

# Quantifying Degrees of Relative Solvability: When Does One Problem Reduce to Another?

Daniel Mourad

University of Connecticut

*Daniel.Mourad@Uconn.edu*

April 22, 2022

# Introduction

# Corny Joke

A mathematician finds that a fire has broken out in their office. They ask them-self,

“How can I put out this fire?”

Then, they remember that they have a fire blanket locked in their desk drawer.

“Putting out the fire reduces to getting the blanket from the drawer!”

## Question

*What does it mean for problem  $Q$  to be reducible to problem  $P$ ?*

## Answer (Naive I)

*$Q$  is reducible to  $P$  if it is easy to prove  $Q$  using  $P$ .*

*“Suppose that I could get the fire blanket out of the drawer. Then I could put out the fire!”*

## Answer (Naive II)

*$Q$  is reducible to  $P$  if we know how to use a solution to  $P$  to get a solution to  $Q$ .*

*“If I had a way to get the fire blanket out of the desk, then I could use it to put out the fire!”*

These answers seem similar because they are almost equivalent for statements whose proof consists of finding a witness.

The mathematician has successfully picks the lock and uses the fire blanket to put out the fire. They leave the fire blanket on the floor and go home.

The next day, another fire breaks out. Once again, they ask them-self

“How can I put out this fire?”

# Punchline!

They pick up the fire blanket and lock it back into their desk.

# Punchline!

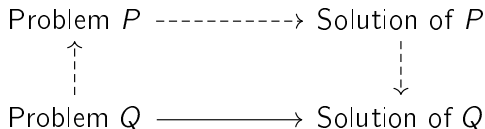
They pick up the fire blanket and lock it back into their desk.

“I have reduced the problem into one I have already solved!”

Satisfied that they have solved the problem, they go home.

## Answer (Naive III)

*Q is reducible to P if we can transform Q into P and then use the solution of P to get a solution of Q.*



# Problems

We will formalize these notions for a specific class of problems.

## Idea

*Instead of thinking about problems that are solved once and done with, we collect classes of similar problems into a set of **instances**. Each instance has its own set of **solutions**.*

Many theorems can be stated as the existence of a solution to each instance of such a problem.

## Theorem (Heine-Borel [HB])

*Each open cover of  $[0, 1] \subset \mathbb{R}$  has a finite subcover.*

Instances are open covers  $\{U_i\}_{i \in I}$  of  $[0, 1]$ .

Solutions of  $\{U_i\}_{i \in I}$  are  $\{j_k\}_{k \in \mathbb{N}}$  such that  $\{U_{j_k}\}_{k \in \mathbb{N}}$  covers  $[0, 1]$ .



# More Examples of Problems

## Theorem (CRhPI)

*Every commutative ring has a prime ideal*

Instances are commutative rings  $R$ .

Solutions of  $R$  are prime ideals of  $R$ .

## Theorem (Weak König's Lemma [WKL])

*Each infinite binary tree has an infinite path.*

Instances are infinite binary trees  $T$ .

Solutions of  $T$  are paths through  $T$ .

## Question

*What does it mean for problem  $Q$  to be reducible to problem  $P$ ?*

## Answer (Reverse Math)

*$Q$  is reducible to  $P$  if  $P$  can be used to prove  $Q$  over some weaker system of axioms.*

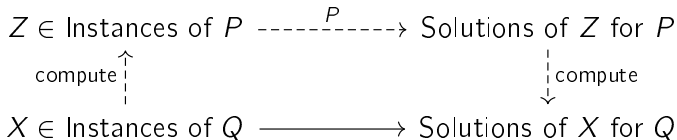
# Reducibility For Problems

## Question

What does it mean for problem  $Q$  to be reducible to problem  $P$ ?

## Answer (Computable Combinatorics)

$Q$  is reducible to  $P$  if, from each instance  $X$  of  $Q$ , we can **compute** an instance  $Z$  of  $P$  such that we can use solutions of  $Z$  to **compute** solutions of  $X$ .



Both answers, along with the connections between them, yield rich mathematics. For today, we will focus on the computable combinatorics answer.

# Computability

# Computability (Informal)

## Definition

We say that a function  $f : \mathbb{N} \rightarrow \mathbb{N}$  is computable if there is an algorithm that takes  $x$  as an input and returns  $f(x)$  as an output.

## Definition

We say that  $X \subset \mathbb{N}$  is computable if there is an algorithm that computes its characteristic function

$$\mathbb{1}_X(n) = \begin{cases} 1 & \text{if } n \in X \\ 0 & \text{if } n \notin X \end{cases}$$

# Computability (Informal)

Consider the bijection  $\langle \cdot, \cdot \rangle : \mathbb{N}^2 \rightarrow \mathbb{N}$

$$\langle x, y \rangle = \frac{(x + y)(x + y + 1)}{2} + y.$$

We can code arbitrary tuples  $(x_1, x_2, \dots, x_n)$  by the bijective function  $\ulcorner \cdot \urcorner : \{\text{ordered tuples}\} \rightarrow \mathbb{N}$

$$\ulcorner (x_1, x_2, \dots, x_n) \urcorner = \langle n, \langle x_1, \langle x_2, \langle x_3, \dots, \langle x_{n-1}, \langle x_{n-1}, x_n \rangle \rangle \dots \rangle \rangle \rangle \rangle.$$

## Example

Consider the projection function  $f(\ulcorner (x_1, x_2, x_3, \dots, x_n) \urcorner) = x_1$ . Then,  $f$  is computable by the following algorithm: first, find  $a$  and  $b$  such that  $\langle a, b \rangle = \ulcorner (x_1, x_2, x_3, \dots, x_n) \urcorner$ . Then, find  $a'$  and  $b'$  such that  $\langle a', b' \rangle = b$ . By definition of  $\ulcorner (x_1, x_2, \dots, x_n) \urcorner$  we have that  $a' = x_1$ .

# Partial Functions

Depending on starting parameters, some algorithms enter infinite loops and never produce an output. We model this behavior using partial functions.

- A partial function  $f : \mathbb{N} \rightarrow \mathbb{N}$  is a classical function  $f : \mathbb{N} \rightarrow \mathbb{N} \cup \{\uparrow\}$ .
- If  $f(x) = \uparrow$ , we say that  $f(x)$  diverges and write  $f(x) \uparrow$ .
- If  $f(x) = n \in \mathbb{N}$ , we say that  $f(x)$  converges and write  $f(x) \downarrow$  as well as  $f(x) \downarrow = n$ .
- A partial function  $f : \mathbb{N} \rightarrow \mathbb{N}$  is total if  $f(x) \downarrow$  for all  $x \in \mathbb{N}$ .

## Example

Let  $f(n) = \lceil (p_1, p_2) \rceil$  where  $p_1$  and  $p_2$  are the  $n$ 'th pair of twin primes. The twin prime conjecture can then be rephrased as “ $f$  is a total function”.

# Computability (Formal)

The set of partial computable functions from  $\mathbb{N}$  to  $\mathbb{N}$  is the set of partial functions built up from

- the constant functions  $C_n(x) = n$  for all  $x$
- the successor function  $S(x) = x + 1$

and the following operations given partial computable functions  $g, h$ :

- Composition:  $f(x) = g(h(x))$
- Primitive recursion: A for-loop for  $f$  whose initial value is determined by  $g$  and whose iterative step is determined by  $h$ .
- $\mu$ -recursion:  $f(x) = \mu x.(g(x) = 0)$  is a while loop that returns the least  $x$  such that  $g(x) = 0$  and  $g(y) \downarrow \neq 0$  for each  $y < x$ .

## Definition

The set of **computable** functions is the subset of partial computable functions that are also total.  $X \subset \mathbb{N}$  is computable if  $\mathbb{1}_X$  is computable.



# Computability Examples

## Example

$f(\ulcorner x, y \urcorner) = x^y$  is computable.

## Theorem (Church-Turing Thesis)

*Any function you could code on a computer is partial computable*

By encoding the recursive definition of each partial computable function into a number, we can enumerate them by  $\varphi_1, \varphi_2, \varphi_3, \dots$

# Primitive Recursive Functions

Partial computable functions that have no  $\mu$ -recursion in their definition are always total.

## Definition

The **primitive recursive** functions is the subset of the partial computable functions that can be defined without  $\mu$ -recursion.

- By encoding the recursive definition of each primitive recursive function into a number, we can enumerate them by  $f_1, f_2, f_3, \dots$
- This gives us an computable list of algorithms that we know are total.

# A Set That is Not Computable

- By encoding the recursive definition of each primitive recursive function into a number, we can enumerate them by  $f_1, f_2, f_3, \dots$
- Then,

$$K = \{e : \mu x. (f_e(x) = 0) \downarrow\} = \{e : 0 \in \text{range}(f_e)\}$$

is not computable. There is no algorithmic way to confirm that there does not exist an  $x$  such that  $f_e(x) = 0$ .

## Example

The function

$$f(x) = \begin{cases} 1 & \text{if } 2x = p_1 + p_2 \text{ for some prime } p_1, p_2 < x \\ 0 & \text{otherwise} \end{cases}$$

is primitive recursive, but knowing whether or not  $\mu x. (f(x) = 0)$  converges is very difficult.

# Relative Computability

To state our notion of reducibility, we need a notion of what it means to use a solution of an instance of  $P$  to compute another.

## Definition

Fix a set  $X \subset \mathbb{N}$ . The partial computable functions relative to  $X$  is the set of partial functions built up from

- The successor and constant functions
- $\mathbb{1}_X$

and closed under

- Composition
- Primitive recursion (for loops)
- $\mu$ -recursion (while loops)

We say that  $Y \subset \mathbb{N}$  is computable from  $X \subset \mathbb{N}$  if the characteristic function of  $Y$  is  $X$ -total computable. We write  $Y \leq_T X$ . Note that we can code sets  $\{X_i\}_{i \in \mathbb{N}}$  into one set  $\bigoplus X_i = \{\ulcorner (i, x) \urcorner : x \in X_i\}$ .

# Turing Functionals

Consider the recursive definition of a partial  $X$ -computable function  $f$ . By replacing all instances of  $\mathbb{1}_X$  with  $\mathbb{1}_Y$ , we can use the same recursive definition to define partial  $Y$ -computable function  $g$ . By encoding the recursion, this gives us a list of functionals

$$\Phi_1, \Phi_2, \Phi_3, \dots$$

where, for each  $Y \subset \mathbb{N}$ ,  $\Phi_e^Y$  is the  $e$ 'th partial  $Y$ -computable function.

## Example

Let  $\Phi^X(n)$  be the sum of the  $n$  least non-zero elements of  $X$ . Then,

- $\Phi^{2\mathbb{N}}(2) = 2 + 4 = 6$
- $\Phi^{3\mathbb{N}}(2) = 3 + 6 = 9$
- $\Phi^{\{1,5,7\}}(2) = 6$  and
- $\Phi^{\{1,5,7\}}(17) \uparrow$ .

# Computable Reducibility and Weihrauch Reducibility

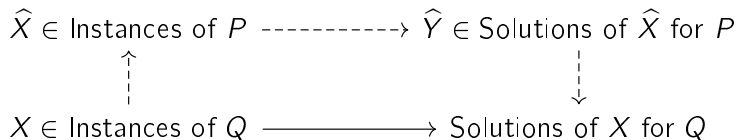
# Computable Reducibility

## Answer (Computable Combinatorics)

$Q$  is reducible to  $P$  if, from each instance  $X$  of  $Q$ , we can **compute** an instance  $Z$  of  $P$  such that we can use solutions of  $Z$  to **compute** solutions of  $X$ .

## Definition

$Q$  is **computably reducible** to  $P$  if, for each instance  $X$  of  $Q$ , there is an  $X$ -computable instance  $\hat{X}$  of  $P$  such that for any solution  $\hat{Y}$  of  $\hat{X}$ ,  $X \oplus \hat{Y}$  computes a solution to  $X$ . We write  $Q \leq_c P$ .



# Computable Reducibility

## Definition

$Q$  is **computably reducible** to  $P$  if, for each instance  $X$  of  $Q$ , there is an  $X$ -computable instance  $\hat{X}$  of  $P$  such that for any solution  $\hat{Y}$  of  $\hat{X}$ ,  $X \oplus \hat{Y}$  computes a solution to  $X$ . We write  $Q \leq_c P$ .

## Definition

$Q$  is **strongly computably reducible** to  $P$  if, for each instance  $X$  of  $Q$ , there is an  $X$ -computable instance  $\hat{X}$  of  $P$  such that any solution  $\hat{Y}$  of  $\hat{X}$  computes a solution to  $X$ . We write  $Q \leq_{sc} P$ .

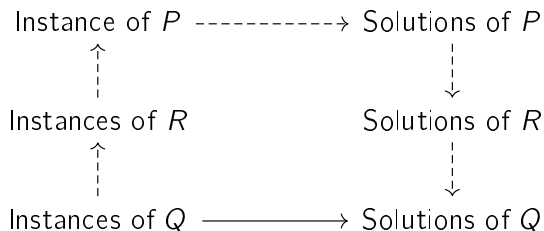


# Transitivity

## Theorem

*Both forms of computable reductions are transitive: If  $Q \leq_c R$  and  $R \leq_c P$  then  $Q \leq_c P$ .*

Proof: In the following diagram:



# A Strong Computable Reduction

First, we define  $\text{id}$ , the identity problem. Every  $X \subset \mathbb{N}$  is an instance of  $\text{id}$  and the solution of  $X$  is  $X$  itself.

- $\text{id}$  is computably reducible (but not strongly computably reducible) to every problem with a computable instance.
- If  $P \leq_{sc} \text{id}$  then every instance  $X$  of  $P$  has an  $X$ -computable solution. In this case, we say that  $P$  is *computable*.

## Theorem

$\text{HB} \leq_{sc} \text{id}$ : *The Heine-Borel theorem is strongly computably reducible to the identity problem. In other words, from an open cover  $\{U_n\}_{n \in \mathbb{N}}$  of  $[0, 1]$ , we can compute  $k$  such that  $[0, 1] \subset \bigcup_{n=1}^k U_n$ .*

Proof: Finite sets are computable, so  $\{k\}$  is computable for every  $k$ .

# A Uniform Strong Computable Reduction

## Theorem

$\text{HB} \leq_{sc} \text{id}$ : *The Heine-Borel theorem is strongly computably reducible to the identity problem. In other words, from an open cover  $\{U_n\}_{n \in \mathbb{N}}$  of  $[0, 1]$ , we can compute  $k$  such that  $[0, 1] \subset \bigcup_{n=1}^k U_n$ .*

Uniform Proof:

- Enumerate the basic open sets as  $(a_1, b_1), (a_2, b_2), \dots$  where  $a_i, b_i \in \mathbb{Q}$ .
- We code open sets  $U$  as the set of indices  $\underline{U} = \{i : (a_i, b_i) \subset U\}$
- We code an open cover  $\mathcal{U} = \{U_n\}_{n \in \mathbb{N}}$  by  $\underline{\mathcal{U}} = \bigoplus \underline{U}_i$ .

# Uniform proof (Cont)

Let

$$\mathcal{U}_s = \left\{ i < s : (a_i, b_i) \subset \bigcup_{j=1}^s U_j \right\}.$$

Let  $k$  be the least  $s$

$$[0, 1] \subset \bigcup_{i \in \mathcal{U}_s} (a_i, b_i),$$

The existence of  $k$  is guaranteed because the Heine-Borel theorem is true.

To algorithmically find  $k$  using oracle  $\underline{U} = \{(n, i) : (a_i, b_i) \subset U_n\}$ , compute the finite sets  $\mathcal{U}_s$  and check if they cover the unit interval until you find one that does.

# Pigeonhole Principle

## Theorem (Infinite Pigeonhole Principle [ $RT_k^1$ ])

*For each  $k$ -coloring  $c : \mathbb{N} \rightarrow \{0, 1, \dots, k - 1\}$  there is an infinite  $X \subset \mathbb{N}$  such that  $c$  restricted to  $X$  is constant.*

- This is a special case of infinite Ramsey's Theorem, hence the notation  $RT_k^1$ .
- Note that we are splitting up the problem for different values of  $k$ .

## Theorem

$RT_2^1 \leq_{sc} id$ : *The infinite pigeonhole principle is computable.*

Proof:

- Let  $c : \mathbb{N} \rightarrow \{0, 1\}$  be a 2-coloring of the natural numbers.
- Then, the characteristic functions of  $X = \{x : c(x) = 0\}$  and  $Y = \{x : c(x) = 1\}$  are both computable when using  $c$  as an oracle.
- Because  $RT_2^1$  is true, we have that either  $X$  is a solution of  $c$  or  $Y$  is a solution of  $c$ .
- Since both  $X$  and  $Y$  are computable from  $c$ ,  $c$  has a  $c$ -computable solution.

This proof is non uniform. However, there is no uniform version.

## Theorem

*There is no Turing Functional  $\Phi$  such that, for each  $c : \mathbb{N} \rightarrow \{0, 1\}$ ,  $\Phi^c(x)$  is total and the characteristic function of an infinite  $X \subset \mathbb{N}$  on which  $c$  is constant*

Suppose there were such a  $\Phi$ . Enumerate the primitive recursive functions by  $f_1, f_2, \dots$ . The existence of  $\Phi$  implies that we can compute  $\mu x.(f_e(x) = 0)$  for each  $e$ , which is impossible. To see this,

- Define total computable  $c_e$  by

$$c_e(x) = \begin{cases} 0 & \text{if there exists } y \leq x \text{ such that } f_e(y) = 0 \\ 1 & \text{otherwise} \end{cases}$$

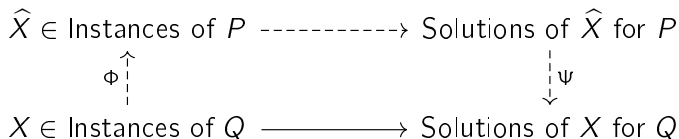
- Emulate  $\Phi^{c_e}(x)$  by performing all the same steps, except that when  $\Phi^{c_e}$  asks whether  $c_e(x) = i$ , compute  $c_e(x)$  to find the result. Call this algorithm  $\varphi(e, x)$ .
- To determine if  $\mu x.(f_e(x) = 0)$  halts, find  $x$  such that  $\varphi(e, x) = 0$ , then compute  $c_e(x)$ .

# Weihrauch Reducibility

We formalize this notion of uniformly computable reducible.

## Definition

$Q$  is **Weihrauch reducible** to  $P$  if there exist oracle-algorithms  $\Phi$  and  $\Psi$  such that for each instance  $X$  of  $Q$ ,  $\Phi^X$  is an instance of  $P$  such that for any solution  $\hat{Y}$  of  $\Phi^X$ ,  $\Psi^{\hat{Y} \oplus X}$  is a solution of  $Q$ . We write  $Q \leq_W P$ .



We write  $Q \leq_{sW} P$  if such a  $\Phi$  and  $\Psi$  exist where  $\Psi$  is not allowed to use  $X$ .



# Examples

## Theorem

$RT_2^1 \not\leq_W \text{id}$ .

## Theorem

$HB \leq_{sW} \text{id}$ .

## Theorem

$RT_k^1 \leq_{sW} RT_{k'}^1$  for  $k \leq k'$ . *The reverse direction does not hold.*

## Theorem (Simpson, 2009)

$CRhPI \leq_{sW} WKL$

# Proof of $\text{CRhPI} \leq_{sW} \text{WKL}$

## Definition

$I$  is a prime ideal of ring  $(R, +, \cdot)$  if

- $(I, +)$  is a proper subgroup of  $(R, +)$
- For each  $r \in R, x \in I, r \cdot x \in I$
- For all  $r, q \in R$ , if  $r \cdot q \in I$  then either  $r \in I$  or  $q \in I$ .

## Definition

$f : \mathbb{N} \rightarrow \{0, 1\}$  is a path through binary tree  $T \subset \{\text{finite binary strings}\}$  if  $f|_n \in T$  for each  $n \in \mathbb{N}$ .

Strategy: suppose we are given functions  $+_R : \mathbb{N}^2 \rightarrow \mathbb{N}$  and  $\cdot_R : \mathbb{N}^2 \rightarrow \mathbb{N}$  that define a commutative ring on  $R = \mathbb{N}$  with additive identity 0 and multiplicative identity 1. We have  $\Phi^{+_R \oplus \cdot_R}$  construct a tree  $T$  of finite binary strings such that  $f$  is a path through  $T$  if and only if  $f$  is the characteristic function of a prime ideal of  $R$ .

# Proof of $\text{CRhPI} \leq_{sW} \text{WKL}$ (cont.)

Let  $\sigma \in T$  if and only if  $|\sigma| \leq 2$  or, for all  $i, j, k < |\sigma|$ ,

- $\sigma(0) = 1$
- $\sigma(1) = 0$
- If  $\sigma(i) = \sigma(j) = 1$  and  $i +_R j = k$  then  $\sigma(k) = 1$
- If  $\sigma(i) = 1$  and  $i \cdot_R j = k$  then  $\sigma(k) = 1$
- If  $\sigma(i) = \sigma(j) = 0$  and  $i \cdot_R j = a_k$  then  $\sigma(k) = 0$ .

$T$  is a tree because these conditions are preserved by initial segment.  $T$  is infinite because  $R$  has a prime ideal. Thus, WKL gives us a path that is also an ideal ( $\Psi^{\hat{Y}} = \hat{Y}$ ).

# Operations on Problems

What if we want to allow multiple uses of a problem?

## Theorem (Informal)

$RT_4^1$  is reducible to two uses of  $RT_2^1$ .

Proof: Given  $c : \mathbb{N} \rightarrow \{0, 1, 2, 3\}$ , define  $c_1 : \mathbb{N} \rightarrow \{0, 1\}$  by

$$c_1(x) = \begin{cases} 0 & \text{if } c(x) = 0 \text{ or } c(x) = 1 \\ 1 & \text{if } c(x) = 2 \text{ or } c(x) = 3 \end{cases}.$$

Apply  $RT_2^1$  again to get an infinite  $X$  such that  $c_1$  is constant on  $X$ . Then,  $c$  is two-valued on  $X$ , either taking values in  $\{0, 1\}$  or  $\{2, 3\}$ . Let  $g$  be an increasing bijection from  $\mathbb{N}$  to  $X$ . Let

$$c_2(x) = c(g(x)) \pmod{2}.$$

Apply  $RT_2^1$  to  $c_2$  to get  $Y$  such that  $c$  is constant on  $Y$ . Then,  $g(Y)$  is computable from  $X$  and  $c$  is constant on  $g(Y)$ .

# Compositional Product

The previous theorem was a proof that  $RT_4^1 \leq_{sW} RT_2^1 \star RT_2^1$ , where  $\star$  denotes the compositional product. Below, we give a precise definition of  $\star$ .

## Definition

Suppose all solutions to every instance of  $P$  are also instances of  $Q$ . Then, we can define  $P \circ Q$ , whose instances are instances of  $P$ .  $(P \circ Q)$ -Solutions of  $X$  are all  $Q$ -solutions of any  $P$ -solution of  $X$ .

## Definition (Brattka, Gherardi, and Marcone, 2012)

$R \leq_W P \star Q$  if and only if there are  $P' \leq_W P$  and  $Q' \leq_W Q$  such that  $R \leq_W P' \circ Q'$ .

Note that  $P \star Q$  is not a problem, but a degree.

Write  $X \oplus Y$  as  $\langle X, Y \rangle$ .

We showed that  $RT_4^1 \leq_{sW} P \circ Q$  for  $P, Q \leq_{sW} RT_2^1$  defined by

- Instances of  $P$  are pairs  $\langle c, c_1 \rangle$  such that  $c : \mathbb{N} \rightarrow \{0, 1, 2, 3\}$  and  $c_1 : \mathbb{N} \rightarrow \{0, 1\}$ .
- $P$ -solutions of  $\langle c, c_1 \rangle$  are  $\langle c_2, X \rangle$  such that  $c_2(x) = c(x) \pmod 2$  and  $c_1$  is constant on  $X$ .
- Instances of  $Q$  are  $\langle c_2, X \rangle$  such that  $c_2 : \mathbb{N} \rightarrow \{0, 1\}$  and  $X$  is any subset of  $\mathbb{N}$ .
- $Q$ -solutions of  $c_2 \oplus X$  are  $\langle Y, X \rangle$  such that  $c_2$  is constant on  $Y$ .

$$\begin{array}{ccccc}
 \langle c, c_1 \rangle & \overset{P}{\dashrightarrow} & \langle c_2, X \rangle & \overset{Q}{\dashrightarrow} & \langle Y, X \rangle \\
 \uparrow \Phi^c & & & & \downarrow \Psi^{Y \oplus X} \\
 c & \xrightarrow{\quad\quad\quad} & & & g(Y)
 \end{array}$$

# Outlook



## Theorem (Brattka and Pauly, 2018)

$P \star Q$  is a well defined  $\equiv_W$ -degree. In other words, for each  $P$  and  $Q$ , there exists a problem  $R$  such that for each  $S$  and  $T$ ,  $S \leq_W R \leq_W T$  if and only if  $S \leq_W P \star Q \leq_W T$ .

There are many connections between reverse math and this account of reducibility. In the following example, let  $P^{\star n}$  be the the  $\star$ -product of  $n$  copies of  $P$ .

## Theorem (Dzhafarov, Hirschfeldt, and Reitzes, 2020)




Let  $\Gamma$  consist of  $\text{RCA}_0$  together with all  $\Pi_1^1$  formulas true over  $\mathbb{N}$  and suppose that  $P$  has computable instances. If  $Q \not\leq_W P^{\star n}$  for all  $n$ , then

$$\Gamma \not\vdash P \rightarrow Q$$

# Further Reading

- Vasco Brattka, Guido Gherardi, and Arno Pauly (2021). “Weihrauch Complexity in Computable Analysis”. In: *Handbook of Computability and Complexity in Analysis*. Ed. by Vasco Brattka and Peter Hertling. Cham: Springer International Publishing, pp. 367–417. ISBN: 978-3-030-59234-9. DOI: 10.1007/978-3-030-59234-9\_11. URL: [https://doi.org/10.1007/978-3-030-59234-9%7B%5C\\_%7D11](https://doi.org/10.1007/978-3-030-59234-9%7B%5C_%7D11)
- Damir D Dzhafarov, Jun Le Goh, Denis R Hirschfeldt, Ludovic Patey, and Arno Pauly (2020). “Ramsey’s theorem and products in the Weihrauch degrees”. In: *Computability 9*, pp. 85–110. ISSN: 2211-3576. DOI: 10.3233/COM-180203

Thank you!

-  Brattka, Vasco, Guido Gherardi, and Alberto Marcone (Jan. 2012). “The Bolzano-Weierstrass Theorem is the Jump of Weak  $K\backslash$ ’onig’s Lemma”. In: *Annals of Pure and Applied Logic* 163.6. DOI: [10.1016/j.apal.2011.10.006](https://doi.org/10.1016/j.apal.2011.10.006). arXiv: 1101.0792. URL: <http://arxiv.org/abs/1101.0792><http://dx.doi.org/10.1016/j.apal.2011.10.006>.
-  Brattka, Vasco, Guido Gherardi, and Arno Pauly (2021). “Weihrauch Complexity in Computable Analysis”. In: *Handbook of Computability and Complexity in Analysis*. Ed. by Vasco Brattka and Peter Hertling. Cham: Springer International Publishing, pp. 367–417. ISBN: 978-3-030-59234-9. DOI: [10.1007/978-3-030-59234-9\\_11](https://doi.org/10.1007/978-3-030-59234-9_11). URL: [https://doi.org/10.1007/978-3-030-59234-9%7B%5C\\_%7D11](https://doi.org/10.1007/978-3-030-59234-9%7B%5C_%7D11).
-  Brattka, Vasco and Arno Pauly (2018). “On the algebraic structure of Weihrauch degrees”. In: *Logical Methods in Computer Science* 14.4. ISSN: 18605974. DOI: [10.23638/LMCS-14\(4:4\)2018](https://doi.org/10.23638/LMCS-14(4:4)2018).

## References II

-  Dzhafarov, Damir, Denis Hirschfeldt, and Sarah Reitzes (2020). “Reduction Games, Provability, and Compactness”. In: *To Appear*.
-  Dzhafarov, Damir D, Jun Le Goh, Denis R Hirschfeldt, Ludovic Patey, and Arno Pauly (2020). “Ramsey’s theorem and products in the Weihrauch degrees”. In: *Computability 9*, pp. 85–110. ISSN: 2211-3576. DOI: 10.3233/COM-180203.
-  Simpson, Stephen G (2009). *Subsystems of Second Order Arithmetic*. 2nd ed. Cambridge: Cambridge University Press. ISBN: 9780521884396. DOI: DOI:10.1017/CB09780511581007. URL: <https://www.cambridge.org/core/books/subsystems-of-second-order-arithmetic/EA16CB4305831530B7015D6BC46B7424>.

# Primitive Recursion

Primitive recursion: A for-loop for  $f$  whose initial value is  $g$  and whose iterative step is  $h$ .  $f(\ulcorner \ \urcorner) = f(\ulcorner x \urcorner) = f(\ulcorner x, y \urcorner) = 0$  for all  $x$  and  $y$ , and

$$f(\ulcorner k, 0, x_1, \dots, x_k \urcorner) = g(\ulcorner x_1, \dots, x_k \urcorner)$$

and

$$f(\ulcorner k, S(y), x_1, \dots, x_k \urcorner) = h(\ulcorner y, f(\ulcorner k, y, x_1, \dots, x_k \urcorner), x_1, \dots, x_k \urcorner)$$