Weihrauch Problems as Containers

(arXiv:2501.17250)

Cécilia Pradic

Ian Price
countingishard.org

TYPES
June 13th, 2025

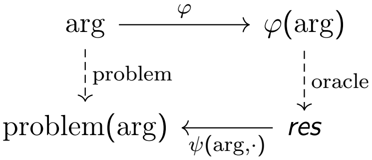
Reducibility: A refresher

- A problem A is reducible to a problem B if you can solve A given access to an oracle for B
- What "problem" and "solve" mean changes the theory.
- Undergraduates: problem = decision procedure, solve = give a (polytime) TM
- This is a useful framework for studying the structure of non-computable logical principles

Weihrauch Reducibility: Big Picture

What if you could only make a single oracle call?

```
def problem(arg):
  x = phi(arg)
  res = oracle(x)
  ans = psi(arg, res)
  return ans
```



Weihrauch Reducibility: Formally

Definition (Problem)

A Weihrauch problem is a family $(F_i)_{i \in I}$, where $I \subseteq \mathbb{N}^{\mathbb{N}}$ and $\emptyset \neq F_i \subseteq \mathbb{N}^{\mathbb{N}}$ for all $i \in I$.

Definition (Reducible)

Given problems $f = (F_i)_{i \in I}$ and $g = (G_j)_{j \in J}$, f is *Weihrauch reducible to* g if there exists partial type 2 computable maps

- $\varphi : I \to J$
- $\forall i \in I$, $\psi(i, \cdot)$ is a map $G_{\varphi(i)} \to F_i$

f is *strongly reducible* to g if ψ "ignores" i, i.e., $\psi(i, x) = \psi'(x)$ for some $\psi : \cup_{i \in I} G_{\phi(i)} \to \cup_{i \in I} F_i$.

Example: LPO and KL

- LPO: Decide if $w \in \{0,1\}^{\mathbb{N}}$ is constantly 0
- KL: Find an infinite path in an infinite binary tree *given by enumeration*
- Q: Is LPO reducible to KL, or vice versa? Equivalent? Incomparable?

Example: LPO and KL

- LPO: Decide if $w \in \{0,1\}^{\mathbb{N}}$ is constantly 0
- KL: Find an infinite path in an infinite binary tree *given by enumeration*
- Q: Is LPO reducible to KL, or vice versa? Equivalent? Incomparable?
- A: LPO is (strongly) reducible to KL
- A: KL is not reducible to LPO (argue by continuity)

LPO $\leq_{sW}$ KL

Algorithm φ

Require: $A = (a_n \in \{0, 1\})_{n \geq 1}$
Ensure: t is a binary tree with an infinite path

$t \leftarrow \emptyset$
for $a_n \in A, a_n = 0$ **do**
  add 0^n to t
for $m \in \mathbb{N}$ **do**
  add 1^m to t

$$\psi(a_n, p_n) = \begin{cases} \mathit{true}, \text{if } p_1 = 1 \\ \mathit{false}, \text{if } p_1 = 0 \end{cases}$$

LPO $\leq_{sW}$ KL

Algorithm φ

Require: $A = (a_n \in \{0,1\})_{n \geq 1}$
Ensure: t is a binary tree with an infinite path

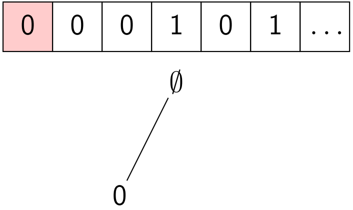
$t \leftarrow \emptyset$
for $a_n \in A, a_n = 0$ **do**
 add 0^n to t
for $m \in \mathbb{N}$ **do**
 add 1^m to t

$$\psi(a_n, p_n) = \begin{cases} \mathit{true}, \text{if } p_1 = 1 \\ \mathit{false}, \text{if } p_1 = 0 \end{cases}$$

0	0	0	1	0	1	...

$\emptyset$

LPO $\leq_{sW}$ KL



Algorithm φ

Require: $A = (a_n \in \{0, 1\})_{n \geq 1}$
Ensure: t is a binary tree with an infinite path

$t \leftarrow \emptyset$
for $a_n \in A, a_n = 0$ **do**
  add 0^n to t
for $m \in \mathbb{N}$ **do**
  add 1^m to t

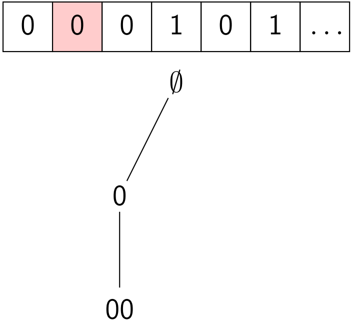
$$\psi(a_n, p_n) = \begin{cases} \mathit{true}, \text{if } p_1 = 1 \\ \mathit{false}, \text{if } p_1 = 0 \end{cases}$$

LPO $\leq_{sW}$ KL

Algorithm φ

Require: $A = (a_n \in \{0,1\})_{n\geq 1}$
Ensure: t is a binary tree with an infinite path
  $t \leftarrow \emptyset$
  for $a_n \in A, a_n = 0$ **do**
    add 0^n to t
  for $m \in \mathbb{N}$ **do**
    add 1^m to t

$$\psi(a_n, p_n) = \begin{cases} \mathit{true}, \text{if } p_1 = 1 \\ \mathit{false}, \text{if } p_1 = 0 \end{cases}$$

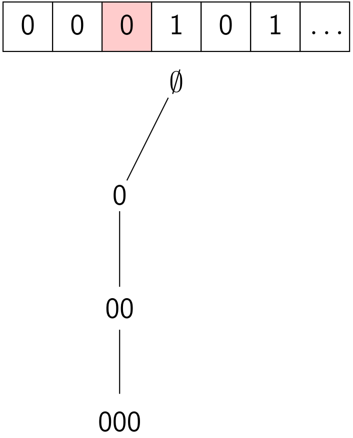


LPO $\leq_{sW}$ KL

Algorithm φ

Require: $A = (a_n \in \{0,1\})_{n\geq 1}$
Ensure: t is a binary tree with an infinite path
  $t \leftarrow \emptyset$
  for $a_n \in A, a_n = 0$ **do**
    add 0^n to t
  for $m \in \mathbb{N}$ **do**
    add 1^m to t

$$\psi(a_n, p_n) = \begin{cases} \mathit{true}, \text{if } p_1 = 1 \\ \mathit{false}, \text{if } p_1 = 0 \end{cases}$$



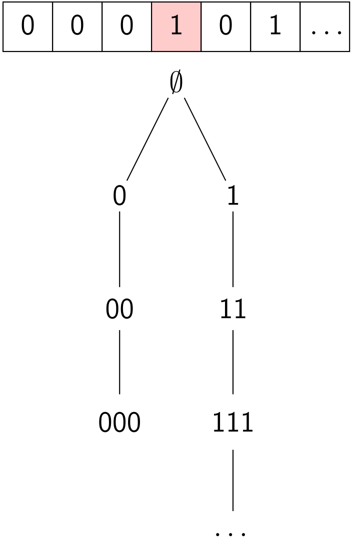
LPO $\leq_{sW}$ KL

Algorithm φ

Require: $A = (a_n \in \{0,1\})_{n \geq 1}$
Ensure: t is a binary tree with an infinite path

$t \leftarrow \emptyset$
for $a_n \in A, a_n = 0$ **do**
  add 0^n to t
for $m \in \mathbb{N}$ **do**
  add 1^m to t

$$\psi(a_n, p_n) = \begin{cases} \mathit{true}, \text{if } p_1 = 1 \\ \mathit{false}, \text{if } p_1 = 0 \end{cases}$$



Structure of Degrees

The Weihrauch ordering is pretty complicated [Brattka et al., 2021].

- There exist infinite chains and anti-chains
- No non-trivial suprema exist, but some non-trivial infima do
- ...

Weihrauch degrees (equivalence classes of $\leq_W$) have lots of structure

- Forms a lattice
  - $p \wedge q$: Ask two questions p & q, get the answer to one (chosen by oracle)
  - $p \vee q$: Ask either p or q, get the corresponding answer
- Parallel Product: Ask two questions at the same time, get both answers
- Composition: Ask a question, then dependent on the answer, ask another question and get its answer.
- ...

Generalising WR

The definition of WR given doesn't fundamentally depend on the type of computation.

Definition (Problem)

A Weihrauch problem is a family $(F_i)_{i \in I}$, where $I \subseteq \mathbb{N}^{\mathbb{N}}$ and $\emptyset \neq F_i \subseteq \mathbb{N}^{\mathbb{N}}$ for all $i \in I$.

Definition (Reducible)

Given problems $f = (F_i)_{i \in I}$ and $g = (G_j)_{j \in J}$, f is *Weihrauch reducible to* g if there exists partial type 2 computable maps

- $\varphi : I \to J$
- $\forall i \in I$, $\psi(i, \cdot)$ is a map $G_{\varphi(i)} \to F_i$

What do we need to generalise it to other categories?

Families and Bundles

Q: What is the category-theoretic equivalent of a family of sets indexed by a set I?

A: It's maps into I!

$$\begin{array}{rcl} \mathsf{Sets}/I & \simeq & \mathsf{Sets}^I \\ f : X \to I & \mapsto & \left(f^{-1}(i)\right)_{i \in I} \\ \pi : \coprod_{i \in I} X_i \to I & \leftarrow\!\shortmid & (X_i)_{i \in I} \end{array}$$

Reindexing families of sets becomes pullbacks of bundles.

$$\begin{array}{ccc} \coprod_{i \in I} G_{\varphi(i)} & \longrightarrow & \coprod_{j \in J} G_j \\ \big\downarrow{\scriptstyle \pi} & \lrcorner & \big\downarrow{\scriptstyle \pi} \\ I & \xrightarrow{\quad \varphi \quad} & J \end{array}$$

Generalising WR via Bundles

Definition (Problem)

A Weihrauch problem in a category $\mathcal{C}$ is a map $X \to I$.

Definition (Reduction)

Given two problems $f : F \to I$ and $g : G \to J$, a *reduction from* $f \to g$ is a pair of maps (φ, ψ)

- $\varphi : I \to J$ in the category $\mathcal{C}$
- $\psi : G \times_I J \to F$ in the slice $\mathcal{C}/I$

This is (essentially) the definition of container $(I \triangleright X)$ & container morphisms

History of Containers

Containers showed up in a lot of different places

- "Bidirectional Transformations" inspired by DB views [Foster et al., 2007]
- Functional Programming as "functional references" / "lenses" [van Laarhoven, 2007] [Kmett and contributors, 2012]
- Theory of Datatypes as "Containers" [Abbott et al., 2003]
- Category Theory as "Polynomials" [Gambino and Kock, 2012]
- Topological Complexity [Hirsch, 1990]

See this blog post by Jules Hedges for more history.

Are all containers Weihrauch problems?

Weihrauch problems were defined in terms of families of *non-empty* sets.

What is the corresponding condition on containers?

Definition (Answerable Containers)

We call a container *answerable* if the underlying map is a pullback-stable epimorphism.

Essentially, the projection maps from bundles must be surjective, i.e., all questions have answers.

Theorem

The Weihrauch degrees are isomorphic to the posetal reflection of the category of answerable containers over the category $\mathrm{pMod}(\mathcal{K}_2^{\mathrm{rec}}, \mathcal{K}_2)$.

Structure of Containers

Containers also have a lot of structure

- Forms a (Bi)category
- Inherits limits / colimits from base category
- Has a composition product
- Has a monoidal product
- Fixed points
- Derivatives (zippers)
- ...

How does this structure line up with Weihrauch Reducibility?

Where we're at

Containers	Reducibility	Status
Answerable Containers over pMod	Weihrauch Degrees	✓
Containers over pAsm	Extended Degrees	✓
Dependent Adaptors	Strong Degrees	✓
Product $p \times q$	Meet $p \wedge q$	✓
Coproduct $p + q$	Join $p \vee q$	✓
Tensor Product $p \otimes q$	Parallel Product $p \times q$	✓
Composition Product $p \circ q$	Composition $p \circ q$	✓*
Free monad on p	Iterated Composition $p^\diamond$	
Derivative ∂p	?	✗
?	First-order part ${}^1p$	
...	...	

References I

Abbott, M. G., Altenkirch, T., and Ghani, N. (2003).
Categories of containers.
In Gordon, A. D., editor, *FOSSACS 2003 proceedings*, volume 2620 of *Lecture Notes in Computer Science*, pages 23–38. Springer.

Brattka, V., Gherardi, G., and Pauly, A. (2021).
Weihrauch Complexity in Computable Analysis, pages 367–417.
Springer International Publishing, Cham.

Foster, J. N., Greenwald, M. B., Moore, J. T., Pierce, B. C., and Schmitt, A. (2007).
Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem.
ACM Transactions on Programming Languages and Systems, 29(3):17–es.

References II

Gambino, N. and Kock, J. (2012).
Polynomial functors and polynomial monads.
Mathematical Proceedings of the Cambridge Philosophical Society, 154(1):153–192.

Hirsch, M. D. (1990).
Applications of topology to lower bound estimates in computer science.
PhD thesis, University of California, Berkeley.

Kmett, E. and contributors (2012).
lens: Lenses, folds and traversals.

van Laarhoven, T. (2007).
Overloading functional references.

Thank You!

Any Questions?

i.j.price@swansea.ac.uk
countingishard.org

KL $\not\leq$ LPO

- Suppose KL $\leq$ LPO and have a weihrauch reduction (φ, ψ).
- $\varphi(t) = 000\ldots$ for some infinite tree t.
    - Otherwise, $\psi(\cdot, \mathit{false})$ implies KL is computable.
- Set $(p_n)_{n\in\mathbb{N}} = \psi(t, \mathit{true})$.
- p_0 will have been output after reading a finite part of t, say t_1.
- $\varphi(t)$ will output 0 after reading a finite part of t, say t_2.
- Any infinite tree that agrees with t on $t_1 \cup t_2$ will output the same p_0.
- So pick one whose only infinite path doesn't start with p_0. $\lightning$

Strong Weihrauch Reducibility

- Q: How does Strong reducibility fit into this framework
- A: "Dependent adapters"
- This is recent (unpublished) work in the Containers community [Hedges et al.]
- Key idea: Relations have two projections, not just one.
- Fact: Containers come from the opposite of the codomain fibration.

$$\begin{aligned} \mathrm{cod} : \mathcal{C}^{\rightarrow} &\rightarrow \mathcal{C} \\ (f : X \rightarrow Y) &\mapsto Y \end{aligned}$$

- Fact: Adapters are the opposite of a different fibration.

$$\begin{aligned} F : \mathsf{RelSpan}(\mathcal{C}) &\rightarrow \mathcal{C} \\ (X \leftarrow Y \twoheadrightarrow Z) &\mapsto X \end{aligned}$$

Fixed Points

Theorem

If F is a fibred polynomial endofunctor over containers and $\mathcal{C}$ has dependent M-types and W-types, the following exist:

- *an initial algebra μF for F*
- *a terminal coalgebra νF for F*
- *a (co)algebra ζF for F*

Examples:

- $P^{\circ} = \mu(X \mapsto \mathsf{I} + X \circ P)$, the free monad on P
- $P^{\otimes} = \mu(X \mapsto \mathsf{I} + X \otimes P)$
- $P^{\otimes\infty} = \zeta(X \mapsto X \otimes P)$
- $P^{\circ\infty} = \zeta(X \mapsto X \circ P)$