)[n+1]$

e *

= k