Post's Problem and The Priority Method in CIC

Haoyi Zeng, Yannick Forster, Dominik Kirst

TYPES 2024

Date:10.06.24

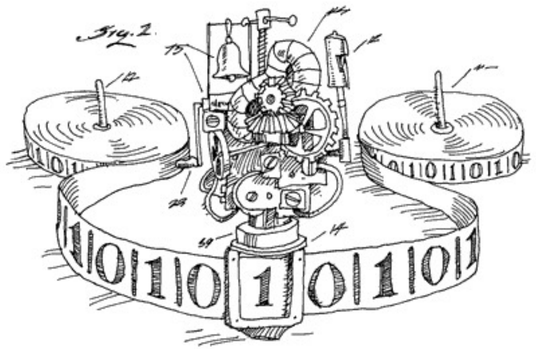
Background

Synthetic Computability [Richman 1983] [Bauer 2006]

P is **Decidable**: $\exists f : X \to \mathbb{B} \,.\, P\ x \leftrightarrow f\ x = \mathsf{tt}$ ~~$\wedge$ f is computable~~

What is computable ?

Turing Machine



λ-Calculus

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

Synthetic Computability

Synthetic Computability

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

Decidable $\quad \exists\, f : X \to \mathbb{B}\,.\, P\ x \leftrightarrow\ f\ x = \text{true}$

Semi-decidable $\quad \exists\, f : X \to \mathbb{N} \to \mathbb{B}\,.\, P\ x \leftrightarrow \exists n\,.\, f\ x\ n = \text{true}$

For any $P : X \to \mathbb{P}$ and $Q : Y \to \mathbb{P}$

Many-one Reduction

$$P \preceq_m Q \ :=\quad \exists f : X \to Y\,.\, \forall x\,.\, P\ x \leftrightarrow Q\ (f\ x)$$

Church's Thesis

There is an enumerator $\theta : \mathbb{N} \to (\mathbb{N} \rightharpoonup \mathbb{N})$ enumerates all partial functions:

$$\forall f : \mathbb{N} \rightharpoonup \mathbb{N}.\ \exists e.\ \forall x\ v.\ f\ x \downarrow v \leftrightarrow \theta_e\ x \downarrow v$$

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

Halting Problem $H\ x\ :=\ \theta_x\ x \downarrow$

Why CIC ? $\mathrm{CIC}\ +\ \mathrm{CT}\ +\ \mathrm{LEM}$ is consistent

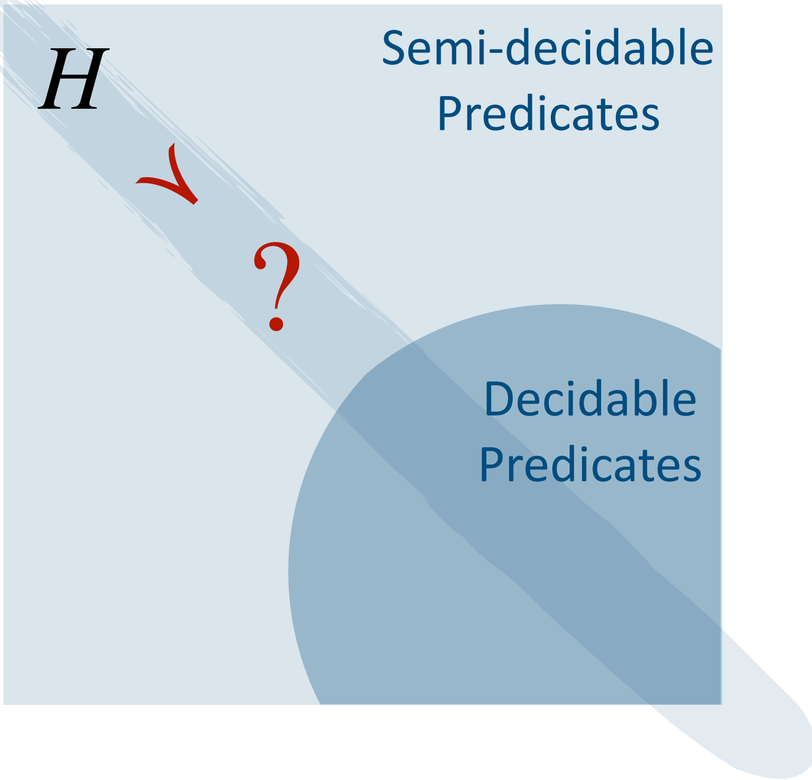
Post's Problem

"Is there an undecidable,

semi-decidable predicate

that is strictly easier than the Halting problem?"

- Post, 1944

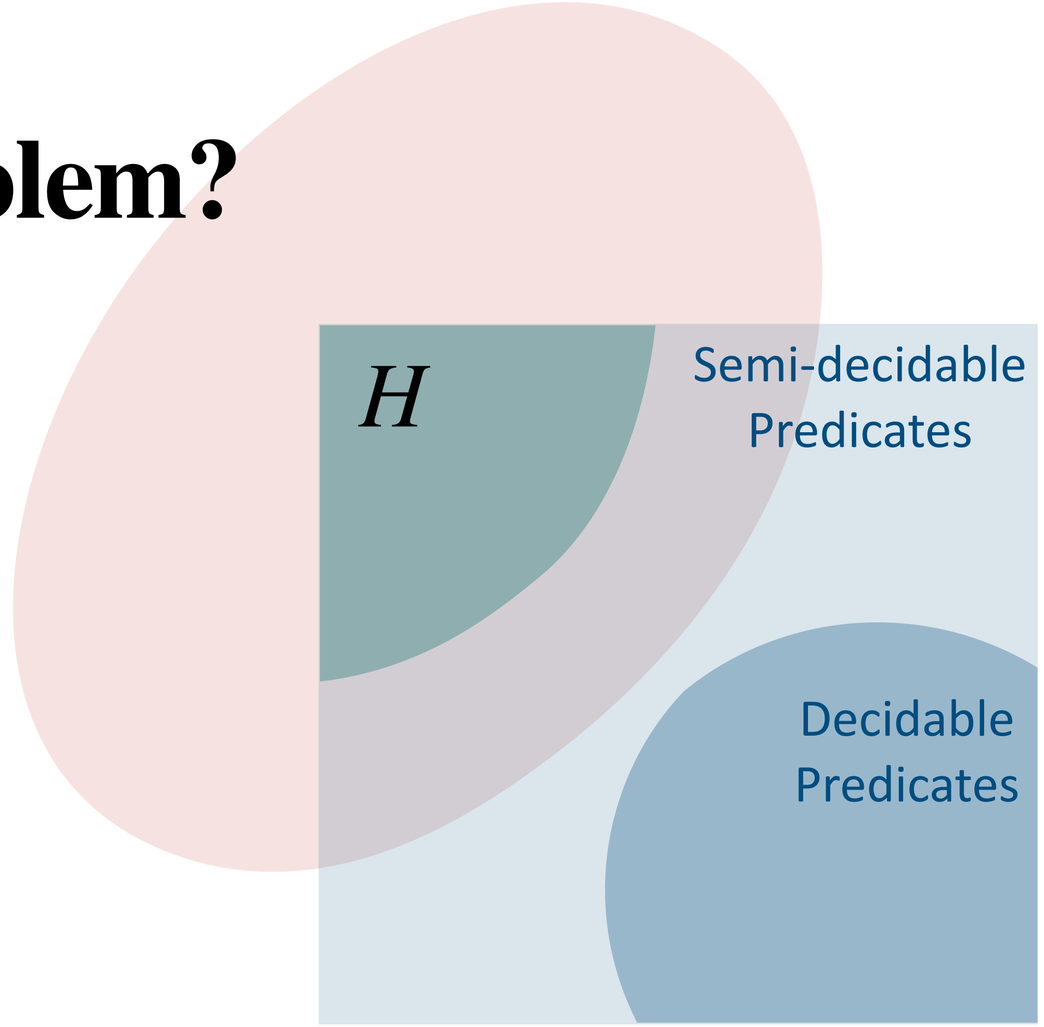


Easier than Halting Problem?

P is reducible to Q

Many-one Reduction: $H \leq_m P$

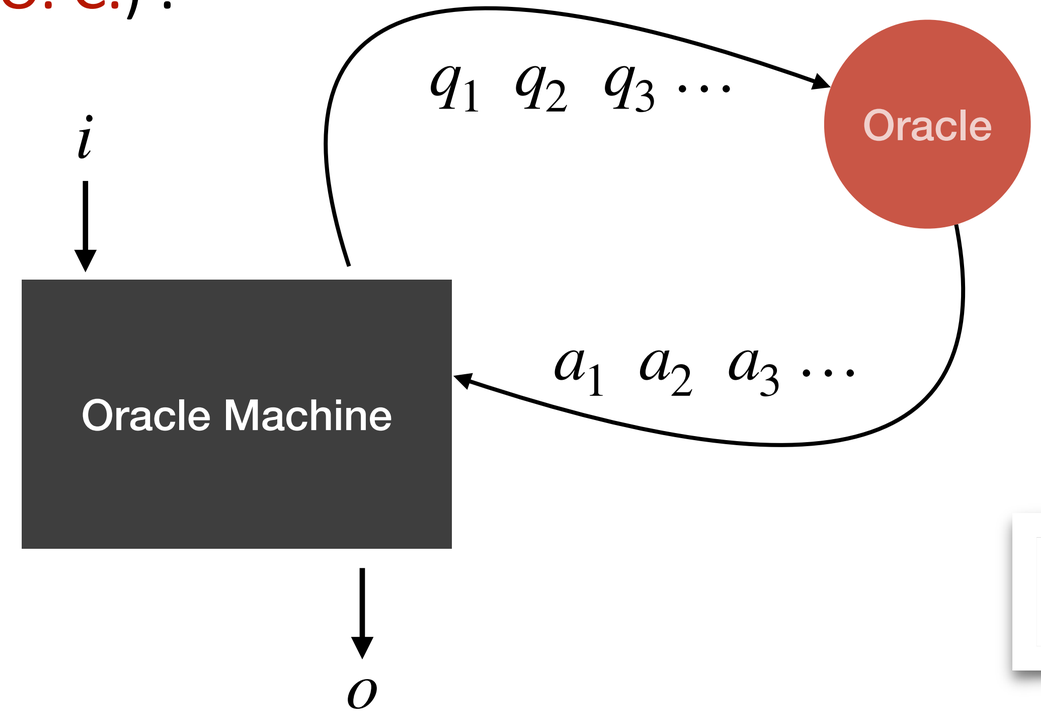
Turing Reduction: $H \leq_T P$



Consider reductions in a more general sense, i.e., Turing reduction, which is also the problem Post left open in his paper [Post 1944].

Turing Reducible in synthetic computability

Synthetic notation of Oracle Computable (O. C.) :



F is O.C. is capturing by some underlying computable object:

- The first definition of oracle computability by modulus continuity [Bauer 2012]
- A definition of oracle computability by sequential continuity [Forster, Kirst & Mück 2023]

14:20 *Andrew Swan*
Oracle modalities (abstract)

[Swan 2024]

Turing reduction $P \preceq Q := \exists F\,.\ F$ is O.C. $\wedge\ \forall x\,.\ \begin{matrix} P\ x \leftrightarrow F\ \hat{Q}\ x\ \text{true} \\ \neg P\ x \leftrightarrow F\ \hat{Q}\ x\ \text{false} \end{matrix}$

Solutions to Post's Problem

Finite extension method [Post 1944]

Simple Predicate

. . . **After 12 years**

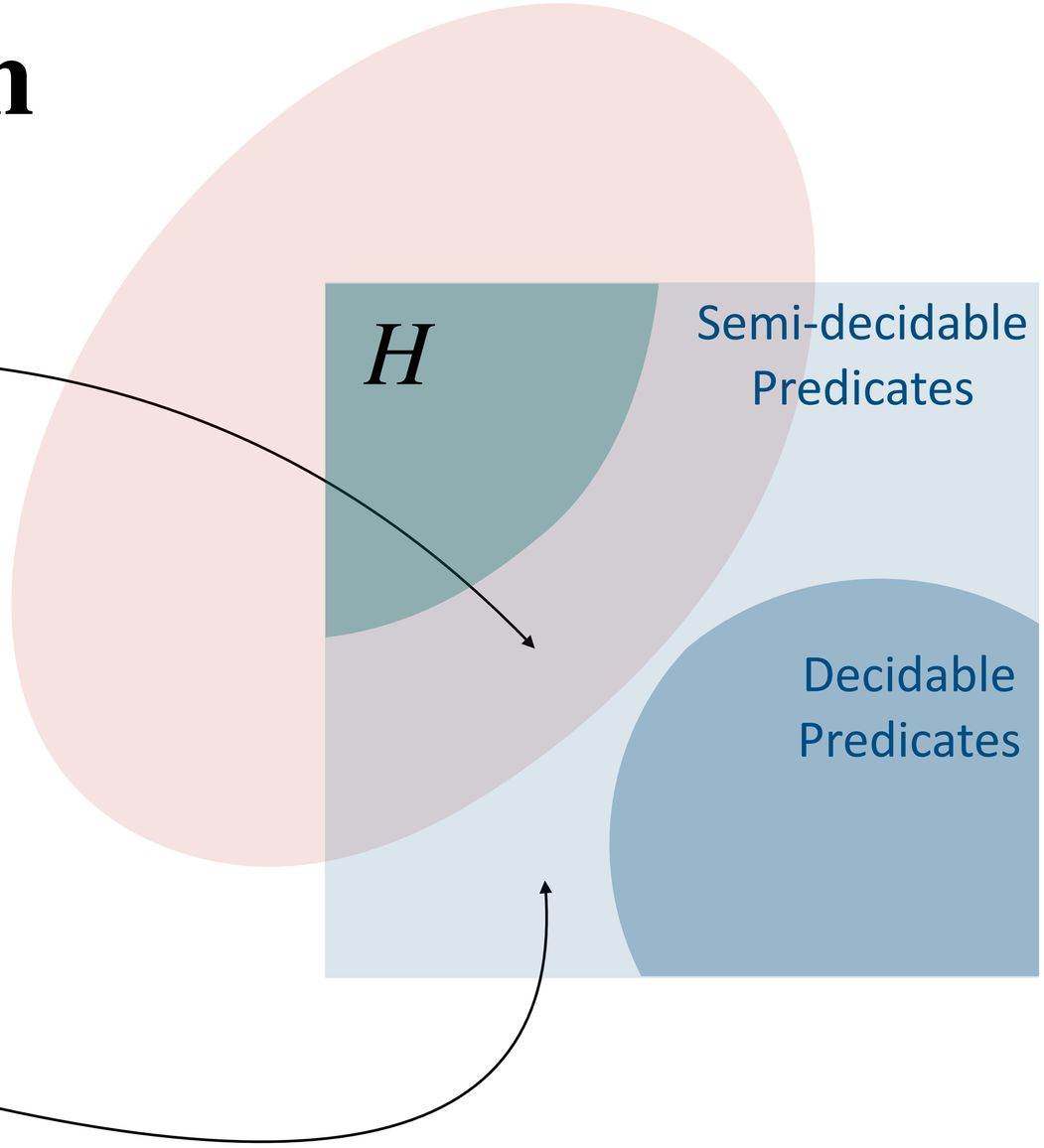
The Priority Method

Friedberg–Muchnik Theorem

[Mučnik 1956] [Friedberg 1957]

Low Simple Predicate

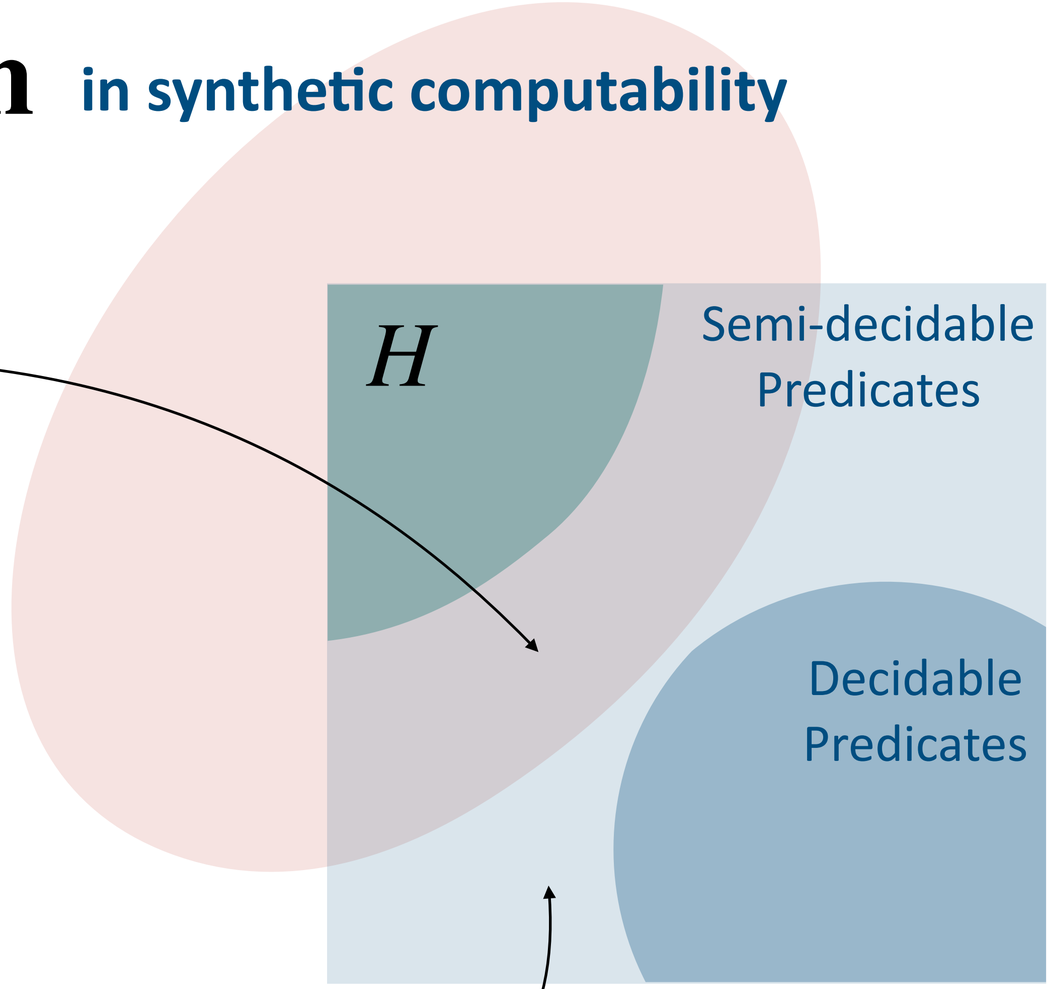
[Lerman & Soare 1980] [Soare 1999]



Solutions to Post's Problem in synthetic computability

in synthetic computability
[Forster & Jahn 2023]

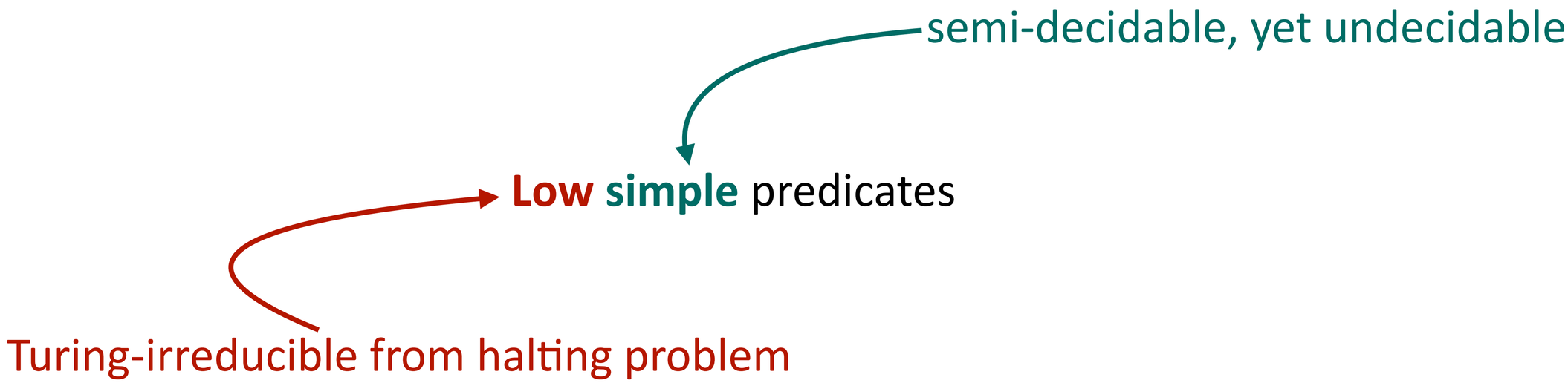
Simple Predicate



~~**Another 12 years ?**~~

Now **Low Simple Predicate**

The Priority Method



In synthetic computability, the priority method is as “hard” as in any textbook

Low Simple Predicate

P_{γ_ω} is a semi-decidable low simple predicate

The Priority Method

$\gamma : \mathbb{N} \to \mathbb{N}^* \to \mathbb{N} \to \text{Prop}$

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

$$x \in P_\gamma \;:=\; \exists n\ L.\ n \rightsquigarrow L \wedge x \in L$$

γ is computable and functional (Extension) $\Rightarrow$ P_γ is **semi-decidable**

Low Simple Predicate

P_{γ_ω} is a **semi-decidable** **predicate**

Simple Extension

$$\begin{aligned}
\pi_n^L\ e\ x &\coloneqq x \in \mathcal{W}_e[n] \wedge \omega_n^L(e) < x \\
\Pi_n^L\ e &\coloneqq L \cap \mathcal{W}_e[n] = \varnothing \wedge \exists x.\ \pi_n^L\ e\ x \\
\gamma_n^L\ x &\coloneqq \exists e.\ e < n \wedge \mathcal{L}\ e.\ \Pi_n^L \wedge \mathcal{L}\ x.\ \pi_n^L\ e
\end{aligned}$$

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

$$P_1 \prec P_2 \prec P_3 \prec P_4 \prec P_5 \prec P_6 \bullet\bullet\bullet$$

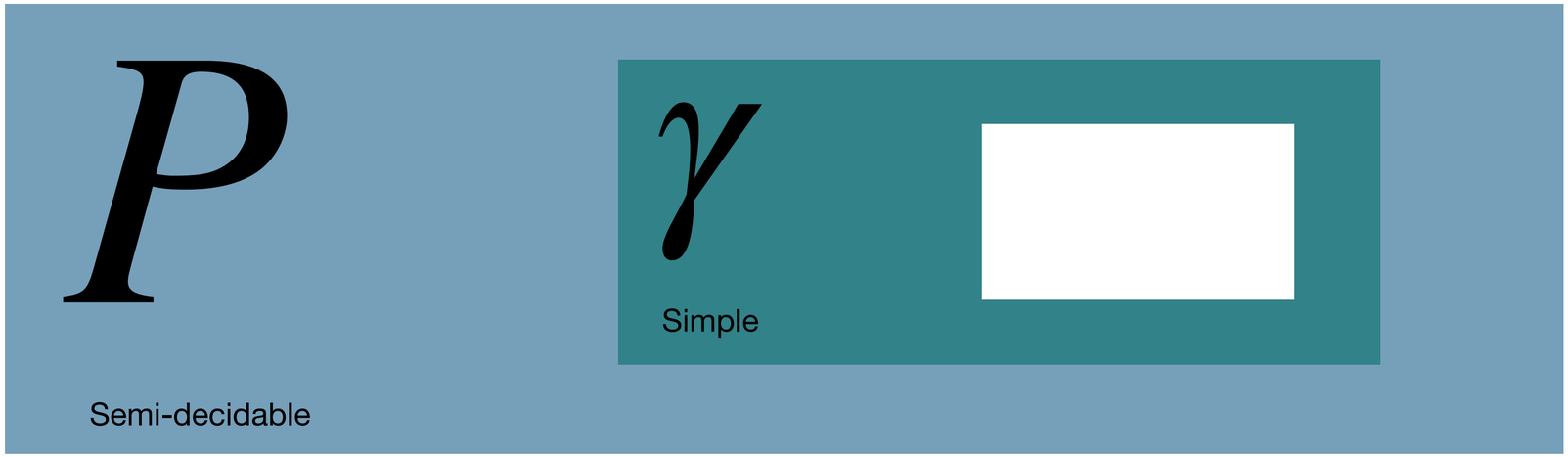
ω is greater than $2e$
and convergent (Wall)

$\Rightarrow$

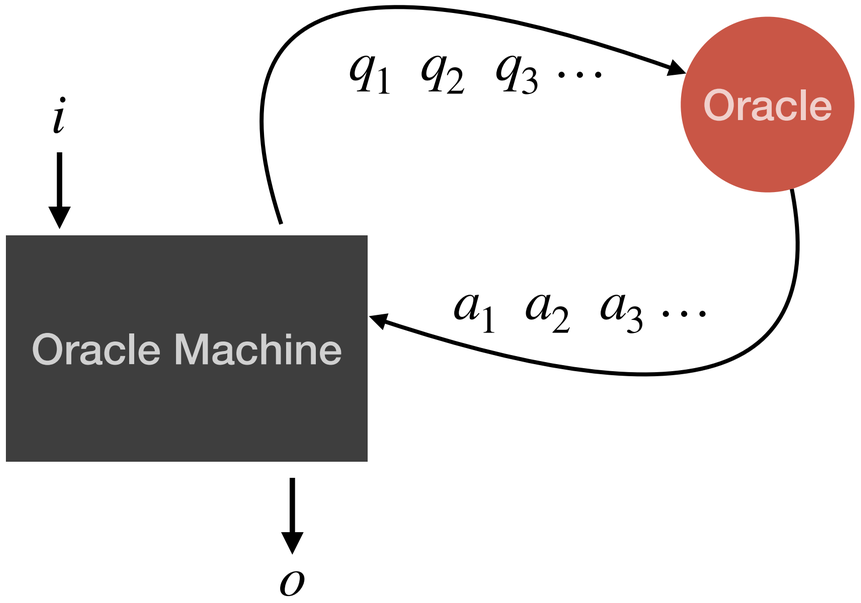
P_γ is **Simple**

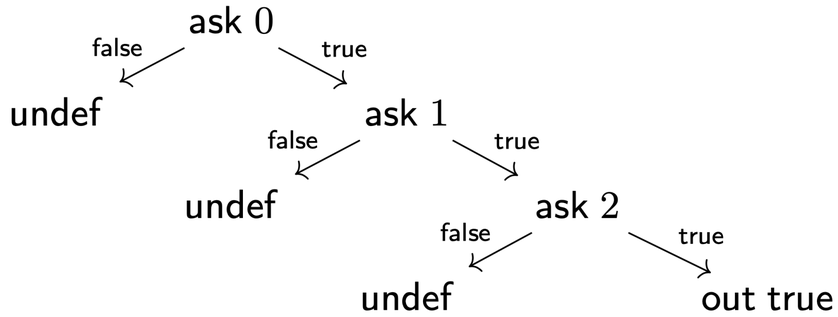
Low Simple Predicate

P_{γ_ω} is a **simple** **predicate**



Use Function





$$\varphi_e^p(x)[n] := \max(\text{ List of questions }) + 1$$

Low Wall

$$\omega_n^L(e) \;:=\; \max(2 \cdot e, \varphi_e^{x \in L}(e)[n])$$

~~Some removed formulas~~

~~Some removed formulas~~ ~~Some removed formulas~~

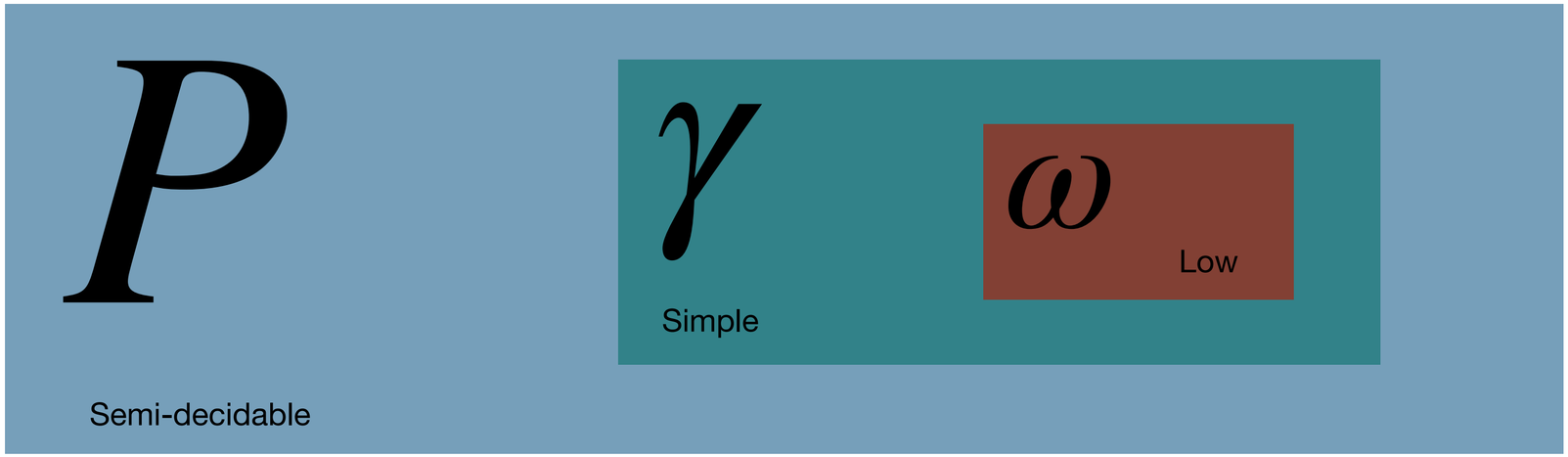
~~Some removed formulas~~ ~~Some removed formulas~~ ~~Some removed formulas~~

~~Some removed formulas~~

P_{γ_ω} is **Low**

Low Simple Predicate

P_{γ_ω} is a **low** **simple** **predicate**



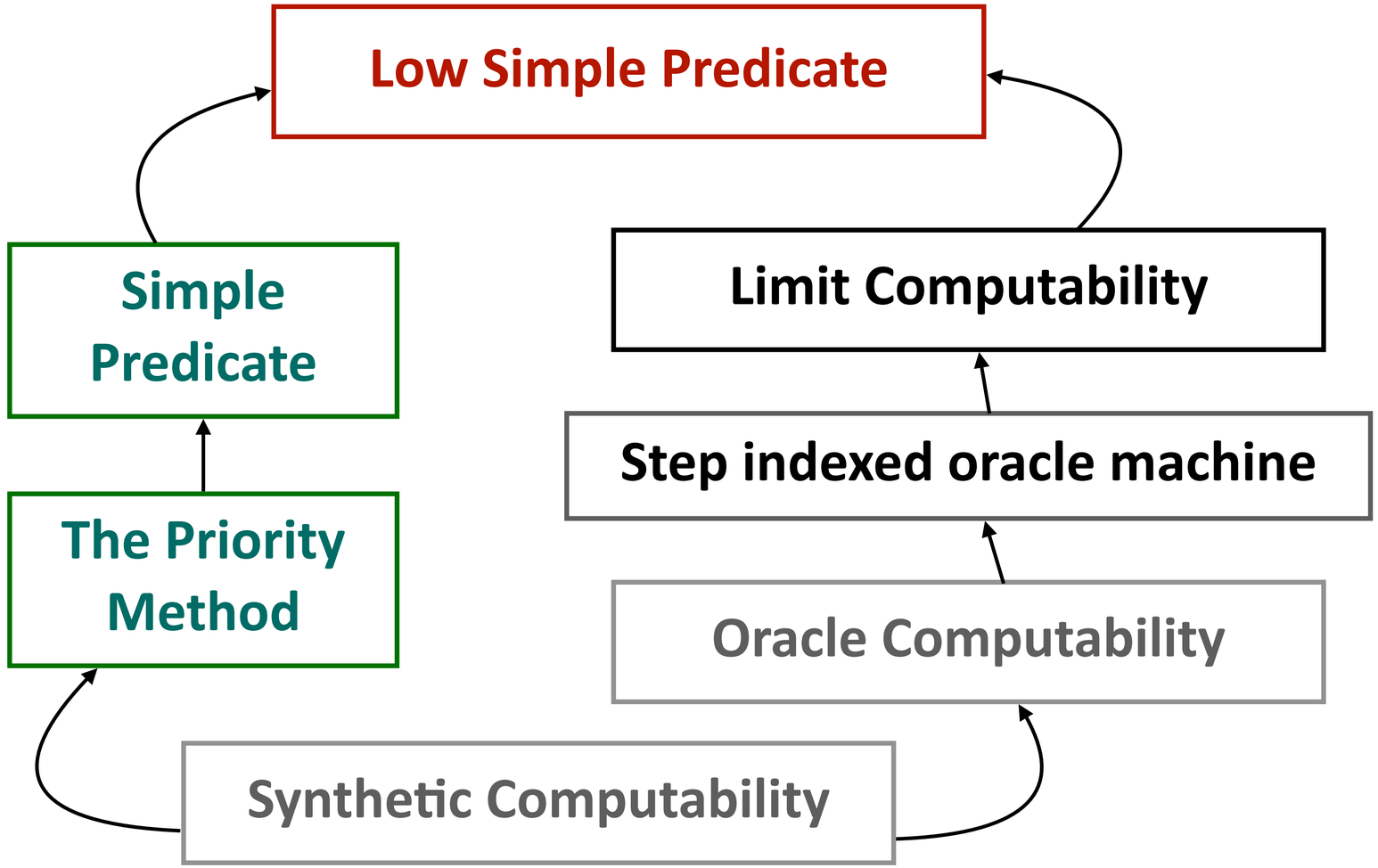
P_{γ_ω} is a positive solution to Post's Problem

Formalisation in Coq

Lines of code*: ~ 900

Lines of code: ~ 1000

Lines of code: ~ 250

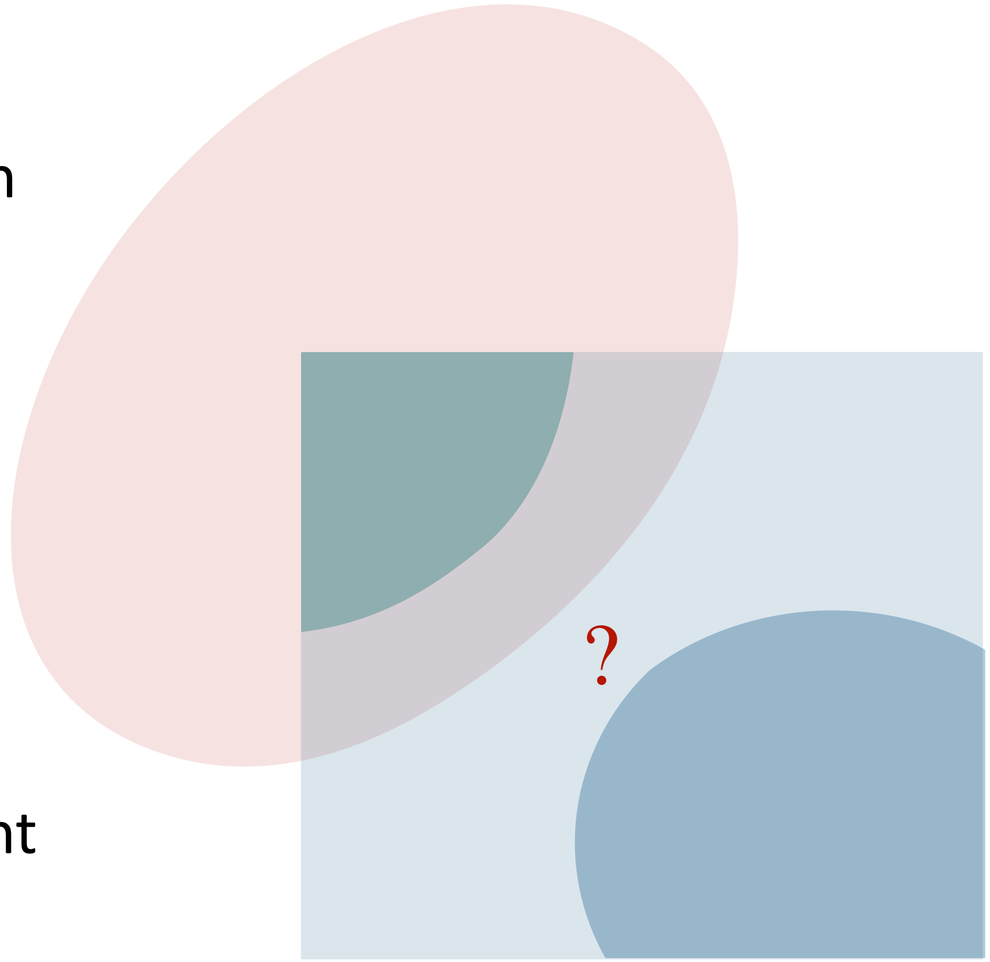


$*$: a technical lemma is missing

Contribution

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

- We give the first synthetic and mechanised solution to Post's problem, laying the foundation for a constructive analysis
- A synthetic notion of synthetic step-indexed oracle machines and use function
- A synthetic notion of limit computability and limit lemma
- All results are mechanised in the Coq proof assistant



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 $\tau : I \to A^* \rightharpoonup (Q + O)$:

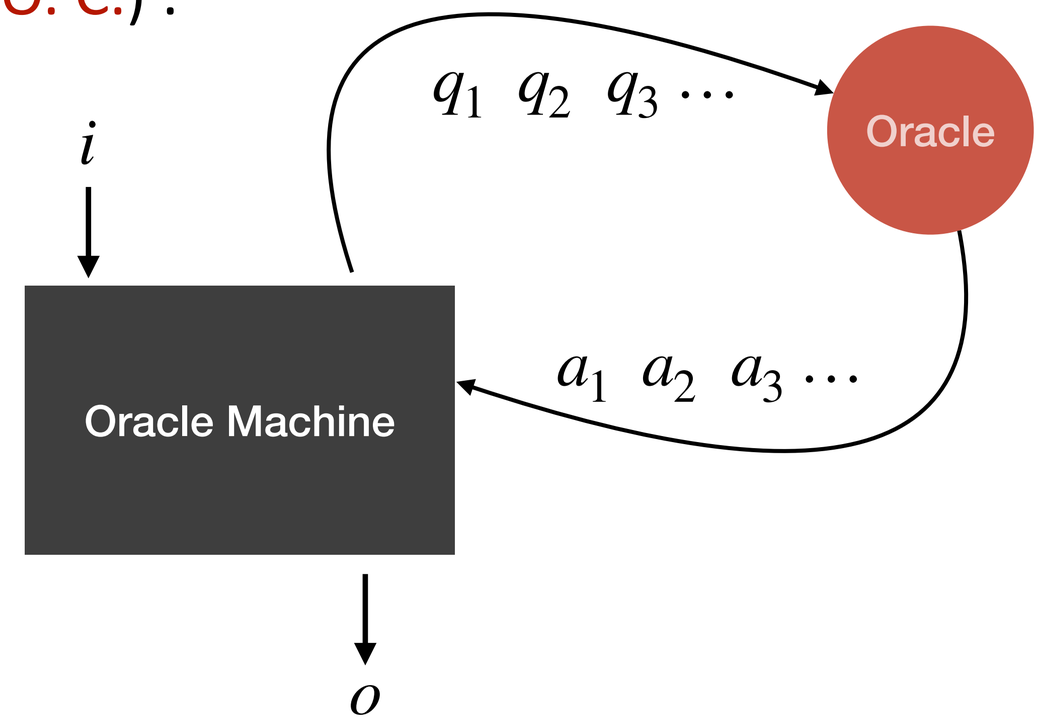
$$\forall R\ x\ b\,.\,F\ R\ x\ b \iff \exists qs\ as\,.\,\tau\ x; R \vdash qs; as \wedge \tau\ x\ as \rhd \text{out}\ b$$

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

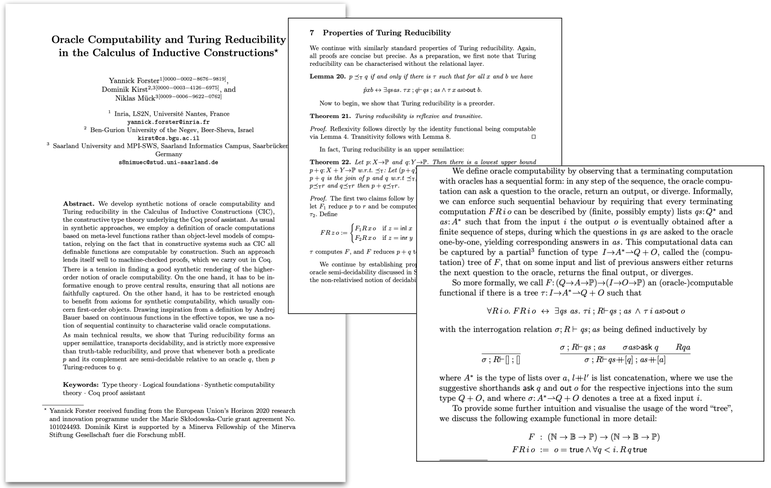
$$\frac{}{\sigma\ ; R \vdash [\,]; [\,]} \qquad \frac{\sigma\ ; R \vdash qs; as \quad \sigma\ as \rhd \text{ask}\ q \quad R(q, a)}{\sigma\ ; R \vdash qs \mathbin{@} [q]; as \mathbin{@} [a]}$$

Turing reducible in synthetic computability

Synthetic notation of Oracle Computable (O. C.) :



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



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

Yannick Forster[1][0000−0002−8676−9810], Dominik Kirst[2][0000−0003−4126−6975], and Niklas Mück[3][0009−0006−9622−0762]

[1] Inria, LS2N, Université Nantes, France
yannick.forster@inria.fr
[2] Ben-Gurion University of the Negev, Beer-Sheva, Israel
kirstdcs.bgu.ac.il
[3] Saarland University and MPI-SWS, Saarland Informatics Campus, Saarbrücken, Germany
s8nimuec@stud.uni-saarland.de

Abstract. We develop synthetic notions of oracle computability and Turing reducibility in the Calculus of Inductive Constructions (CIC), the constructive type theory underlying the Coq proof assistant. As usual in synthetic approaches, we employ a definition of oracle computations based on meta-level functions rather than object-level models of computation, relying on the fact that in constructive systems such as CIC all definable functions are computable by construction. Such an approach lends itself well to machine-checked proofs, which we carry out in Coq. There is a tension in finding a good synthetic rendering of the higher-order notion of oracle computability. On the one hand, it has to be informative enough to prove central results, ensuring that all notions are faithfully captured. On the other hand, it has to be restricted enough to benefit from axioms for synthetic computability, which usually concern first-order objects. Drawing inspiration from a definition by Andrej Bauer based on continuous functions in the effective topos, we use a notion of sequential continuity to characterise valid oracle computations. As main technical results, we show that Turing reducibility forms an upper semilattice, transports decidability, and is strictly more expressive than truth-table reducibility, and prove that whenever both a predicate p and its complement are semi-decidable relative to an oracle q, then p Turing-reduces to q.

Keywords: Type theory · Logical foundations · Synthetic computability theory · Coq proof assistant

* 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.

7 Properties of Turing Reducibility

We continue with similarly standard properties of Turing reducibility. Again, all proofs are concise but precise. As a preparation, we first note that Turing reducibility 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*

$$pxb \mapsto \exists qas.\ \tau x : q\text{-}qs : as \wedge \tau\, x\, as\text{-out}\, b.$$

Now to begin, we show that Turing reducibility is a preorder.

Theorem 21. *Turing reducibility is reflexive and transitive.*

Proof. Reflexivity follows directly by the identity functional being computable via Lemma 4. Transitivity follows with Lemma 8. □

In fact, Turing reducibility is an upper semilattice:

Theorem 22. *Let $p, q : X \to \mathbb{P}$ and $q, Y \to \mathbb{P}$. Then there is a lowest upper bound $p + q : X + Y \to \mathbb{P}$ w.r.t. $\preceq_\tau$: Let $(p+q)$*

$p + q$ is the join of p and q w.r.t $\preceq_\tau$
$p \preceq_\tau r$ and $q \preceq_\tau r$ then $p + q \preceq_\tau r$.

Proof. The first two claims follow by
let F_i reduce p to r and be computa
τ_2. Define

$$FRi\,o := \begin{cases} F_1 R\, x\, o & \text{if } z = \mathsf{inl}\ x \\ F_2 R\, x\, o & \text{if } z = \mathsf{inr}\ y \end{cases}$$

τ computes F, and F reduces $p + q$ t

We continue by establishing pro
oracle semi-decidability discussed in
the non-relativised notion of decidab

We define oracle computability by observing that a terminating computation with oracles has a sequential form: in any step of the sequence, the oracle computation can ask a question to the oracle, return an output, or diverge. Informally, we can enforce such sequential behaviour by requiring that every terminating computation $FRi\,o$ can be described by (finite, possibly empty) lists $qs : Q^*$ and $as : A^*$ such that from the input i the output o is eventually obtained after a finite sequence of steps, during which the questions in qs are asked to the oracle one-by-one, yielding corresponding answers in as. This computational data can be captured by a partial[2] function of type $I \to A^* \to Q + O$, called the (computation) tree of F, that on some input and list of previous answers either returns 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 $\tau : I \to A^* \to Q + O$ such that

$$\forall Ri\,o.\ FRi\,o \leftrightarrow \exists qs\ as.\ \tau i : R\text{-}qs : as \wedge \tau\, i\, as\text{-out}\, o$$

with the interrogation relation $\sigma : R \vdash qs : as$ being defined inductively by

$$\frac{}{\sigma : R \vdash [] : []} \qquad \frac{\sigma : R \vdash qs : as \qquad \sigma\, as\text{-ask}\, q \qquad Rqa}{\sigma : R \vdash qs +\!\!+ [q] : as +\!\!+ [a]}$$

where A^* is the type of lists over a, $l +\!\!+ l'$ is list concatenation, where we use the suggestive shorthands $\mathsf{ask}\ q$ and $\mathsf{out}\ o$ for the respective injections into the sum type $Q + O$, and where $\sigma : A^* \to Q + O$ 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:

$$F\ :\ (\mathbb{N} \to \mathbb{B} \to \mathbb{P}) \to (\mathbb{N} \to \mathbb{B} \to \mathbb{P})$$
$$FRi\,o\ :=\ o = \mathsf{true} \wedge \forall q < i.\, R\, q\, \mathsf{true}$$

Turing reduction $P \leq Q := \exists F\,.\ F$ is O.C. $\wedge\ \forall x\,.$ $P\ x \leftrightarrow F\ \hat{Q}\ x\ \mathrm{tt}$ $\neg P\ x \leftrightarrow F\ \hat{Q}\ x\ \mathrm{ff}$

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:

$$\Phi_\tau^{O(n)}\ x\ i\ j\ :=\ \begin{cases} \ulcorner \text{out } o \urcorner & \text{if } (\tau\ x\ []) \rightsquigarrow_j \text{out } o \\ \ulcorner \text{ask } q \urcorner & \text{if } (\tau\ x\ []) \rightsquigarrow_j \text{ask } q \text{ and } i = 0 \\ \Phi^{O(n)}_{\tau@[\chi_O\ n\ q]}\ x\ i'\ j & \text{if } (\tau\ x\ []) \rightsquigarrow_j \text{ask } q \text{ and } i = S\ i' \\ \text{none} & \text{otherwise} \end{cases}$$

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

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

Low Simple Predicate 1

undecidable predicate

Simple predicate:

If a predicate P is simple, then P is semi-decidable and $\neg(P \preceq \varnothing)$

Turing jump of P:

P' $x \leftrightarrow x$-th oracle machine with oracle P halts on x

Low Simple Predicate 2

Low predicate: A predicate P is low, if the Turing jump of P is reducible to K

$$P' \leq K \Rightarrow \neg(K \leq P)$$

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

A positive solution to Post's Problem

Showing a predicate is reducible to K is difficult!

Use Function

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

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

$$\varphi_e^p(x)[n] = k \to p[n] \equiv_k p[n+1] \to \varphi_e^p(x)[n+1] = k$$