Ordinal Reductions on Choice Principles

MSc Thesis *(Afstudeerscriptie)*

written by

Jasper Stammes
(born 4 September 1999 in Bilthoven)

under the supervision of **Lorenzo Galeotti, UvA** and **Benno van den Berg, ILLC**, and submitted to the Examinations Board in partial fulfillment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam.*

Date of the public defense:	**Members of the Thesis Committee:**
January 29th 2024	Dr. Malvin Gattinger (chair)
	Dr. Franziska Jahnke
	Dr. Robert Passmann
	Dr. Benno van den Berg
	Dr. Lorenzo Galeotti



Institute for Logic, Language and Computation

Abstract

In classical computability theory and computable analysis, two problems' relative computability can be studied through several notions of reducibility. We wish to generalize a few of these notions. Instead of using a classical Turing machine, we consider a model of computation that can handle objects of any cardinality.

The Ordinal Turing Machine (OTM) is a generalization of the classical Turing machine, where computations have unbounded available time and space. In this thesis we will study the relative OTM-computability of construction problems arising from choice principles in set theory, based on similar work by Carl. We will use two existing notions of reducibility: OTM-reductions and ordinal Weihrauch reductions. We also define two similar reductions. In this setting, we prove the first known non-trivial case of OTM-irreducibility.

Contents

1 Introduction **2**

2 Set theoretic preliminaries **5**
2.1 Filters and formulas 5
2.2 Von Neumann-Bernays-Gödel set theory 6
2.3 Constructibility and $0^{\#}$ 8
2.4 Forcing 11
2.5 Computability 14

3 Reducibility and irreducibility **21**
3.1 Construction problems and canonifications 21
3.2 OTM- and Weihrauch reducibility 22
3.3 Weihrauch irreducibility 25
3.4 Comparing forms of choice 27

4 OTM-irreducibility **33**
4.1 OTM-reductions with a bound on the oracle 34
4.2 $\mathsf{PP}_{\mathsf{fin}}$ versus PP 37

5 Reductions with miracle OTM-programs **43**
5.1 The Compactness Theorem 44

6 Conclusion and Future Work **47**

Chapter 1

Introduction

In mathematics, objects can be easy to define, but can at the same time have complex characteristics. A definition such as "all functions from ω to 2" can also *feel* more complex than one like "all primes": we can check for any $n \in \omega$ if it is a prime, but we do not have enough time to go through the whole of ω. This feeling of complexity can be formalized in different ways. In classical computability theory, a set of natural numbers that is computable by a Turing machine is seen as less complex than a set that cannot be computed. Fields such as computable analysis and descriptive set theory describe the computability and complexity of (sets of) real numbers.

When two problems are both not computable, one problem could be easier to solve than the other. Thus, when we study their *relative* computability, we may find that a program that is given access to an oracle for problem A can use this to compute a solution to problem B. Multiple such notions of *reducibility* exist. These can differ in several dimensions, such as the model of computation, the objects that the problems consider and the extent to which the oracle may be accessed.

Of particular interest to us are problems associated with Π_2-sentences, sentences of the form "for all x there is a y such that so and so". Such a problem is *computable* if there is a program that, when given any instance x as input, returns a solution y satisfying the sentence. Many mathematical theorems or statements are Π_2-sentences. In computable analysis, this is used to study the complexity of theorems from real analysis, see, e.g., [20]. We use one reducibility notion from this field. Assuming that objects are appropriately coded, so that they can be manipulated by a Turing machine, a *Weihrauch reduction* from problem A to B consists of two programs P and Q and an oracle F for B: this program P translates instances of A to instances of B and Q translates solutions of B to solutions of A, so that problem A can be computed by executing P, then F, and then Q.

When the model of computation used is a classical Turing machine, we are

restricted to objects that can be represented on the machines' tape, and are therefore countable. But, a mathematician's intuition about the computability of objects can extend beyond countable objects, and so reducibility notions that work on uncountable objects have also been developed, as done in, e.g., [9, 4]. In this thesis, we are concerned with reducibility notions that employ a model of transfinite computability. Many of these models can be found in [5]. A straightforward generalization of the Turing machine that has an unbounded tape length and also an unbounded computation time is the Ordinal Turing machine (OTM) as defined by Koepke in [15].

The OTM's tape is an element of 2^{Ord}, and can easily be interpreted as the characteristic function of a set of ordinals. In doing so, [15] shows that sets that are computable by an OTM without any input are always sets that are elements of the constructible universe L. However, if we assume that all sets can be well-ordered, we find that there is a coding through which any arbitrary set in the universe V can be formatted onto an OTM's tape. This creates a notion of computability for Π_2-sentences about sets: a sentence is *OTM-computable* or *OTM-effective* if there is an OTM-program P that, given any code for a set x, computes a code for a y satisfying the sentence.

A set x is coded onto a tape via a certain well-ordering, so a program with x as input can also use this extra information. As opposed to common practice in computable analysis, in this thesis, as in [3], we choose not to accept computations that depend on this well-ordering. Thus, we only consider *code-independent* OTM-programs, those that, for every possible code for x, will output a code for the same set y. These programs compute a *canonification*, a single-valued class function $F : V \to V$. Just as the main inspiration of this thesis, [3], we will focus on theorems and statements in set theory that are variants of choice. By requiring code-independent OTM-programs, these are not OTM-effective.

In [3], two notions of reducibility are defined for this context. A *miracle OTM-reduction* from A to B is done by a program P^F computing A by making use of an oracle $F : V \to V$ that solves B. The oracle may be accessed any number of times. The stronger *ordinal Weihrauch reduction* is like the earlier mentioned Weihrauch reduction, but where the programs P and Q are code-independent OTM-programs. Generalized versions of Weihrauch reducibility and techniques used for proving (ordinal) Weihrauch irreducibility come from works such as [9, 2, 4]. There are no existing techniques for proving non-trivial OTM-irreducibility.

This thesis studies the set theory underlying relative computability of set theoretical statements (Chapters 2 and 3) and proves the first known non-trivial case of OTM-irreducibility (Chapter 4). Also, we define two new types of reductions (Chapters 4 and 5). We use these to compare the relative computability of three classically equivalent versions of the Boolean Prime Ideal Theorem (Chapters 3 and 5).

The graph below presents the Weihrauch reductions mentioned in this thesis. A (dotted) arrow from A to B should be read as "problem A is Weihrauch reducible to problem B". If we know that the opposite reduction does not exist, a full arrow is used.

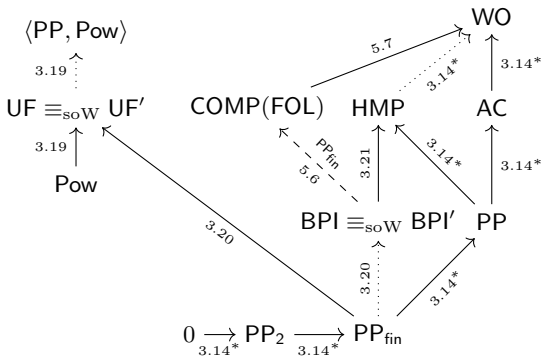


Figure 1.1: Graph of Weihrauch reductions. Numbers refer to propositions in this thesis, and an asterisk indicates a result in [3].

In the following graph, arrows represent miracle OTM-reductions instead. In the case of $\mathsf{PP}_{\mathsf{fin}} <_{\mathrm{OTM}} \mathsf{PP}$, the fact $\mathsf{PP}_{\mathsf{fin}} \leq_{\mathrm{OTM}} \mathsf{PP}$ is from [3].



Figure 1.2: Graph of miracle OTM-reductions. Numbers refer to propositions in this thesis, and an asterisk indicates a result in [3].

Description of the mentioned problems::

PP_2: Every 2-element set has an element.

$\mathsf{PP}_{\mathsf{fin}}$: Every finite non-empty set has an element.

PP: Every non-empty set has an element, *Picking Principle*.

AC: Every set has a choice function, *Axiom of Choice*.

WO: Every set has a well-ordering, *Well-Ordering Theorem*.

HMP: Every partial order has a maximal chain which extends some chain, *Hausdorff Maximal Principle*.

BPI/BPI′: Every Boolean Algebra has a prime ideal (which extends some ideal), *Boolean Prime Ideal Theorem*.

UF/UF′: Every set has an ultrafilter (extending some filter), *Ultrafilter Theorem*.

Pow: Every set has a power set.

COMP(FOL): Every finitely satisfiable set of formulas is satisfiable, *Compactness Theorem*.

Chapter 2

Set theoretic preliminaries

2.1 Filters and formulas

The studying of relative computability of statements requires us to first fix the definitions in these statements. We use terminology from the seminal book on set theory by Jech [13].

Definition 2.1. A **Boolean Algebra** is a set B with bottom and top elements 0 and 1, binary operations $\vee$ and $\wedge$ and unary operation $\neg$, such that $\vee$ and $\wedge$ are commutative and distributive and for all $u \in B$ the following hold:

$$\begin{aligned} u \vee 0 &= u, & u \wedge 1 &= u, \\ u \vee \neg u &= 1, & u \wedge \neg u &= 0. \end{aligned}$$

A Boolean algebra also defines a partial order $(B, \leq)$ by letting $u \leq v$ if and only if $u \vee v = v$, and the Boolean algebra's operations can also be defined from such a partial order, by defining $u \wedge v$ and $u \vee v$ as the $\leq$-infimum and $\leq$-supremum of u and v respectively.

We will represent Boolean algebras mainly as partial orders.

Definition 2.2. Let $(P, \leq)$ be a partial order. A **filter** on $(P, \leq)$ is a set $F \subseteq P$ such that

(i) $F \neq \emptyset$,

(ii) if $x \leq y$ and $x \in F$ then $y \in F$,

(iii) if $x, y \in F$ then there is a $z \in F$ such that both $z \leq x$ and $z \leq y$.

A filter U is an **ultrafilter** if it is a proper subset of P and

(iv) if F is a filter on P such that $U \subseteq F$, then $F = U$ or $F = P$.

If X is a set, an **ultrafilter** on X is an ultrafilter on partial order $(\mathcal{P}(X), \subseteq)$. This partial order also defines a Boolean algebra with operations $\cup, \cap$ and taking complements.

Lemma 2.3 ([13, Lemma 7.4 & p.81])**.** If $(B, \leq)$ defines a Boolean algebra, then U is an ultrafilter on $(B, \leq)$ if and only if $\forall x \in P\ (x \in U \vee \neg x \in U)$.

Definition 2.4. A **(prime) ideal** is the dual of an (ultra)filter: for a partial order $(B, \leq)$, $I \subset P$ is a (prime) ideal on $(B, \leq)$ if and only if I is an (ultra)filter on $(B, \geq)$.

The following definitions we will use in the required background theory.

Definition 2.5 (Lévy Hierarchy)**.** A **bounded** or Δ_0-formula is a formula that is (equivalent to) a sentence without any unbounded quantifiers (quantifiers that are not of the form $\forall x \in X$). We have $\Pi_0 = \Delta_0 = \Sigma_0$.

We then recursively define a formula to be Π_{n+1} if it is equivalent to a formula of the form $\forall x \psi$ with ψ a Σ_n-formula, and a formula to be Σ_{n+1} if it is equivalent to a formula of the form $\exists x \psi$ with ψ a Π_n-formula.

Our special attention goes out to Π_2-sentences, since many choice principles are Π_2.

Definition 2.6. Let ZF^- be like ZF (Zermelo-Fraenkel set theory) but without the axiom of power set.

We note that omitting the axiom of power set is commonplace in constructive set theories, due to the impredicativity (self-reference) of the statement [19]: its Π_3-formula $\forall x \exists p \forall z (z \subseteq x \to z \in p)$ has the innermost quantifier $\forall z$ ranging over the whole universe, including set p. In constructive set theories bounded quantifiers are preferred.

2.2 Von Neumann-Bernays-Gödel set theory

In this thesis we will be working with von Neumann-Bernays-Gödel (NBG) set theory as our metatheory. NBG is a set theory that extends ZFC to include a notion of classes in its language. And crucially, ZFC and NBG agree on any sentence that is in the language of ZFC (NBG restricted to sentences about only sets).

ZFC has a naive concept of classes such as the class of ordinals Ord and the von Neumann universe V already. In NBG all sets are also classes. To explicitly refer to a class that is not a set, we use the term *proper class*. Following convention, uppercase variables refer to classes and lowercase variables to sets. The difference between a set and a proper class is the following:

Definition 2.7. In NBG, a class x is a set if and only if $\exists C\ (x \in C)$.

We do not give a full axiomatization of NBG, this can be found in [17, Chapter 4]. We have the following key theorem:

Theorem 2.8 (Class Existence Theorem, [17, Prop. 4.4])**.**
Let $\varphi(x_1, \ldots, x_n, C_1, \ldots, C_m)$ be a formula with no quantifiers ranging over classes. Then for all classes $C_1, \ldots, C_m$, there is a class A such that

$$\forall x_1, \ldots, x_n[(x_1, \ldots, x_n) \in A \text{ iff } \varphi(x_1, \ldots, x_n, C_1, \ldots, C_m)].$$

Using the Class Existence Theorem we can formally define class Ord by a formula for "x is an ordinal" and can define class V by $x = x$.

Useful to us is that NBG includes the following axiom:

Definition 2.9. Let the **Axiom of Global Choice** be

$$\exists G : V \to V \; \forall x(x \neq \emptyset \to G(x) \in x).$$

We needed the notion of class to formulate this axiom, since no set can possible be a function with every set in its domain. The axiom of global choice is well-known to be strictly stronger than the classical axiom of choice.

Lemma 2.10. In NBG, there is a bijection $H : \mathrm{Ord} \to V$.

Proof. Using $V = \bigcup_{\alpha \in \mathrm{Ord}} V_\alpha$ and global choice function G, we can recursively define

$$H(\alpha) = G(V_\beta \setminus H[\alpha])$$

where β is minimal such that $V_\beta \setminus H[\alpha] \neq \emptyset$. □

Lemma 2.11. Let $R \subset V \times V$ be a binary class relation, and F be a (class) function such that $F \subset R$. Then there is a proper class function $F' : V \to V$ extending F such that $\forall x \in \mathrm{dom}(R)\,(x, F'(x)) \in R$.

Proof. We will define F' to extend F by letting, for every x in V, define $F'(x)$ to be the element that comes first in enumeration H from Lemma 2.10 satisfying the condition $\forall x \in \mathrm{dom}(R), (x, F'(x)) \in R$. By the Class Existence Theorem 2.8, we can define F' as follows:

$$\begin{aligned}(x, y) \in F' \text{ iff } & (x, y) \in F \\ & \vee (x \notin \mathrm{dom}(F) \wedge (x, y) \in R \wedge \forall \alpha < H^{-1}(y)\; (x, H(\alpha)) \notin R) \\ & \vee (x \notin \mathrm{dom}(R) \wedge y = H(0)),\end{aligned}$$

such that $F' : V \to V$ is a proper class function. □

These class functions $F : V \to V$ are important for our purposes later on.

2.3 Constructibility and $0^{\#}$

Assuming that V is a model of ZFC does not fix it to be a specific model. It could be that V is the simplest such model: the constructible universe. Later in this section we describe an axiom that makes it impossible for V to be the constructible universe.

2.3.1 The constructible universe L

Definition 2.12. Let M be a model. A subset $X \subseteq M$ is **definable over** M if $X = \{x \in M \mid \varphi(x, a_1, \ldots, a_n)\}$ for some formula φ and some tuple of elements with all $a_1, \ldots, a_n \in M$. And we let

$$\mathrm{Def}(M) = \{X \subset M \mid X \text{ is definable over } M\}.$$

By letting e.g. $\varphi(x) := x = x$, we see that M is definable over M.

Definition 2.13. We recursively define

$$\begin{aligned} L_0 &= \emptyset, \\ L_{\alpha+1} &= \mathrm{Def}(L_\alpha), \\ L_\lambda &= \bigcup_{\alpha<\lambda} L_\alpha \quad \text{for limit } \lambda, \end{aligned}$$

and let the **constructible universe** be

$$L = \bigcup_{\alpha \in \mathrm{Ord}} L_\alpha.$$

Recall that the von Neumann universe V is defined in the same manner, but with $\mathcal{P}$ instead of Def.

Lemma 2.14. We collect the following facts:

(i) if α is an ordinal, then $\alpha \notin L_\alpha$ but $\alpha \in L_{\alpha+1}$,

(ii) all L_α and L are transitive,

(iii) if α is finite, $V_\alpha = L_\alpha$,

(iv) for all infinite α, $|L_\alpha| = |\alpha|$,

(v) there is a canonical well-ordering of L.

Proof. See [7, Ch. II: Lemma 1.1 & 3.6]. □

Definition 2.15. The **axiom of constructibility**, or $V = L$, is the assertion that every set is in L.

Note that this does not mean that $V_\alpha = L_\alpha$ holds for all ordinals α, since sets may be added at a later stage in L than in V.

We will not accept $V = L$ as an axiom, but quite the opposite: in the next chapter we we wish to assert that $V \neq L$. To that end we must assume another axiom, since L itself is a model of $\mathrm{ZF} + V = L$.

2.3.2 Large cardinal axioms

In the 20th century, many assumptions that lead to provable differences between L and V have been studied, many are mentioned in [14]. Often these are large cardinal axioms, which assert that there is some cardinal κ with certain properties. To illustrate how this may work, we mention one simple example before moving to the main focus.

Definition 2.16. A cardinal κ is **weakly inaccessible** if κ is uncountable, regular, and if λ is such that $\lambda < \kappa$, then $\lambda^+ < \kappa$. It is **strongly inaccessible** if $\lambda < \kappa$ implies $2^\lambda < \kappa$.

The existence of a weakly inaccessible κ cannot be proven from ZFC, if it is consistent, and therefore the statement "there exists a weakly/strongly inaccessible cardinal κ" has to be assumed as any other axiom [13, Theorem 12.12].

We will assume the axiom "$0^\#$ exists". Although $0^\#$ is a real number, the existence of $0^\#$ implies the existence of certain large cardinals including strongly inaccessible cardinals, see Lemma 2.23. It also contradicts $V = L$. We will be discussing some consequences of $0^\#$, but note that many other results are irrelevant to us and omitted.

For more complete accounts of $0^\#$, see e.g. [13, Ch. 18], [14, Ch. 9] or [7, Ch. V].

2.3.3 Consequences of $0^\#$

In model theory, a set of objects is called indiscernible if every formula has the same truth value for every tuple of objects in the set. We wish to consider an indiscernible set of ordinals, but if we take the formula $\varphi(x_1, x_2) = x_1 \in x_2$, then $\varphi(\alpha_1, \alpha_2) \leftrightarrow \neg\varphi(\alpha_2, \alpha_1)$ for every $\alpha \neq \beta$. We adjust for this as folllows:

Definition 2.17. A class $X \subset \mathrm{Ord}$ is called **L-order-indiscernible** if for every n-ary formula of set theory φ, and every two strictly increasing sequences $\alpha_1 < \alpha_2 < \ldots < \alpha_n$ and $\beta_1 < \beta_2 < \ldots < \beta_n$ consisting of elements from X, it holds that

$$L \vDash \varphi(\alpha_1, \alpha_2, \ldots, \alpha_n) \longleftrightarrow \varphi(\beta_1, \beta_2, \ldots, \beta_n).$$

The following definitions are well-known in set theory.

Definition 2.18. If X is a class of ordinals, we say that X is **closed** if and only if $\forall S \subset X : S \neq \emptyset \rightarrow \sup(S) \in X$.

If κ is a cardinal, a class of ordinals X is **unbounded in** κ if and only if $\forall \alpha < \kappa, \exists \beta \in X \ \kappa > \beta > \alpha)$, and a class is **unbounded** if this holds for $\kappa = \mathrm{Ord}$.

Definition 2.19 ([13, Theorem 18.1])**.** We say that a proper class of ordinals I is a class of **Silver indiscernibles** if it is unique in satisfying the following:

(i) I is closed and unbounded,

(ii) I is L-order indiscernible,

(iii) I generates L in the sense that every $x \in L$ is definable through some n-ary formula φ and $(\alpha_i \in I)_{i \leq n}$ as the only $y \in L$ such that $\varphi(y, \alpha_1, \ldots, \alpha_n)$ holds,

(iv) I is unbounded in every uncountable cardinal.

Remark: This definition differs from the one in [13, Theorem 18.1], but the equivalence is shown in [8, Theorem 3.8].

From item (iv), we may derive the following:

Corollary 2.20. Suppose that I is a class of Silver indiscernibles. The following are properties of I:

(i) For any uncountable κ, $\kappa \in I$.

(ii) For every uncountable κ, there are $|\kappa|$ many elements $\gamma \in I$ for which $\gamma < \kappa$.

Proof. (i) Since I is unbounded in every uncountable cardinal, if κ is a cardinal then there is a sequence cofinal for κ. And I is closed, hence $\kappa \in I$.

(ii) Sequences that are cofinal in κ must be of cardinality $\mathrm{cf}(\kappa)$, proving the statement for all regular cardinals. If κ is a singular cardinal instead, it follows by induction on all $\lambda < \kappa$.

□

Definition 2.21 (**Zero sharp**)**.** For any n-ary formula φ, let $\ulcorner\varphi\urcorner$ denote the Gödel number of φ. We define $0^{\#} = \{\ulcorner\varphi\urcorner \mid L_{\aleph_\omega} \vDash \varphi(\aleph_1, \aleph_2, \ldots, \aleph_n)\}$.

Supposing that a class of Silver indiscernibles I exists, we know that all $\aleph_\alpha \in I$ by Corollary 2.20 and that I is L-order-indiscernible. Then the same sentences are true for any increasing sequence of elements of I. So instead of the $\aleph_i$, we could also have taken any other increasing sequence of elements of I in the definition of $0^{\#}$.

Theorem 2.22. A class of Silver indiscernibles exists iff the set $0^{\#}$ exists..

Proof. See, e.g., [13, Chapter 18]. □

Lemma 2.23 ([7, Th. 2.15])**.** If $0^{\#}$ exists, then every uncountable cardinal κ is strongly inaccessible in L.

Proof. Recall Definition 2.16; we have to show that

$$L \vDash \text{“}\kappa \text{ is an uncountable regular limit cardinal”},$$

and that for all L-cardinals λ, $\lambda < \kappa$ implies $L \vDash 2^\lambda < \kappa$.

The following are known facts [7, Th. 2.15]:

$$L \vDash \aleph_1 \text{ is regular,}$$
$$L \vDash \aleph_\omega \text{ is a limit cardinal,}$$
$$L \vDash GCH.$$

We know from Corollary 2.20 that $\aleph_1, \aleph_\omega \in I$, so then by indiscernibility, any $\alpha \in I$ must also be a regular limit cardinal in L. By the same corollary we have $\kappa \in I$, and since GCH states $\lambda^+ = 2^\lambda$ for all λ, $L \vDash$ "κ is a regular limit cardinal" implies that ($\lambda < \kappa$ implies $L \vDash 2^\lambda < \kappa$). Therefore κ is strongly inaccessible in L. □

Corollary 2.24 ([7, Th. 2.15])**.** If $0^\#$ exists, then $V \neq L$.

Proof. By the lemma above, $\aleph_1$ is inaccessible in L. Thus in particular we have $|\mathcal{P}^L(\omega)| < \aleph_1$, and so $\mathcal{P}^L(\omega)$ is countable. However, $\mathcal{P}(\omega)$ is not countable, thus $\mathcal{P}^L(\omega) \neq \mathcal{P}(\omega)$ and so $V \neq L$. □

Lemma 2.25. Assume that $0^\#$ exists. Let ω_1 be the first uncountable ordinal in V. If $\alpha < (\aleph_{\omega_1})^L$, every $x \in L_\alpha$ is countable.

Proof. For finite α this is trivial. Suppose that α is infinite. By letting $\kappa = \aleph_1$, from item (ii) in Corollary 2.20 we find that there are $\aleph_1$-many elements of I below ω_1. We know that $L \vDash \omega_1$ is a cardinal, so all the other elements of I are cardinals in L. Hence if $\alpha < (\aleph_{\omega_1})^L$, then $\alpha < \omega_1$, so α is countable. Since α is infinite, by Lemma 2.14, $|L_\alpha| = |\alpha|$. Thus L_α is countable, and so is every $x \in L_\alpha$. □

Note that this lemma also shows that there are an uncountable number of L-cardinals that are countable in V. In the next section we will use the fact that many levels of the constructible universe are countable in V.

2.4 Forcing

We will not give a full account of forcing, but for this we refer to [13, Chapter 14.]. Let us briefly mention what function forcing may serve. We will only be interested in relatively simple forcing notions.

Definition 2.26. Let P be a partial order. A set D is **dense** if for every $p \in P$, there is a $q \in D$ such that $q \leq p$. A set $G \subset P$ is a **filter** if

(i) $G \neq \emptyset$ and $G \neq P$,

(ii) if $p \leq q$ and $p \in G$ then $q \in F$,

(iii) if $p, q \in G$ then there is a $r \in G$ such that both $r \leq p$ and $r \leq q$.

For any M, a filter $G \subset P$ is M**-generic** over P if

(iv) for all dense (in P) $D \in M$, $D \cap G \neq \emptyset$.

Partial orders are called *forcing notions* when we use them for forcing. Generic filters are crucial in forcing. The following summarises their main use:

Theorem 2.27 ([13, Th. 14.5])**.** Let $M \vDash \mathrm{ZF}^-$ be a transitive model and $P \in M$ a forcing notion. If G is M-generic over P, then there is a transitive model $M[G]$ such that

(i) $M[G] \vDash \mathrm{ZF}^-$,

(ii) if $M \vDash \mathrm{ZF}$, then $M[G] \vDash \mathrm{ZF}$,

(iii) if $M \vDash \mathrm{ZFC}$, then $M[G] \vDash \mathrm{ZFC}$,

(iv) $M \subset M[G]$ and $G \in M[G]$,

(v) if N is transitive such that $M \subset N$ and $G \in N$, then $M[G] \subset N$.

Proof. See [13, Theorem 14.24.]. □

Definition 2.28 ([16, Ch. VII, Def. 5.1])**.** Let M be a model and $A, B \in M$. Then the following defines a partial order

$$\begin{aligned}\mathrm{Fin}(A,B) &= \{\, p \subset A \times B \mid p \text{ injective function with } \mathrm{dom}(p) \subset A, |p| < \omega \,\},\\ &\quad \text{with } \; p \leq q \;\text{ iff }\; p \supseteq q.\end{aligned}$$

This is the partial order of the well-known forcing $P = \mathrm{Fin}(\omega, \omega_1^M) \in M$ that collapses the formerly uncountable ω_1^M to be countable in $M[G]$. In general, we have

Lemma 2.29 ([16, Ch. VII, Lemma 5.2])**.** Let M be a model with $A, B \in M$ and $P = \mathrm{Fin}(A, B) \in M$. If G is an M-generic filter, then $\bigcup G \in M[G]$ is a bijection between A and B.

Proof. See [16, Ch. VII, Lemma 5.2]. □

The existence of a generic filter cannot simply be assumed, but we can prove this existence in certain cases.

2.4.1 Forcing and $0^\#$

The simplest case in which we can prove the existence of a generic filter is well-known:

Lemma 2.30 (Rasiowa-Sikorsi)**.** Let M be countable, P a partial order in M, and $p \in P$. Then an M-generic filter on P containing p exists.

Proof. See [13, Lemma 14.4]. It follows from a countable enumeration $(D_i)(i \in \omega)$ of all of the dense sets, This implies a set of conditions $\{p_i \in D_i\}$ generating a generic filter. □

Under the presence of $0^{\#}$, we can identify many countable sets.

Theorem 2.31. Assume that $0^{\#}$ exists. The following hold:

(i) If $M = L$ and $P \in M$ or countable, then an M-generic filter G on P exists.

(ii) If M is any model and $P \in L_\alpha$ for some $\alpha < (\aleph_{\omega_1})^L$, then an M-generic filter H on P exists.

(iii) If M is any model and $P \in L_\alpha[G]$ for a cardinal-preserving extension G for some $\alpha < (\aleph_{\omega_1})^L$, then an M-generic filter H on P exists.

Proof. (i) The countability of P implies the countability of $\mathcal{P}^L(P)$, for if it were uncountable it would contradict the strong inaccessibility of $\aleph_1$ in L (see Lemma 2.23). Sets that are dense in P are elements of $\mathcal{P}^L(P)$ and then there is a countable enumeration of all dense sets. Then by (the proof of) the Rasiowa-Sikorski lemma, a generic H exists.

(ii) By Lemma 2.25, P is countable and the same reasoning applies.

(iii) Since G is a cardinal-preserving extension, by the relativised result of Lemma 2.25, P is countable and the same reasoning applies.

□

We can use this as a criterion for the existence of generic filters. If at any point we assert that some generic filter exists, it will be a consequence of (perhaps several applications of) this theorem.

2.4.2 Product forcing

After forcing with a notion to obtain a model $M[G]$, forcing with another notion $Q \in M[G]$ and $M[G]$-generic H on Q results in a model $M[G][H]$. For all $Q \in M$, product forcing is the technique giving a notion and generic filter such that $M[G][H]$ can be obtained with a single extension $M[G \times H]$.

Definition 2.32. Let $P = (P, \leq_P), Q = (Q, \leq_Q)$ be forcing notions in a model M. Their **product** $P \times Q$ is the notion in M defined by

$$(p, q) \leq (p', q') \quad \text{iff} \quad p \leq_P p' \ \wedge \ q \leq_Q q'.$$

It is easily shown that if G is an M-generic filter G on $P \times Q$ and G_1 and G_2 the projections of G onto P and Q respectively, then G_1 and G_2 are M-generic filter (on P respectively Q),

Lemma 2.33. Let M be a model and P, Q forcing notions in M. Let G and H be filters on P and Q respectively. The following are equivalent:

(i) $G \times H$ is M-generic over $P \times Q$,

(ii) G is M-generic over P and H is $M[G]$ generic over Q,

(iii) H is M-generic over Q and G is $M[H]$-generic over P.

And if these hold, then $M[G][H] = M[G \times H] = M[H][G]$.

Proof. In Kunen [16, Ch. VIII, Theorem 1.4.] and Jech [13, Lemma 15.9.]. □

Definition 2.34. Let P and Q be forcing notions in M. If G_1 and G_2 are filters such that one of the conditions in Lemma 2.33 is met, we say that they are **mutually generic filters**.

Lemma 2.35. Let M be a model. Suppose that G_1 and G_2 are mutually generic filters of forcing conditions P and Q. Then $M[G_1] \cap M[G_2] = M$.

Proof. From [10].

It trivially holds that $M[G_1] \cap M[G_2] \supseteq M$. For the other direction, suppose $x \in M[G_1] \cap M[G_2]$. Using $\in$-induction, we may assume that elements of x are in M: $x \subset M$. By assumption, there is a P-name σ and Q-name τ such that $\sigma_{G_1} = x = \tau_{G_2}$. So there must be a condition $(p, q) \in P \times Q$ forcing that $\tau = \sigma$ (both canonically read as $P \times Q$-names). And adding our earlier assumption that $x \subset M$, we may strengthen (p, q) such that also holds that $p \Vdash \sigma \subset \check{M}$ and $q \Vdash \tau \subset \check{M}$.

Claim: For every $y \in M$, p *decides* whether or not $y \in \tau_{G_1}$. By deciding, we mean that p forces either $\check{y} \in \tau$ or $\check{y} \notin \tau$. For if p does not decide this for some $y \in M$ then there are two conditions p' and p'' stronger than p, such that $p' \Vdash \check{y} \in \tau$ and $p'' \Vdash \check{y} \notin \tau$. Then if q decides whether or not $y \in \sigma_{G_2}$, assume w.l.o.g. that p' disagrees with q. And if q does not decide this, there is a stronger condition q' that disagrees with p'. Then we have found $(p', q') \leq (p, q)$, such that $(p', q') \Vdash \sigma \neq \tau$, contradicting our earlier assumption.

The claim proves $x \in M$: $p \in M$ and p contains all the info of the elements of x. To be precise: $x \subset M$ means $x \in V_\alpha^M \in M$ for some α. Then since p decides $\check{y} \in \tau$ for all $y \in V_\alpha^M$, we have $x = \{y \in V_\alpha^M \mid p \Vdash \check{y} \in \tau\}$ and so $x \in M$. □

2.5 Computability

2.5.1 Transfinite computation

The Turing machine is widely, but not universally, accepted as a correct model of computability. But by their nature, Turing machines are limited to computing countable objects. Yet, the mathematical intuition of what can be described by an algorithm of any kind goes beyond countable objects. Thus, generalizing the Turing machine to be able to deal with unbounded time and space *may* be the right tool to determine what can or cannot be done *effectively*, following this intuition. For the interested reader, a philosophical motivation for the OTM as apt generalization of the Turing machine can be found in [5, Section 9.3.].

Definition 2.36 (Classical computation)**.** A **Turing Machine** is a tuple $M = \langle Q, q_0, q_f, P \rangle$ where

(i) $Q \subset \omega$ is a finite set of states,

(ii) $q_0, q_f \in Q$ are the starting state and halting state respectively,

(iii) P is the transition function or **program**, $P : Q \times \{0,1\} \to Q \times \{\mathrm{L}, \mathrm{S}, \mathrm{R}\} \times \{0,1\}$.

Many equivalent definitions exist, and we will often refer not to M but to a program P.

Definition 2.37 ([15, Def. 2])**.** Let Ord denote the whole class of ordinals and let $M = \langle Q, q_0, q_f, P\rangle$ be a Turing machine as above.

The **configuration** of the OTM at time $\theta \in \mathrm{Ord}$ is a tuple $\langle T(\theta), H(\theta), Q(\theta)\rangle$ where $T(\theta) \in \{0,1\}^{\mathrm{Ord}}$ is a tape of ordinal length, $H(\theta) \in \mathrm{Ord}$ is the position of the head and as before, $Q(\theta) \in Q$ is the state.

Suppose we are given configuration $\langle T(\theta), H(\theta), Q(\theta)\rangle$ such that $Q(\theta) \neq q_f$. Let b be the local tape content $T(\theta)_{H(\theta)}$. Let $(Q', D, b') = P(Q(\theta), b)$. The next step of the computation is $\langle T(\theta+1), H(\theta+1), Q(\theta+1)\rangle$, with

$$
\begin{aligned}
T(\theta+1)_\xi &= \begin{cases} b' & \text{if } \xi = H(\theta) \\ T(\theta)_i & \text{otherwise,} \end{cases} \\
H(\theta+1) &= \begin{cases} H(\theta)+1 & \text{if } D = \mathrm{R} \\ H(\theta) & \text{if } D = \mathrm{S} \\ H(\theta)-1 & \text{if } D = \mathrm{L} \wedge H(\theta) \text{ is a successor ordinal} \\ 0 & \text{otherwise,} \end{cases} \\
Q(\theta+1) &= Q'.
\end{aligned}
$$

At limit times $\theta \in \mathrm{Ord}$, the configuration is given by the inferior limits:

$$
\begin{aligned}
\forall \xi \in \mathrm{Ord}\ T(\theta)_\xi &= \liminf_{\sigma \to \theta} T(\sigma)_\xi \\
H(\theta) &= \liminf_{\sigma \to \theta} H(\sigma) \\
Q(\theta) &= \liminf_{\sigma \to \theta} Q(\sigma).
\end{aligned}
$$

If at some time θ, $Q(\theta) = q_f$, we say that the tuple of the three functions $T : \theta \to 2^{\mathrm{Ord}}$, $H : \theta \to \mathrm{Ord}$, $Q : \theta \to \omega$ is the **computation of P on input** $T(0)$. If there is no such time θ, the computation never stops and so we say that the computation is the tupple of class functions T, H and Q with domain $\theta = \mathrm{Ord}$.

The important behavior to note is of course the limit step. If the program loops through some loop containing a subloop, after a limit λ-many times of looping, we want the program to first go through the main loop again. Therefore Koepke assumes that the states are ordered as the lines of a computer script are, so that the inferior limit corresponds to the start of this main loop.

We note two important similarities between traditional Turing machines and OTMs.

Lemma 2.38. Let $n \in \omega$. An OTM with n tapes and heads can be simulated by an OTM with one tape and head.

Proof. See [15, Ch. 3]. □

In the light of this lemma, for the sake of clarity we may choose to make use of a finite number of tapes without increasing the computing strength. The tapes can for example be used to store several variables or to simulate multiple OTMs at the same time, or to speak of an input tape and an output tape.

Lemma 2.39 (Halting Problem)**.** Let $(P_n)_{n<\omega}$ be some enumeration of all OTM programs, and let $H = \{n \mid P_n(n) \text{ halts}\}$. There is no OTM program P that, starting with all 0s on the input tape, halts and returns the characteristic function of H.

Proof. The classic diagonalization proof still holds. □

This Lemma can be prased as "H is not *OTM-computable*", but we will use this term to mean something else.

2.5.2 Coding of sets of ordinals

We want to use OTMs for set theory, and need a way of inputting and outputting sets. This is what we call 'coding'. For sake of completeness, we will detail how this works, but this idea is largely taken from [6, Section 2].

As a naive first step, we can code a single ordinal α as its characteristic function χ_α, a string of 0s until a 1 appears on the tape at position α. If we now wish to encode a set of ordinals X, this naive approach fails. Let χ_X be the characteristic function of X, a tape having a 1 at position α if and only if $\alpha \in X$. An OTM reading this tape will find all the $\alpha \in X$, but will never know whether some bigger element might also be in X.

To simplify matters, for this part we use an extra tape. We will call the first `Elements` and the second `Delimiters`. The `Elements` tape will be the characteristic function of X on the relevant domain $0 \leq \alpha \leq \sup(X)$. Then, the `Delimiters` tape will start off with only 0s, until position $\sup(X)$ will have a 1 and the following position a 0. Now an OTM can read for any set X when the code of its elements is over on the `Delimiters` tape. The reason for the 0 following the delimiting 1 is that later on we want to be able to distinguish between a single delimiter and a double delimiter.

The code for ω, for example, is the following:

$$\texttt{Elements} = \overbrace{11111\ldots}^{\omega}00\ldots$$
$$\texttt{Delimiters} = \underbrace{00000\ldots}_{\omega}10\ldots$$

Now let's say we want to code a pair $\langle \alpha, X \rangle$ where $X \in \mathcal{P}(\text{Ord})$ and $\alpha \in X$. Then we just add a `Pointers` tape with a 1 on position α. Next, we encode

a sequence of any length η of the form $\langle\langle\alpha_\gamma, X_\gamma\rangle, \gamma < \eta\rangle$. Since we can only have a finite number of tapes, we cannot create three different tapes for every pair $\langle\alpha_\gamma, X_\gamma\rangle$. Instead we do this by simply stacking the initial segments coding each pair after one another. Since the `Delimiters` tape tells us when the code for one pair $\langle\alpha_\gamma, X_\gamma\rangle$ is over, we can then have the tapes immediately continue with the code for $\langle\alpha_{\gamma+1}, X_{\gamma+1}\rangle$. The end of the sequence of codes is marked on the `Delimiters` tape by a double 1 instead of a single 1.

Note that this is only one out of many possible ways to code ordinals on a tape. This coding has the following properties.

Theorem 2.40 ([6, Lemma 4])**.** Let X be a set of ordinals. Then the following are OTM-computable:

1. the function that, given codes for an ordinal α and a sequence X, returns 1 if $\alpha \in X$, and 0 otherwise,
2. the function that, given codes for an OTM-computable function $f : \text{Ord} \to \text{Ord}$ and X, returns a code for the image $\{f(\alpha)|\alpha \in X\}$ of X under f,
3. the function that, given codes for $\langle\langle\alpha_\gamma, X_\gamma\rangle|\gamma < \eta\rangle$ and an ordinal $\beta < \eta$, returns a code for $\langle\alpha_\beta, X\beta\rangle$,
4. the function that, given codes for $\langle\langle\alpha_\gamma, X_\gamma\rangle|\gamma < \eta\rangle$ and an ordinal $\beta < \eta$, returns a code for the list obtained by removing the β-th element from $\langle\langle\alpha_\gamma, X_\gamma\rangle|\gamma < \eta\rangle$,
5. the function that, given codes for $\langle\langle\alpha_\gamma, X_\gamma\rangle|\gamma < \eta\rangle$ and $\langle\alpha, X\rangle$, returns a code of the list $\langle\langle\alpha_\gamma, X_\gamma\rangle|\gamma < \eta + 1\rangle$ where $\langle\alpha_\eta, X_\eta\rangle = \langle\alpha, X\rangle$,
6. bounded searches through sets of ordinals.

This lets us treat the OTMs tape(s) as queues, and we will perform many tasks on the tape without explicitly stating the actions the OTM actually does, instead taking a pseudo-code approach.

2.5.3 Coding of sets

Again, this is from [6, Section 2.]. The previous coding describes how to code any pair $\langle\alpha, X\rangle$, with X a set of ordinals, onto a tape, as an element 2^θ for some $\theta \in \text{Ord}$. Now we describe a second coding: to code any set as some pair $\langle\alpha, X\rangle$.

For this we will make use of an ordinal pairing function, let's take the following from [13, p. 30-31]. Let $\mathfrak{g} : \text{Ord} \times \text{Ord} \to \text{Ord}$ be the class function

$$\mathfrak{g}(\alpha, \beta) = \text{the order-type of } \{(\xi, \eta) : (\xi, \eta) < (\alpha, \beta)\}$$

where $<$ is the following well-ordering on Ord^2:

$$\begin{aligned}(\xi, \eta) < (\alpha, \beta) \text{ iff either } &\max(\xi, \eta) < \max(\alpha, \beta)\\ &\text{or } \max(\xi, \eta) = \max(\alpha, \beta) \text{ and } \xi < \alpha\\ &\text{or } \max(\xi, \eta) = \max(\alpha, \beta) \text{ and } \xi = \alpha \text{ and } \eta < \beta.\end{aligned}$$

$\mathfrak{g}$ is one-to-one, uniquely coding any pair of ordinals as an ordinal.

Now given a transitive set T, let $d : \alpha_0 \to T$ be a (one-to-one) enumeration of its elements. We use enumeration d to code T as a set of ordinals by defining

$$C_d(T) = \{\mathfrak{g}(\iota, \xi) : \iota, \xi < \alpha_0 \text{ and } d(\iota) \in d(\xi)\}.$$

By viewing a set as a possibly infinite directed graph (V, E) where the vertices V are the elements of T and the set of arcs (*or directed arrows*) E is such that $(x, y) \in E \leftrightarrow x \in y$, assuming extensionality, the set T can be constructed from the code $C_d(T)$ by letting $E = \{(d(\iota), d(\xi)) : \mathfrak{g}(\iota, \xi) \in C_d(T)\}$.

In this view, the code $C_d(T)$ is simply enumerating all the arcs in T by passing all arcs' endpoints to $\mathfrak{g}$. Different enumerations d and d' will lead to different codes for the same set, so formally we would have to specify which enumeration we are using. Since the choice is often arbitrary and without real consequences, we may omit mentioning which enumeration we use and simply write $C(T)$.

Now finally, this enables us to code any set x, transitive or not, by a pair $\langle \alpha, C\rangle \in \text{Ord} \times \mathcal{P}(\text{Ord})$. First we form $\text{tc}(\{x\})$, the transitive closure of $\{x\}$ (note that $x \in \text{tc}(\{x\})$ but $x \notin \text{tc}(x)$). Then we let $C = C_d(\text{tc}(\{x\}))$ and α the unique ordinal such that $d(x) = \alpha$. We will write c_x or $c(x)$ to refer to a pair $\langle \alpha, C\rangle$ coding x like this.

These codes c_x will be more important than the exact tape content, since we have a more pseudo-code approach to describing OTM-programs. But at any time where we say that some c_x is on the tape, we do not indeed mean the sequence of 0s and 1s but the enumeration of $\text{tc}(\{x\})$.

The correspondence between codes for x and $\text{tc}(\{x\})$ is important. The following is important for reasoning about the cardinalities of outputs of OTM-computations.

Lemma 2.41. Let $x \in V$ be a set so that $\text{tc}(\{x\})$ is infinite, $c_x = \langle \alpha, C_d\rangle$ be a code for x, and $T \in 2^\theta$ be the tape content of length $\theta \in \text{Ord}$ that codes c_x. Then $|T| = |\text{tc}(\{x\})|$.

Proof. For every $z \in \text{tc}(\{x\})$ there are $|z|$-many arcs in C_d. Therefore we have $|C_d| = \sum_{z \in \text{tc}(\{x\})} |z|$. Naturally $1 \leq |z| \leq |\text{tc}(\{x\})|$ for all $z \neq \emptyset$, so we get $|\text{tc}(\{x\})| \leq |C_d| \leq |\text{tc}(\{x\})| \cdot |\text{tc}(\{x\})| = |\text{tc}(\{x\})|$, and thus $|C_d| = |\text{tc}(\{x\})|$.

We observe $|T| = |\texttt{Elements}| + |\texttt{Delimiters}| + |\texttt{Pointers}| = |\texttt{Elements}|$. The coding on $|\texttt{Elements}|$ works by stacking all the $\gamma \in C_d$ on the `Elements` tape, thus $|\texttt{Elements}| = \sum_{\gamma \in C_d} |\gamma|$. Now $\gamma = \mathfrak{g}(\xi, \eta)$ for some $\xi, \eta \in d$. Then since $|\xi| \leq |\text{tc}(\{x\})|$ and $|\eta| \leq |\text{tc}(\{x\})|$, we have $|\gamma| = |\mathfrak{g}(\xi, \eta)| \leq |\text{tc}(\{x\})|$, and so $|\texttt{Elements}| = |C_d| \cdot |\text{tc}(\{x\})| = |\text{tc}(\{x\})|$. □

Lemma 2.42. There is no OTM-program P and no code c_x for an infinite set x such that P, when run on input c_x, halts and outputs a code for a set y with $|\text{tc}(\{y\})| > |\text{tc}(\{x\})|$. As a corollary, power sets cannot be computed.

Proof. Suppose for contradiction that there is such a program. Let $|T_x|$ and $|T_y|$ be the length of the part of the tape that code c_x and c_y respectively. By 2.41, $|T_x| = |\mathrm{tc}(\{x\})|$ and $|T_y| = |\mathrm{tc}(\{y\})|$. Then the OTM-program P has an input of size $|T_x|$ and output of size $|T_y| > |T_x|$. However, it is a known fact that halting OTM-computations can't output larger infinite cardinalities than their input, see e.g. [5, p. L. 8.6.3]. □

Definition 2.43. We define *decode* to be the map from any code c_x to set x coded by it.

Lemma 2.44. Let M be a transitive model of ZF^- and let $c_x \in M$. Then $\mathit{decode}(c_x) = x \in M$.

Proof. By $\in$-induction on x. This trivially holds for $x = \emptyset$.

Let $c_x = \langle \alpha, C_d \rangle$, for some $d : \alpha_0 \to \mathrm{tc}(\{x\})$ and for some α_0. By transitivity, $C_d \in M$. Then for every $y \in x$, let $c_y = \langle \beta, C_d \rangle$ be such that $d(\beta) = y$. Then $c_y \in M$, and so by $\in$-induction we have $y \in M$.

Now $d^{-1}[x] = \{\gamma < \alpha_0 \mid \mathfrak{g}(\gamma, \alpha) \in C_d\} \in M$, and so by Replacement we get $\{\mathit{decode}(\langle \gamma, C_d \rangle) \mid \gamma \in d^{-1}[x]\} \in M$. But for every such γ, we have $\mathit{decode}(\langle \gamma, C_d \rangle) = y_\gamma \in x$ and for every $y \in x$ we have a $\gamma_y \in d^{-1}[x]$. So we find $\{y \mid y \in x\} = x \in M$. □

Theorem 2.45. Let M be a transitive model of ZF^-. For any OTM-program P, if $c_x \in M$, and if $\theta \in M$ with $\theta \in \mathrm{Ord}$ the time needed for the computation of P on c_x, then the full computation of P on c_x is in M.

Proof. See [15, Lemma 1] □

2.5.4 Code-independence

A code for a set x depends on the exact well-ordering for $\mathrm{tc}(\{x\})$ that is used. For a fixed $\mathrm{tc}(\{x\})$ there may be many different well-orderings, so x may be coded by a numerous number of different codes. Since an OTM has access to the coded set, it might use the specific features of the code, and not just those of the set that is coded by it. Thus, on a set-level, OTMs generally do not compute single-valued functions, but multi-valued functions.

In computable analysis, these multi-valued functions are accepted. However, here we will usually restrict our attention to OTM-programs computing single-valued functions:

Definition 2.46. If an OTM-program P is such that for all $x \in V$ and all codes c_x, c'_x for x, it holds that $\mathit{decode}(P(c_x)) = \mathit{decode}(P(c'_x))$, it is called a **functional** or **code-independent** program. We write F_P for the class function $F_P : V \to V, x \mapsto \mathit{decode}(P(c_x))$ it computes.

Lemma 2.47. Let x, y be sets. There are code-independent programs computing the following:

(i) $x = y$, $x \in y$,

(ii) The set $\{y \in x \mid \varphi(x, y)\}$ for an OTM-computable property $\varphi(v_1, v_2)$,

(iii) $x \cup y$, $x \times y$, $\bigcup x$, ω,

(iv) $\mathcal{P}^{<\omega}(x)$, the set of all finite subsets of x.

Proof.

(i) - (ii) See [6, Lemma 11.].

(iii) Also in [6, Lemma 11.]. We shortly note why this is less trivial than the items above. For example, consider $x \cup y$. The different enumerations $d : \alpha_0 \to \mathrm{tc}(\{x\})$, $e : \beta_0 \to \mathrm{tc}(\{y\})$ may not agree on their common domain. The program must therefore form one correct enumeration of $\mathrm{tc}(\{x, y\})$.

(iv) For a proof and pseudo-code for this in the case that x is an ordinal, see [15, Chapter 4.]. This can be generalized to any set x: Given code $c_x\langle\alpha, C_d\rangle$ based on enumeration $d : \alpha_0 \to T_x$, program P collects those ordinals that are an index of the elements of x, so $E(x) := \{\gamma < \alpha \mid d(\gamma) \in x\}$. Then enumerate $E(x)$ by letting $C : \beta \to E(x)$ be defined as follows

$$\begin{aligned} C(0) &= \min\{\gamma \in E(x)\} \\ C(\alpha + 1) &= \min\{\gamma \in E(x) \mid \gamma > C(\alpha)\} \\ C(\lambda) &= \min\{\gamma \in E(x) \mid \forall \alpha < \lambda(\gamma > C(\alpha))\}, \end{aligned}$$

until C enumerates all the elements of $E(x)$.

After P has computed C, P executes the program from [15, Chapter 4.] to obtain $\mathcal{P}^{<\omega}(\beta)$. Finally, for every subset $X \in \mathcal{P}^{<\omega(\beta)}$, P computes $\bigcup\{d(\gamma) \mid \gamma \in X\} \subseteq x$.

□

Chapter 3

Reducibility and irreducibility

In this chapter we will motivate and explore several types of reducibility. Next, we prove theorems that can be used to show that a reduction does not exists and finally, we compare different set theoretical statements.

The definitions and terminology discussed in this chapter are all generalizations developed by Carl in [3] and [4], except for the new notion that we call *code-containing*, Definition 3.4.

3.1 Construction problems and canonifications

Instead of computing statements, we compute the corresponding construction problem. These are defined as pairs of sets:

Definition 3.1. If $R \subset V \times V$ is a binary (class) relation on sets, we call it a **construction problem**. An $x \in \mathrm{dom}(R)$ may be called an **instance of** R, or an R-instance, and any y such that $(x, y) \in R$ may be called an **R-solution to** x.

A class function $F : V \to V$ is a **canonification of** R if $(x, F(x)) \in R$ for every $x \in \mathrm{dom}(R)$.

If φ is a Π_2-sentence in the language of set theory, then there is a Δ_0-formula ψ such that $\varphi \equiv \forall x\, \exists y\, \psi(x, y)$. We say that $F : V \to V$ is a **canonification of** φ if $\psi(x, F(x))$ holds for all $x \in V$.

The mention of R or of φ may be omitted if it is clear from the context. By defining construction problem $R_\psi := \{(x, y) \mid \psi(x, y)\}$, note that F is a canonification for φ if and only if F is a canonification for R_ψ.

Definition 3.2 ([3, Def. 2.])**.** A canonification $F : V \to V$ is called **OTM-computable** only when a functional program P exists such that $F = F_P$.

If there is a canonification F of a binary relation R or of a Π_2-sentence φ such that F is OTM-computable, we say that R or respectively φ is **OTM-computable** or **OTM-effective**.

The prefix OTM- may be dropped when it is clear that we are not talking about classical programs.

Proposition 3.3 ([3, Prop. 3.])**.** If $V = L$, then every Π_2 sentence φ such that φ is a consequence of ZFC is effective.

Proof. See [3, Proposition 3.]. □

Before, assuming the existence $0^\#$ was an assumption that was useful for guaranteeing the existence of generic filters, see Theorem 2.31. This proposition shows that, unsurprisingly, restricting ourselves to the constructible universe would make computations relatively trivial.

We introduce the following definition, which will be a key feature of many later proofs.

Definition 3.4. Let M be a model and F a canonification. We say that F is **code-containing for** M if for every code $c_x \in M$, there is also a code for $c_{F(x)} \in M$. We say that F is **code-containing** if F is code-containing for every transitive ZF^- model M. A construction problem $R \subset V^2$ is **always code-containing** if every canonification of R is code containing.

Whether or not a code for $c_{F(x)}$ exists in a model M depends only on whether an enumeration $d : \alpha \to \mathrm{tc}(F(x))$ exists, which again exists precisely when $\mathrm{tc}(F(x))$ is well-orderable in M. We will later give examples of many natural choices for R that are in fact always code-containing.

3.2 OTM- and Weihrauch reducibility

As is done with classical Turing machines for Turing reducibility $\leq_T$, we will extend the definition of the OTM to include access to (class) function F as an oracle. An OTM-program with access to oracle F will be denoted as P^F and will have one additional `Miracle` tape and a set of inner states $q_m \subset Q$ as miracle states, such that when the run of P^F assumes a miracle state one of two things happens: if the content of the `Miracle` tape does not code a set, then nothing is changed and the run continues. If the `Miracle` tape codes a set $x \in V$, its contents are replaced by a code for the set $F(x)$ and the run continues.

Throughout, we may use the words 'oracle' and 'miracle' interchangeably. This can be understood by a shift of perspectives: if we anthropomorphise a bit, then, to a Turing program, the miracle command has the effect of miraculously replacing the contents of the `Miracle` tape. But, to a canonification, this is simply what an oracle does, answering the question that the machine has posed.

Definition 3.5 ([3, Def.4])**.** Let $R, R' \subseteq V \times V$ be two binary (class) relations. We say that R is **OTM-reducible** to R', or that $R \leq_{\mathrm{OTM}} R'$, if there is a (miracle) OTM-program P^F such that for every canonification F of R', P^F computes a canonification for R.

Suppose that program P^F witnesses that $R \leq_{\mathrm{OTM}} R'$. Note that P^F may use arbitrarily many applications of the miracle command meaning that, when P is given one instance of R, it can use any number of instances of R' in finding one y for which $(x, y) \in R$. We give an example reduction that will also be discussed later in this thesis.

Example 3.6. Let $\mathsf{PP} \subset V^2$ denote the **Picking Principle**: the construction problem for the sentence $\forall x \, \exists y \; x \neq \emptyset \to y \in x$. Thus $(x, y) \in \mathsf{PP}$ if and only if $x = \emptyset$ or $y \in x$.

Let $\mathsf{WO} \subset V^2$ denote the **Well-Ordering Theorem**: the construction problem corresponding to the Well-Ordering Theorem. Thus $(x, y) \in \mathsf{WO}$ if and only if y is a well-order on x.

Then $\mathsf{PP} \leq_{\mathrm{OTM}} \mathsf{WO}$ and $\mathsf{WO} \leq_{\mathrm{OTM}} \mathsf{PP}$.

Proof. We give a more complete version of [3, Proposition 14.].

To show $\mathsf{PP} \leq_{\mathrm{OTM}} \mathsf{WO}$: let F be a canonification of WO. Define a program P^F to take an input c_x and use the miracle command to obtain a code for $F(x) = <$, a well-order on x. Then P^F may find the $<$-least element of x by iterating through all the elements of x and for any $y \in x$, look through $<$ to see if there is any element $z < y$. At some point P^F finds the $<$-least element $y \in x$ and returns it. Then P^F witnesses $\mathsf{PP} \leq_{\mathrm{OTM}} \mathsf{WO}$.

We have to check that the output y does not depend on the code c_x, and that it does not depend on the code for $F(x)$. Now, for any code c_x, program P^F will query a perhaps different code for x on the `Miracle` tape, but still a code for the same set x. Then the set $F(x)$ does not depend on the code c_x, and therefore neither does the y that is the $<$-least element of x depend on the code for $F(x)$ that is returned.

To show that $\mathsf{WO} \leq_{\mathrm{OTM}} \mathsf{PP}$: let F be a canonification of PP. Define a program P^F to take an input c_x and do the following: create an empty set $<$, and then as long as x is not empty, P^F uses F on set x, and obtains a $F(x) \in x$. Then for all $z \in x$, P^F adds pair $(F(x), z)$ to $<$, and then deletes $F(x)$ from x. At some point, x is empty, and P^F returns the code for $<$, which is then a well-order on x. Thus P^F witnesses $\mathsf{WO} \leq_{\mathrm{OTM}} \mathsf{PP}$.

To check that $<$ does not depend on any of the codes, note that the first time that F is used, this is done on x, which is independent of the code c_x. But then $x \setminus \{F(x)\}$ is independent of c_x as well as of $c_{F(x)}$, and so on. □

This may be counter-intuitive: provability-wise the Picking Principle is trivial in any set theory, but the Well-Ordering Theorem can't be proven in ZF [12, Theorem 5.16]. Moreover, the fact that $\mathsf{WO} \leq_{\mathrm{OTM}} \mathsf{PP}$ will be used later to show that PP is not OTM-effective!

Also we briefly note that $\leq_{\mathrm{OTM}}$ is reflexive and transitive, so that we may write $\equiv_{\mathrm{OTM}}$ when both sides reduce to each other.

Weihrauch reductions stem from computable analysis, and are named after Klaus Weihrauch, see [20]. In Weihrauch reductions only one $(x', y') \in R'$ is used to solve the instance of R.

Definition 3.7 ([3, Def. 5.])**.** Let $R, R' \subseteq V \times V$ be binary (class) relations. We say that R is **ordinal Weihrauch reducible** to R', or that $R \leq_{\mathrm{oW}} R'$, if there are functional OTM-programs Q and P such that for every canonification F' of R', $F_Q \circ (F', \mathrm{id}) \circ F_P$ is a canonification of R.

Here $(F', \mathrm{id}) : V \to V^2, x \mapsto (F'(x), x)$.

If P and Q are such that $F_Q \circ F' \circ F_P$ is a canonification of R, we have that R is **strongly ordinal Weihrauch reducible** to R', denoted as $R \leq_{\mathrm{soW}} R'$.

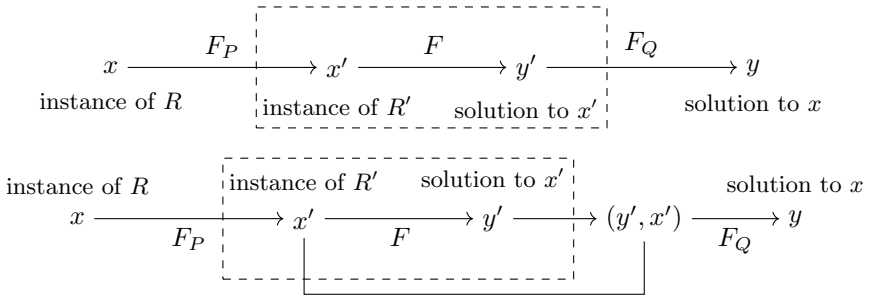


Figure 3.1: Above: strong ordinal Weihrauch reduction. Below: ordinal Weihrauch reduction.

Both relations are reflexive and transitive, so we will use $\equiv_{\mathrm{oW}}$ and $\equiv_{\mathrm{soW}}$ for when both sides are (strongly) ordinal reducible to each other.

3.2.1 Inherently strong Weihrauch problems

Naturally, strong ordinal Weihrauch reducibility implies ordinal Weihrauch reducibility, but in many situations the converse is true as well. This can occur by in some way coding the initial instance x into the output of OTM-program P.

The exact way in which this works may depend on the specifics of the construction problems that are being considered, so will only consider one example.

Lemma 3.8. Let construction problem R' represent the Axiom of Choice: $(x', f') \in R'$ if and only if f' is a choice function on x', i.e. it holds that $\forall w' \in x' : w' \neq \emptyset \to f'(w') \in w'$. Let P and Q be OTM-programs witnessing $R \leq_{\mathrm{oW}} R'$, for some construction problem R. Then there are programs P' and Q' such that the instance given to P' may be used by Q' and in particular, P' and Q' witness $R \leq_{\mathrm{soW}} R'$.

Proof. First, let's recap what program P must do to witness its part of $R \leq_{\mathrm{oW}} R'$:

1. if x is an instance of R,
2. and c_x is any code for x,
3. then $P(c_x)$ halts and returns a code $c_{x'}$,
4. where $c_{x'}$ is a code for x',
5. such that x' is an instance of R'.

By our choice of R' as the Axiom of Choice, x' is a set of sets. Now we can slightly change P into a program P' that does exactly what P does, but when P has finished and computed $c_{x'}$, such that x' is a family of sets, P' will take $c_{x'}$ and instead computes a code for the following set of sets:

$$X' := \{w' \times \{x\} \mid w' \in x'\}.$$

Now for all $w' \in x'$, all elements $v' \in w'$ have been replaced by (v', x).

There is a clear one-to-one correspondence between choice functions F' on X' and choice functions f' on x'. But given any such F', let OTM-program Q' be a small adaptation of Q, first computing from F' the choice function f' on x', and also from F', the initial instance x . Then since P and Q were such that they witness $R \leq_{\mathrm{oW}} R'$, we can conclude that P' and Q' witness $R \leq_{\mathrm{soW}} R'$. □

The construction problems arising from the choice principles that we consider are all inherently strong Weihrauch, but this is not the focus of this thesis and we will not mention it further. We have defined both oW− and soW-reductions since the existing literature always mentions these two types. The difference is significant in other settings, but ours does not justify making this an important distinction.

3.3 Weihrauch irreducibility

There are several techniques available for proving that an oW-reduction does not exist. The ones we discuss here are based on techniques that already existed in classical computability theory and all appear in [3]. Since we previously established many of the facts that are needed, this will not be hard.

3.3.1 Forcing method

For any model M of ZF^- and any $x \in M$ we will write $\mathbb{P}_x$ for $\mathrm{Fin}(\omega, \mathrm{tc}(\{x\}))$. By Lemma 2.29, this is a forcing notion in M to well-order $\mathrm{tc}(\{x\})$ in order type ω. Then if G is a generic filter on $\mathbb{P}_x$, extension $M[G]$ contains a well-order of $\mathrm{tc}(\{x\})$ and thus, crucially, a code c_x for x.

Theorem 3.9 ([3, Th. 9.])**.** Let $R, R' \in V^2$ and $M \vDash \mathrm{ZF}^-$ transitive such that $M \cap \mathrm{Ord} = \mathrm{Ord}$. Suppose there are both a code-containing canonification F of R' and a set x such that M is F-closed, but does not contain a z with $(x, z) \in R$.

Suppose that M and $\mathbb{P}_x$ satisfy one of the conditions in Theorem 2.31. Then $R \not\leq_{\mathrm{oW}} R'$.

Proof. From [3, Theorem 9.]. Because of this theorem's importance we repeat the proof here. Note that we have added the requirement that F is code-containing, to patch a hole in the proof.

Suppose for a contradiction that there are programs P and Q that witness $R \not\leq_{\mathrm{oW}} R'$.

If M and $\mathbb{P}_x$ satisfy one the conditions in Theorem 2.31 then so do M and $\mathbb{P}_x \times \mathbb{P}_x$. Then by Theorem 2.31 and Lemma 2.33 there are two filters G_1 and G_2 that are mutually $\mathbb{P}_x$-generic over M.

Let $c_x \in M[G_1]$ and $c_x \in M[G_2]$ be codes for x. Program P is code-independent, so let $y = \mathit{decode}(P(c_x)) = \mathit{decode}(P(c'_x))$. Now by Lemma 2.44 and Theorem 2.45, we get $y \in M[G_1]$ and $y \in M[G_2]$. The generic filters G_1 and G_2 are mutually generic so, by Lemma 2.35, $M[G_1] \cap M[G_2] = M$ and so $y \in M$.

We know that M is F-closed, so $F(y) \in M$. Since F is code-containing and codes for y exist in $M[G_1]$ and $M[G_2]$, there are also codes $c_{F(y)} \in M[G_1]$ and $c'_{F(y)} \in M[G_2]$. Then with the same reasoning, when Q is executed on these codes it returns codes for some z , and $z \in M$. But since P and Q witness $R \leq_{\mathrm{oW}} R'$, output z is an R-solution to x. But, we assumed such an z to not exist in M, completing the contradiction. □

How effective is this theorem? In the context of choice principles, it is quite useful. There is a large amount of literature on independence proofs (e.g. collections such as [12] or [11]) on models of ZF or ZF$^-$ such that one form of choice is satisfied, but there is a counterexample to another form of choice. If the construction of this model and counterexample are such that one of the conditions of Theorem 2.31 is met, the theorem applies, since these forms of choice are almost always code-containing.

3.3.2 Cardinality method

The following theorem is used to show that construction problems that raise cardinalities may not be reduced to problems that do not.

Theorem 3.10 ([3, Lemma 7])**.** Let $R, R' \subset V$ be construction problems such that $R \leq_{\mathrm{oW}} R'$. Suppose that R is such that there is an infinite ordinal α and an $x \in \mathrm{dom}(R)$ such that $|\mathrm{tc}(\{x\})| \leq \alpha$, but $(x, y) \in R$ implies that $|\mathrm{tc}(y)| > \alpha$. Then there is a $x' \in \mathrm{dom}(R')$ such that $|\mathrm{tc}(x')| \leq \alpha$, but $(x', y') \in R'$ implies $|\mathrm{tc}(y')| > \alpha$.

Proof. See [3, Lemma 7]. By Lemma 2.42, the programs P and Q witnessing this reduction can't output a larger cardinality then their input, so the R'-canonification must do so. □

Note that this theorem can be used to prove that given a code c_x, the power set of x is not computable. If power sets were computable, many forms of choice would be ordinal Weihrauch-equivalent to each other.

In a later chapter, we will discuss whether or not we can prove that two problems are not OTM-reducible. First, we move on to applications on choice.

3.4 Comparing forms of choice

We can now finally apply the reductions and theorems that we have discussed to compare effectivity of different statements. In this section we will focus on choice principles.

We first mention results in this setting that are already established before moving on to new contributions.

3.4.1 Some existing results

It is important that the construction problems that we consider have existing canonifications, but for this we assumed the Axiom of Global Choice. The existence of canonifications then follows from Lemma 2.11.

Definition 3.11. The following are relevant construction problems:

- Let PP be the **Picking Principle**: "For every non-empty x, there is a $y \in x$."
- Let $\mathsf{PP}_{\mathsf{fin}}$ be: "For every finite non-empty x, there is a $y \in x$."
- Let PP_2 be: "For every two-element set x, there is a $y \in x$."
- Let AC be the **Axiom of Choice**: "For every family of sets x, there is a choice function y on x."
- Let WO be the **Well-Ordering Theorem**: "For every set x, there is a $y = (x, \leq)$ that defines a well-order on x."
- Let HMP be the **Hausdorff Maximal Principle**: "For every partial order $x = (X, \leq)$, there is a maximal chain $y = T$ on $(X, \leq)$"

Note that there is no extra structure assumed on any of the sets: e.g. in the case of AC, an instance $x = \{s_i \mid i \in I\}$ is just a family of sets, not a well-ordered family of sets of the form $\{s_\gamma \mid \gamma \in \alpha\}$. This would not only be different construction problems in terms of OTM-effectivity, this would also be a strictly weaker axiom (over ZF) than the axiom of choice, see [12, Problem 4.14.].

The Hausdorff maximal principle is less well-known, but easily shown to be equivalent over ZF to Zorn's Lemma and thus a classical equivalent to AC. Zorn's Lemma itself is soW-equivalent to PP and not of interest to us, but we will consider Hausdorff maximal principle later.

Lemma 3.12. The following construction problems are always code-containing for models of ZF^-:

(i) PP_2, $\mathsf{PP}_{\mathsf{fin}}$, PP,

(ii) AC,

(iii) HMP.

Proof. (i) A construction problem R is always code-containing if for every $(x, y) \in R$ the existence in M of a well-ordering of $\mathrm{tc}(\{x\})$ implies the existence (in M) of a well-ordering of $\mathrm{tc}(\{y\})$. In the cases of PP_2, $\mathsf{PP}_{\mathsf{fin}}$ and PP, this holds because $y \in x$ and then $\mathrm{tc}(\{y\}) \subset \mathrm{tc}(\{x\})$.

(ii) In the case of AC, every y is a choice function on x, so $y \subset x \times \bigcup x$. The well-order of $\mathrm{tc}(\{x\})$ induces a well-order of $\bigcup x$ and then of $x \times \bigcup x$ too.

(iii) The final case is similar.

□

Construction problem WO is not always code-containing: let F be a canonification of WO. A code $c_x \in M$ for x comes with a well-ordering for $\mathrm{tc}(\{x\})$, but $F(x)$ may be a different well-ordering of x than the one in code c_x, and thus may not be in M.

Even though PP seems trivial, note that there is no general algorithm to pick an element from a set. Formally, we have this result:

Proposition 3.13. Construction problem PP is not OTM-effective.

Proof. Recall Example 3.6, where we showed that $\mathsf{PP} \equiv_{\mathrm{OTM}} \mathsf{WO}$. So, if PP is OTM-effective, then so would WO be. Also, a construction problem R is OTM-effective if and only if $R \leq_{\mathrm{oW}} 0$, (with 0 the empty construction problem), so we are going to use Theorem 3.9 to show that $\mathsf{WO} \not\leq_{\mathrm{oW}} 0$.

To satisfy the conditions of Theorem 3.9, note that there is a transitive model $M \vDash \mathrm{ZF}+\neg WO$, where there is a countable set $x \in M$ that can't be well-ordered in M [12, Theorem 5.20.]. Since x is countable, forcing notion $\mathbb{P}_x$ to well-order $\mathrm{tc}(\{x\})$ is countable, so it satisfies the first condition of Theorem 2.31. Then Theorem 3.9 gives $\mathsf{WO} \not\leq_{\mathrm{oW}} 0$, and so WO and PP are not effective. □

Proposition 3.14 ([3, Proposition 13. - Theorem 21.])**.** We have the following:

(i) $\mathsf{PP}_2 <_{\mathrm{soW}} \mathsf{PP}_{\mathsf{fin}} <_{\mathrm{soW}} \mathsf{PP} <_{\mathrm{soW}} \mathsf{AC} <_{\mathrm{soW}} \mathsf{WO}$,

(ii) $\mathsf{PP} <_{\mathrm{soW}} \mathsf{HMP} \leq_{\mathrm{soW}} \mathsf{WO}$,

and these are all not OTM-effective.

Proof. See [3, Proposition 13. - Theorem 21.]. The proofs there of $R \leq_{\mathrm{soW}} R'$ are from OTM-programs carrying out the usual proofs of the implication $R' \to R$ over ZF^-, and the proofs there of $R \not\leq_{\mathrm{oW}} R'$ are by Theorem 3.9.

We shortly note how this is possible. For example, the standard proof[1] of "Axiom of Choice implies the Well-Ordering Theorem" is a proof over ZFC, by well-ordering a set x via a choice function on $\mathcal{P}(x)$. But this power set cannot be computed by Lemma 2.42, and indeed Theorem 3.9 requires only a model of ZF^-. Such a model $M \vDash \mathrm{ZF}^- + AC + \neg WO$ exists by [21]. □

3.4.2 Classical equivalents of the Boolean Prime Ideal Theorem

We wish to consider the effectivity of the Boolean Prime Ideal Theorem (BPI), and some of its classical equivalents. Classically, BPI is strictly weaker than choice: it is a theorem in ZFC, but is independent of ZF [12, Theorems 7.1 & 7.8]. Many mathematical theorems can be proven with only ZF + BPI, without assuming AC [12, Sec. 2.3].

Theorems equivalent to the Boolean Prime Ideal Theorem are the Ultrafilter Theorem, and the Compactness Theorem (as in first-order logic). We will move to the Compactness Theorem later.

Definition 3.15. The following are construction problems:

Let BPI be the **Boolean Prime Ideal Theorem**: $(x, y) \in \mathsf{BPI}$ if and only if "if $x = (B, \leq)$ is a Boolean algebra, y is a prime ideal on x."

Let BPI' be: $(x, y) \in \mathsf{BPI}'$ iff "If $x = ((B, \leq), I)$ with $(B, \leq)$ a Boolean algebra and I an ideal on $(B, \leq)$, then y is a prime ideal on $(B, \leq)$ that extends I"

Let UF' be the **Ultrafilter Theorem**: $(x, y) \in \mathsf{UF}'$ iff "If $x = (X, F)$ with F a filter on X, then y is an ultrafilter on X extending F."

Let UF be: $(x, y) \in \mathsf{UF}$ iff "If $x = X$ is any set, then y is an ultrafilter on X."

Note that the commonly-used form of the Ultrafilter Theorem is denoted as UF' and not UF, in order to maintain some consistency between the four statements.

Proposition 3.16. We have $\mathsf{BPI} \equiv_{\mathrm{soW}} \mathsf{BPI}'$. As a corollary, $\mathsf{UF} \equiv_{\mathrm{soW}} \mathsf{UF}'$.

Proof. To show $\mathsf{BPI} \leq_{\mathrm{soW}} \mathsf{BPI}'$, suppose we are given a code for $x = (B, \leq)$. Define the program P to first find the least element 0. Since $\{0\}$ is an ideal in B, program P returns $((B, \leq), \{0\})$. Then, a canonification F of BPI' will return any prime ideal (all ideals extend $\{0\}$), and we let a program Q simply return this prime ideal.

[1] See e.g. [13, Theorem 5.1.]

For the other direction, we can compute the proof in [12, Section 2.3] of the notions being equivalent. Suppose that c_x codes Boolean algebra $(B, \leq)$ and ideal I on B. We define the equivalence relation

$$u \sim v \text{ iff } (u \wedge -v) \vee (v \wedge -u) \in I.$$

The binary operations $\wedge$ and $\vee$ are computable from $(B, \leq)$, so $u \sim v$ is computable by verifying if $(u \wedge -v) \vee (v \wedge -u) \in I$. Note that all the elements of I are in the same equivalence class. By Lemma 2.47, let the program P compute $B \times B$ and iterate through this list of pairs. For every $(u, v) \in B^2$ it determines whether $u \sim v$ and collects all those that are equivalent in $\sim$-equivalence classes.

By [12, Section 2.3], these classes are the domain of a new Boolean algebra B/I when ordered by $[u] \leq_I [v] \leftrightarrow u \leq v$, and they are independent of the given code.

Then, a canonification F of BPI will return a prime ideal K on B/I. This ideal K is a set of equivalence classes, and from this program Q computes, by Lemma 2.47, the union to return $J = \{u \in B : [u] \in K\}$, a prime ideal extending I.

The corollary follows from the observation that $(\mathcal{P}(X), \subseteq)$ is a Boolean algebra, and that prime ideals and ultrafilters are each others duals, see Definition 2.4. □

Proposition 3.17. We have $\mathsf{UF} \not\leq_{\mathrm{soW}} \mathsf{BPI}$.

Proof. For every X, an ultrafilter on X has cardinality $|\mathcal{P}(X)| > |X|$ but a prime ideal I on BA $(B, \leq)$ has cardinality $|B|$. Then the irreducibility follows from Theorem 3.10. □

Question 1. Is it the case that $\mathsf{BPI} <_{\mathrm{soW}} \mathsf{UF}$?

An existing proof of $\mathrm{ZF} \vdash UF \to BPI$ could, if computable, show a reduction $\mathsf{BPI} \leq_{\mathrm{soW}} \mathsf{UF}$. However, all of the proofs of $\mathrm{ZF} \vdash UF \to BPI$ we know have the Compactness Theorem as intermediate statement, and we will see that we do not know if BPI reduces to the Compactness Theorem.

The parallel product combines two construction problems into one.

Definition 3.18. Let $R, R' \in V \times V$. The **parallel product** of R and R', $\langle R, R' \rangle$, is defined as

$$\langle R, R' \rangle := \{((x_1, x_2), (y_1, y_2)) \in V^2 \mid (x_1, y_1) \in R \wedge (x_2, y_2) \in R'\}.$$

Different forms of the parallel product derive from computable analysis, see e.g. [1]. Note that the definition does not imply that there are two `Miracle` tapes, but an instance of $\langle R, R' \rangle$ is a pair (x_1, x_2) with x_1 an R-instance and x_2 a R'-instance on the single `Miracle` tape.

Proposition 3.19. Let $\mathsf{Pow} \subset V^2$ be the construction problem of power sets, i.e. $(x, y) \in \mathsf{Pow}$ iff $y = \mathcal{P}(x)$. Then $\mathsf{Pow} \leq_{\mathrm{soW}} \mathsf{UF} \leq_{\mathrm{soW}} \langle \mathsf{Pow}, \mathsf{PP} \rangle$.

Proof. For the first reduction, let a program P simply return input X so that the UF-canonification F returns an ultrafilter U on X. By definition of ultrafilter, we have $\forall S \in \mathcal{P}(X)(S \in U \vee S^c \in U)$. To compute S^c, the complement of S in X, first set X must be computed. Since $X = \bigcup U$, this is computable by Lemma 2.47. Then program Q may iterate through all $S \in U$, computing S^c and collecting all sets S and S^c to obtain $\mathcal{P}(X)$.

The second reduction follows from the construction of a principal ultrafilter: for any $x \in X$, we have that $\{S \in \mathcal{P}(X) \mid x \in S\}$ is an ultrafilter. For a program P to return a $\langle \mathsf{Pow}, \mathsf{PP} \rangle$-instance, let it take input X and return (X, X). Then, $\langle \mathsf{Pow}, \mathsf{PP} \rangle$-canonification F returns a code for $F((X, X)) = (\mathcal{P}(X), x)$ with $x \in X$. Then, a program Q can compute $U = \{S \in \mathcal{P}(X) \mid x \in S\}$ by Lemma 2.47. □

Proposition 3.20. We have $\mathsf{PP}_{\mathsf{fin}} <_{\mathrm{soW}} \mathsf{UF}$ and $\mathsf{PP}_{\mathsf{fin}} \leq_{\mathrm{soW}} \mathsf{BPI}$.

Proof. From Theorem 3.10 we can immediately derive $\mathsf{UF} \not\leq_{\mathrm{soW}} \mathsf{PP}_{\mathsf{fin}}$.

To show that $\mathsf{PP}_{\mathsf{fin}} \leq_{\mathrm{soW}} \mathsf{UF}$, let c_x be a code for a finite set x. We use the fact that for a finite set every ultrafilter is a principal filter. So if a program P simply returns x, the UF-canonification returns a principal ultrafilter U on X.

Then program Q gets a code for an ultrafilter U of the form $\{S \in \mathcal{P}(x) \mid y \in S\}$. Now there is no other singleton than $\{y\}$ in U, so a program Q may iterate through the elements of U, and return the first singleton it finds.

To show $\mathsf{PP}_{\mathsf{fin}} \leq_{\mathrm{soW}} \mathsf{BPI}$, the same proof works, since for a finite set X, power set $\mathcal{P}(X) = \mathcal{P}^{<\omega}(X)$ and is thus computable by Lemma 2.47. Then Boolean algebra $(\mathcal{P}(X), \supseteq)$ is also computable. Any BPI-canonification then returns a prime ideal on $(\mathcal{P}(X), \supseteq)$, which is also a principal ultrafilter on X, so the same program Q as above returns $y \in x$. □

Proposition 3.21. We have $\mathsf{BPI} \leq_{\mathrm{soW}} \mathsf{HMP}$.

Proof. Let c_x be a code for Boolean algebra $(B, \leq)$. Since this is also a partial order, program P just returns $(B, \leq)$. Then HMP-canonification F returns a maximal totally ordered set $T \subset B$.

Since $1 \geq u$ for all $u \in B$, by maximality $1 \in T$. Let $T' = T \setminus \{1\}$. Then a program Q can iterate through B and compute $T'{\downarrow} = \{u \in B \mid \exists t \in T'(u \leq t)\}$, by Lemma 2.47. We claim that $T'{\downarrow}$ is a prime ideal, so that the following hold.

(i) If $a \leq u$ and $u \in T'{\downarrow}$, then $a \in T'{\downarrow}$,

(ii) If $u, v \in T'{\downarrow}$, $u \vee v \in T'$,

(iii) For every $a \in B$, either $a \in T'{\downarrow}$ or $\neg a \in T'{\downarrow}$,

(iv) $1 \notin T'{\downarrow}$.

Then Q returns prime ideal $T'{\downarrow}$, showing that $\mathsf{BPI} \leq_{\mathrm{oW}} \mathsf{HMP}$. We show that the conditions hold:

(i) By definition of downwards closure.

(ii) Let $t, t' \in T'$ be such that $u \leq t$ and $v \leq t'$. T is totally ordered, so assume w.l.o.g. that $t \leq t'$. Then also $u \leq t'$, so t' is above both u and v, and so $u \vee v \leq t'$, giving $u \vee v \in T'{\downarrow}$.

(iii) For all $t \in T'$, we have $t \vee a \geq t \leq t \vee \neg a$. Therefore, by maximality of T, for a fixed $t \in T'$, $t \vee a, t \vee \neg a \in T$. Suppose that $t \vee a = 1 = t \vee \neg a$. Then $t = t \vee 0 = t \vee (a \wedge \neg a) = (t \vee a) \wedge (t \vee \neg a) = 1 \wedge 1 = 1$. But this contradicts $t \in T'$. So either $t \vee a < 1$ and then $t \vee a \in T'$ and so $a \in T'{\downarrow}$, or $t \vee \neg a$ holds, and then $\neg a \in T'{\downarrow}$.

(iv) 1 is not an element of T', so $1 \notin T'{\downarrow}$.

□

We wish to also consider the Compactness Theorem of first-order logic, but will only do so in Chapter 5. We first need the main result of the next chapter.

Chapter 4

OTM-irreducibility

The previous chapter discussed theorems that can show ordinal Weihrauch irreducibility, namely Theorems 3.9 and 3.10. We do not have any such general theorem that may show OTM-irreducibility. However, we will prove one case of OTM-irreducibility, but first we look more closely at OTM-reductions.

A program that witnesses $R \leq_{\mathrm{OTM}} R'$ may use the miracle command an arbitrary number of times. Since this can be used in many different ways, showing that an OTM-reduction does not exist is not straightforward. As a small example of this, we give a counterexample to an incorrect result by Carl.

Lemma 4.1. The following, **Lemma 7.** in [3], is *false*:

Let $R, R' \subset V^2$ be construction problems such that $R \leq_{\mathrm{OTM}} R'$ and α an infinite ordinal. Suppose that there is an $x \in \mathrm{dom}(R)$ such that $|\mathrm{tc}(\{x\})| \geq \alpha$, and if $(x, y) \in R$, then $|\mathrm{tc}(\{y\})| > \alpha$. Then there is also an $x' \in \mathrm{dom}(R')$ such that $|\mathrm{tc}(\{x'\})| \geq \alpha$, and if $(x', y') \in R'$, then $|\mathrm{tc}(\{y'\})| > \alpha$.

Proof. Let $R' = \{(\alpha, 1) \mid \alpha \in \mathrm{Ord}, |\alpha| \leq \aleph_0\} \cup \{(\alpha, 0) \mid \alpha \in \mathrm{Ord}, |\alpha| > \aleph_0\}$ and $R = \{(\omega, \omega_1)\}$.

Now with $\alpha = \omega$, R satisfies the condition of the lemma, yet R' does not satisfy the consequent of the lemma. So, if $R \leq_{\mathrm{OTM}} R'$, we have shown that the lemma is false.

Construction problems R and R' suggest how we can do this: a program P^F witnessing $R \leq_{\mathrm{OTM}} R'$ must construct a code for ω_1, with a code for ω being given. Now it may not immediately construct any output of size $|\omega_1|$, since halting OTM-programs (without miracle command) cannot raise cardinalities. But a program P can easily write larger cardinalities than α on the tape if it loops, every time computing α's successor. Then P^F may witness $R \leq_{\mathrm{OTM}} R'$ by using the R'-canonification F to halt, precisely when $F(\alpha) = 0$. The first time this happens, by definition of R' is when $\alpha = \omega_1$, so this computes the only R-solution. □

In [3], the proof of this lemma is said to follow from [4, Lemma 1.]. We think it depends on some unstated extra condition on R or R'.

4.1 OTM-reductions with a bound on the oracle

Instead of diving into OTM-reductions, we bound the number of oracle calls first.

Definition 4.2. Let α be an ordinal and F some canonification. An **α-OTM-program** P^F is a miracle OTM-program where the miracle command executes F and for every input c_x, the uses of the miracle command must be of order type less than or equal to α during the computation $P^F(c_x)$. We write $R \leq_{\alpha\text{-OTM}} R'$ when there is a α-OTM-program P such that for every R'-canonification F, P^F witnesses the OTM-reduction from R to R'.

When the canonification F is not yet specified, we may still write P^F for these programs, to show that the OTM does actually have access to an oracle. We do not know if the definition can be done with cardinals instead of ordinals:

Question 2. We do not know if $\leq_{\alpha\text{-OTM}}$ and $\leq_{\beta\text{-OTM}}$ are identical when $|\alpha| = |\beta|$.

As an illustrating example of the workings of miracle OTMs, we will examine 1-OTMs and 1-OTM-reductions more closely. Suppose that $R, R' \subseteq V^2$, let F be an R'-canonification and suppose there exists a 1-OTM-program P^F witnessing $R \leq_{1\text{-OTM}} R'$. Let x be an instance of R. Then for any code c_x for x, $P^F(c_x)$ will output a code for a z such that $(x, z) \in R$. During the run of $P^F(c_x)$, the miracle command is used either 0 times or 1 time. If it is used 0 times, then naturally c_z is computable in the input c_x.

If the miracle command is used on some code c_y, a code for $y \in \mathrm{dom}(R')$, we are in a situation just like we are with a Weihrauch reduction. The key difference is that for two different codes for x, say we have codes c_x and c'_x, the oracle may be called on different R'-instances, say y and y'.

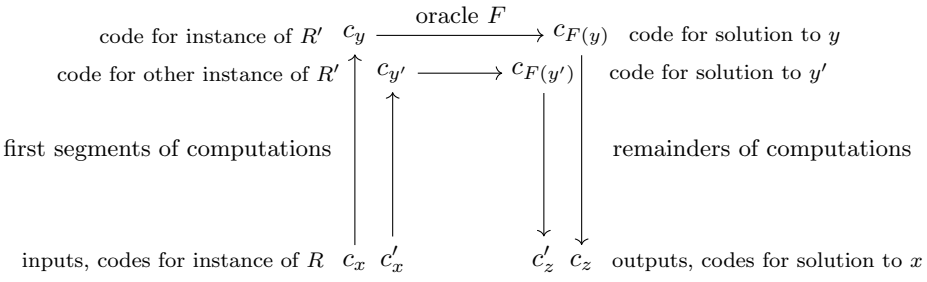


Figure 4.1: Schematic of two runs of a P^F witnessing $R \leq_{1\text{-OTM}} R'$.

In a Weihrauch reduction $R \leq_{\mathrm{soW}} R'$ there is not one program involved, but two: P and Q. The set y that the oracle is called on is the output of P, and P is code-independent, such that the same y is coded by its outputs $P(c_x)$ and $P(c_y)$.

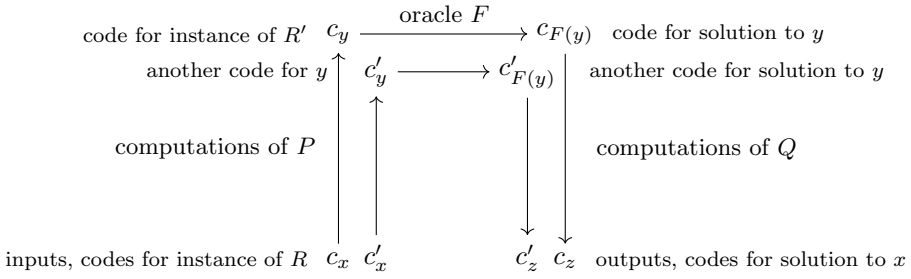


Figure 4.2: Schematic of two runs of a P and Q witnessing $R \leq_{\mathrm{soW}} R'$.

As a metaphor, we could say that in the case of a Weihrauch reduction, the oracle has been kept by the OTM's operator. When program P has computed a code c_y, the operator then inputs this into the oracle, and inputs the oracle's response to program Q. In the case of the 1-OTM-reduction, the OTM-program is given the oracle button and can apply it once, but however it wishes, with the operator only caring about whether the final output is independent of the codes.

The 1-OTM-computation is clearly more similar to a full miracle OTM-computation, and we will study it further. The following shows that these reductions will, to some extent, still call the oracle on identical sets.

Theorem 4.3. Let $R, R' \subset V^2$ be construction problems that are both not effectively computable. If P^F is an OTM-program witnessing $R \leq_{\text{1-OTM}} R'$ and R' is such that there is an OTM-program Q such that for every $y \in \mathrm{dom}(R')$ and for every code c_y for y, $Q(c_y)$ outputs a code c_w such that $(y, w) \in R'$, then there is an $x \in \mathrm{dom}(R)$ such that for every code c_x for x, when P^F is run on c_x, it calls the oracle on the same $y \in \mathrm{dom}(R')$.

Remark: Here Q is code-dependent in the sense that it could be that $\mathit{decode}(Q(c_y)) \neq \mathit{decode}(Q(c'_y))$ for two different codes c_y and c'_y. Otherwise Q witnesses that R' is effectively computable. If, e.g., $R' \in \{\mathsf{PP}_{\mathsf{fin}}, \mathsf{PP}, \mathsf{AC}, \mathsf{WO}\}$ there is such a Q: for instance, if $R' = \mathsf{PP}$, program Q can use c_y to return the first $c_w \in c_y$ as induced by the enumeration of code c_y.

Proof. Suppose for contradiction that there is no such x. Let P^* be like P^F, but when the miracle command is used on code c_y, instead run $Q(c_y)$, placing its response on the `Miracle` tape as if it were F's response.

Claim: P^* effectively computes R.

Let $x \in \mathrm{dom}(R)$ be arbitrary, and c_x and c'_x be two codes for x. We have to show that $P^*(c_x) = P^*(c'_x)$ and that this is an R-solution to x.

In these two runs, define c_{y_x} and $c_{y'_x}$ as the codes for y_x and y'_x respectively that are on the `Miracle` tape when P^* starts to run Q. There are two cases:

First, suppose that $y_x \neq y'_x$. Then we let Q induce part of an R' canonification F as follows: by assumption $(y_x, \mathit{decode} \circ Q(c_{y_x})) \in R'$ and $(y'_x, \mathit{decode} \circ Q(c_{y'_x})) \in R'$, and since $y_x \neq y'_x$, we may extend this to a full R'-canonification F. Now since P^F was a witness to $R \leq_{\text{1-OTM}} R'$, it must be that for this F, $\mathit{decode} \circ P^F(c_x) = \mathit{decode} \circ P^F(c'_x)$ is an R-solution for x. And since Q agrees with F on y_x and y'_x, we have that $\mathit{decode} \circ P^*(c_x) = \mathit{decode} \circ P^*(c'_x)$ is the same R-solution to x.

Now suppose that $y_x = y'_x$. By assumption, there is another code c''_x such that, defining $c_{y''_x}$ and y''_x analogously, we have $y_x \neq y''_x \neq y'_x$.

We apply the argument from the previous case twice: since there is an R'-canonification that extends $\{(y_x, \mathit{decode} \circ Q(c_{y_x})), (y''_x, \mathit{decode} \circ Q(c_{y''_x}))\} \subset R'$, we get that $\mathit{decode} \circ P^*(c_x) = \mathit{decode} \circ P^*(c''_x)$, and another canonification extends $\{(y'_x, \mathit{decode} \circ Q(c_{y'_x})), (y''_x, \mathit{decode} \circ Q(c_{y''_x}))\} \subset R'$, so $\mathit{decode} \circ P^*(c'_x) = \mathit{decode} \circ P^*(c''_x)$. Thus, $\mathit{decode} \circ P^*(c_x) = \mathit{decode} \circ P^*(c'_x)$ is an R-solution to x.

Now we have proven the claim, contradicting the assumption that R is not effectively computable. Then there must have been an $x \in \mathrm{dom}(R)$ for which no matter the code c_x, when P^F is run on c_x, it will call the oracle on the same set y. □

Theorem 4.4. There exist two construction problems $R, R' \subseteq V^2$ such that $R \leq_{\text{1-OTM}} R'$ but $R \not\leq_{\text{oW}} R'$.

Proof. Recall that in Lemma 3.8 we gave an example of an inherently strong Weihrauch reduction, by coding an initial input into the instance of R' in such a way that it may be recovered from the R'-solution. We can easily choose a construction problem where this is impossible.

We define $(x, y) \in R$ iff $y = (x, \omega_1)$ and $(x, y) \in R'$ iff $x = 0 \wedge y = \omega_1$.

Then $R \leq_{\text{1-OTM}} R'$ by a program P^F taking the input x, calling the oracle with 0 on the `Miracle` tape to obtain ω_1, and pairing this with x to return (x, ω_1), the R-solution to x.

However, suppose that there are programs P and Q witnessing $R \leq_{\text{oW}} R'$. Let c_x be a countable input. If $P(c_x) \neq 0$, then the R'-canonification does not return ω_1 and so program Q has a countable input. But then it is impossible that Q returns (x, ω_1) since by Lemma 2.42, OTM-programs cannot return outputs larger than their input.

Thus $P(c_x) = 0$. But then Q has the same input, a code for $(0, \omega_1)$, no matter what the original countable instance x was. For Q to witness the oW-reduction, it has to return (x, ω_1), yet Q is code-independent and thus returns a certain code c_z for some fixed z instead. We conclude that $R \not\leq_{\text{oW}} R'$.

□

It is obvious that $\leq_{\alpha\text{-OTM}}$ is reflexive for any $\alpha > 0$, and the following shows that we can speak of 1-OTM equivalence classes.

Proposition 4.5 (Transitivity of $\leq_{1\text{-OTM}}$)**.** Suppose $R, R', R'' \subseteq V^2$ are such that $R \leq_{\alpha\text{-OTM}} R'$ and $R' \leq_{\beta\text{-OTM}} R''$. Then $R \leq_{\beta\cdot\alpha\text{-OTM}} R''$. In particular, if $\alpha = \beta = 1$, we have $R \leq_{1\text{-OTM}} R''$.

Proof. Let programs $P^{F'}$ and $P'^{F''}$ witness $R \leq_{\alpha\text{-OTM}} R'$ and $R' \leq_{\beta\text{-OTM}} R''$ respectively. To show $R \leq_{\alpha\cdot\beta\text{-OTM}} R''$, let F'' be an R''-canonification. We define program $Q^{F''}$ as program P, but whenever the oracle is called on some c_y, $Q^{F''}$ instead computes $P'(c_y)$. When during this inner run the miracle command is used Q does just that, replacing input c_w with some $c_{F''(w)}$. By β-OTM-reduction, $P'(c_y)$ halts, calling the oracle at most β-often, and outputs a code c_z such that $(c_y, c_z) \in R'$. Then the outer run of P can continue, with c_z placed on the `Miracle` tape as solution to c_y.

Suppose c_y and c'_y are codes for the same set y. Then by β-OTM-reduction, $c_z := P'(c_y)$ and $c'_z := P'(c'_y)$ will both be codes for the same set z. The $(c_y, c_z) \in R'$ as computed by $P'^{F''}$ can by Lemma 2.11 be extended to a full R'-canonification and so, by α-OTM-reduction, Q halts and is code-independent. By construction, Q runs the inner program P' at most α-often. Since the oracle F'' is called at most β-often for all of the runs of P', Q calls F'' at most $\beta \cdot \alpha$-often. □

We will not research many different α-OTM-reductions. We will discuss only one 1-OTM-reduction to better understand general OTM-reductions.

4.2 $\mathsf{PP}_{\mathsf{fin}}$ versus PP

Recall Definition 3.11: PP refers to picking an element from any set, $\mathsf{PP}_{\mathsf{fin}}$ to picking an element from any finite set.

Here we wish to compare the two in terms of OTM-reducibility. We do not yet have a single theorem that can show $R \not\leq_{\text{OTM}} R'$, however, the known proof for $\mathsf{PP}_{\mathsf{fin}} \not\leq_{\text{oW}} \mathsf{PP}$ also does not any of the general theorems that may be used for showing $R \not\leq_{\text{oW}} R'$. Thus, we adapt this more specific approach.

Theorem 4.6. We have $\mathsf{PP} \not\leq_{1\text{-OTM}} \mathsf{PP}_{\mathsf{fin}}$.

Proof. This is an adaptation of Theorem 16 in [3]. We write AC_{fin} for the axiom of choice for a family of finite sets.

By [12, Theorem 5.21], there is a cardinal-preserving extension G and model M with $L \subseteq M \subseteq L[G]$, such that $M \vDash \text{ZF} + LO + \neg AC$, where LO denotes that any set has a linear order. This implies AC_{fin} and so $M \vDash \text{ZF} + AC_{fin} + \neg AC$. Let $X \in M$ be without a choice function, where by construction we can take $X \in L_{\omega_5^L}[G]$, guaranteeing the existence of the generic filters below by Theorem 2.31.

Suppose for contradiction that P^F is a program witnessing $\mathsf{PP} \leq_{1\text{-OTM}} \mathsf{PP}_{\mathsf{fin}}$. We do not yet fix F, but will do so later. Let $\mathbb{P}_X$ be the forcing notion to well-order $\mathrm{tc}(\{X\})$. There are two mutually generic filters G_1 and G_2 on $\mathbb{P}_X$, and these give rise to codes $c_X \in M[G_1]$ and $c'_X \in M[G_2]$ for X. For every $x \in X$, we write c_x and c'_x for the codes for x as induced by c_X and c'_X respectively.[1]

When P^F is executed on c_x, let c_{y_x} be the code for set y_x that is on the `Miracle` tape when the oracle is called during the run, and have c'_{y_x} and y'_x be analogous. If the oracle is not called or if $y_x^{(\prime)}$ is not finite, let $y_x^{(\prime)} = \emptyset$.[2]

Collect these oracle calls in $Y = \{y_x \mid c_x \in c_X\}$ and $Y' = \{y'_x \mid c'_x \in c'_X\}$. Note that the oracle has not yet been applied, so Y and Y' are independent of the canonification F. Since $M[G_1]$ and $M[G_2]$ are closed under OTM-computations by Replacement, $Y \in M[G_1]$ and $Y' \in M[G_2]$.[3] From Lemma 2.35 it follows that $Y \cap Y' \subset M$ (but not $Y \cap Y' \in M$!). Then by how the mappings $c_x \mapsto y_x$ and $c'_x \mapsto y'_x$ are defined, we have

$$\begin{aligned} Y \cap Y' \;=\; \{y \in M \mid \exists c_x \in c_X, c'_{x'} \in c'_X : \\ P^F(c_x) \text{ and } P^F(c'_{x'}) \text{ both call the oracle on } y\}. \end{aligned}$$

Let, for every c_{y_x}, $w_x \in y_x$ be the least element w.r.t. the enumeration in the coding c_{y_x}. Then, the image of c_X under $c_x \mapsto (y_x, w_x)$ is a choice function f_1 on Y and by Replacement, $f_1 \in M[G_1]$. Similarly, there is a choice function $f_2 \in M[G_2]$ on Y'.[4]

Since we have $Y \cap Y' \subset M$, let $C \in M$ be a set of finite sets such that $C \supset Y \cap Y'$, for instance $C = \mathcal{P}^{<\omega}(V_\alpha^M)$ for a suitably large α. This is possible since $Y \cap Y'$ cannot be cofinal in V^M, since $|X| \geq |Y|$ and $|X| \geq |Y'|$. By AC_{fin} in M, there is a choice function f_C on C. Since $M[G_1], M[G_2] \supseteq M$, we have $C \in M[G_1] \cap M[G_2]$ and $f_C \in M[G_1] \cap M[G_2]$.

Now since $Y \setminus C \in M[G_1]$ and $Y' \setminus C \in M[G_2]$ we may combine f_C, f_1 and f_2 to get the following choice functions on Y and Y' respectively:

$$\begin{aligned} F_1 \;&=\; [f_C \restriction (Y \cap C)] \cup [f_1 \restriction (Y \setminus C)] \in M[G_1] \\ F_2 \;&=\; [f_C \restriction (Y' \cap C)] \cup [f_2 \restriction (Y' \setminus C)] \in M[G_2]. \end{aligned}$$

Crucially, these two agree on their common domain, since that is exactly $Y \cap Y'$ and handled by f_C. Therefore, by Lemma 2.11 let F be a complete canonification of $\mathsf{PP}_{\mathsf{fin}}$ extending $F_1 \cup F_2$ so that $F(y_x^{(\prime)}) \in y_x^{(\prime)}$ for all elements of $Y \cup Y'$. For this F, since P^F is a program witnessing $\mathsf{PP} \leq_{1\text{-OTM}} \mathsf{PP}_{\mathsf{fin}}$, we have $\mathit{decode}(P^F(c_x)) = \mathit{decode}(P^F(c'_x)) \in x$ for every $x \in X$.

When P^F is run on all the $c_x \in c_X$, the used part of the oracle F is just $F_1 \in M[G_1]$, and since $M[G_1]$ is closed under OTM-computation the full

[1] For every $x \in X$, code $c_X = \langle \alpha, C_d \rangle$ induces code $c_x = \langle \alpha_x, C_d \rangle$ where α_x is such that $d(\alpha_x) = x$.

[2] We use $y_x^{(\prime)}$ as a shorthand for speaking about y_x and y'_x

[3] To be precise, Y is the image of c_X under $c_x \mapsto \mathit{decode}(c_{y_x}) = y_x$

[4] Although $M \vDash AC_{fin}$, we can't find these choice functions by appealing to AC_{fin}, since it is not generally true that extensions $M[G_1], M[G_2] \vDash AC_{fin}$. See [18, Theorem 3.4. & Corollary 4.9].

runs of P^F are defined in $M[G_1]$. Then so is map $c_x \mapsto \big(x, \mathit{decode}(P^F(c_x))\big)$ and using Replacement, $Z = \{(x, P^F(c_x)) \mid c_x \in c_X\} \in M[G_1]$. Analogously, $Z' = \{\big(x, \mathit{decode}(P^F(c'_x))\big) \mid c'_x \in c'_X\} \in M[G_2]$. We had established $\mathit{decode}(P^F(c_x)) = \mathit{decode}(P^F(c'_x))$, hence $Z = Z'$. But then, by Lemma 2.35, $Z \in M[G_1] \cap M[G_2] = M$. Now Z is a choice function on X, while in M there was no such function. We conclude that $\mathsf{PP} \not\leq_{1\text{-OTM}} \mathsf{PP}_{\mathsf{fin}}$. □

We can adapt the construction from the theorem above to obtain a stronger result.

Theorem 4.7. We have $\mathsf{PP} \not\leq_{\mathrm{OTM}} \mathsf{PP}_{\mathsf{fin}}$.

Proof. Again, we write AC_{fin} for the axiom of choice for families of finite sets.

By [12, Theorem 5.21], there is a cardinal-preserving extension G and model M with $L \subseteq M \subseteq L[G]$, such that $M \vDash \mathrm{ZF} + LO + \neg AC$, where LO denotes that any set has a linear order. This implies AC_{fin} and so $M \vDash \mathrm{ZF} + AC_{fin} + \neg AC$. Let $X \in M$ be without a choice function, where by construction we can take $X \in L_{\omega_5^L}[G]$, guaranteeing the existence of the generic filters below by Theorem 2.31.

Suppose for contradiction that there is a P^F witnessing $\mathsf{PP} \not\leq_{\mathrm{OTM}} \mathsf{PP}_{\mathsf{fin}}$. Let $\mathbb{P}_X$ be the forcing notion to well-order $\mathrm{tc}(\{X\})$. There are two mutually generic filters G_1 and G_2 on $\mathbb{P}_X$, and these give rise to codes $c_X \in M[G_1]$ and $c'_X \in M[G_2]$ for X. For every $x \in X$, we write c_x and c'_x for the codes for x as induced by c_X and c'_X respectively.

Let $c_{y_{\gamma,x}}$ and $c'_{y_{\gamma,x}}$ be the codes for $y_{\gamma,x}$ and $y'_{\gamma,x}$ that are on the `Miracle` tape when the oracle is called for the γ-th time, letting $y^{(')}_{\gamma,x} = \{\emptyset\}$ if the oracle is not called. And define $Y_\gamma = \{y_{\gamma,x} \mid c_x \in c_X\}$ and $Y_\gamma = \{y_{\gamma,x} \mid c_x \in c_X\}$.

Now Y_1 and Y'_1, are independent of the canonification F, but the others may depend on it. We build F step-by-step, such that in the end all of the mentioned computations go through as claimed.

Claim: For all $\alpha \in \mathrm{Ord}$, there are choice functions $F_\alpha \in M[G_1]$ and $F'_\alpha \in M[G_2]$ on $\bigcup_{\gamma\leq\alpha} Y_\gamma$ and $\bigcup_{\gamma\leq\alpha} Y_\gamma$ respectively, a set of finite sets $C_\alpha \in M$ such that $\mathrm{dom}(F_\alpha) \cap \mathrm{dom}(F'_\alpha) \subseteq C_\alpha$, a choice function f_{C_α} on C_α and F_α, F'_α and f_{C_α} all agree.

We prove this by induction on α. When $\alpha = 0$ this is is trivial since the oracle has not been called.

Successor: Suppose that the IH holds for some ordinal α, and thus we have choice functions $F_\alpha \in M[G_1]$ and $F'_\alpha \in M[G_2]$ on $\bigcup_{\gamma\leq\alpha} Y_\gamma$ and $\bigcup_{\gamma\leq\alpha} Y_\gamma$ respectively, a choice function $f_{C_\alpha} \in M$ on $C_\alpha \supseteq \mathrm{dom}(F_\alpha) \cap \mathrm{dom}(F'_\alpha)$ and F_α, F'_α and f_{C_α} agree.

Maps $c_x \mapsto c_{y_{\alpha+1,x}}$ and $c'_x \mapsto c'_{y_{\alpha+1,x}}$ are defined by program P^F, parameter α and choice functions F_α and F'_α and so $Y_{\alpha+1} \in M[G_1]$ and $Y'_{\alpha+1} \in M[G_2]$. These are mutually generic extensions, so by Lemma 2.35, $Y_{\alpha+1} \cap Y'_{\alpha+1} \subset M$.

The same reasoning as in Theorem 4.6 shows that there is a set of finite sets $C_{\alpha+1} \in M$ such that $C_{\alpha+1} \supset (Y_{\alpha+1} \cap M) \cup (Y_{\alpha+1} \cap M) \cup \mathrm{dom}(f_{C_\alpha})$ and then

by AC_{fin} in M, there is a choice function $f_{C_{\alpha+1}} \in M$ on $C_{\alpha+1}$ that extends f_{C_α}. Also, the codings $c^{(\prime)}_{y_{\alpha+1},x}$ give rise to choice functions $f_{\alpha+1} \in M[G_1]$ and $f'_{\alpha+1} \in M[G_2]$ on Y_{k+1} and Y'_{k+1} respectively.

Let us then build the following choice functions on $\bigcup_{i\leq\alpha+1} Y_i$ and $\bigcup_{i\leq\alpha+1} Y_i$:

$$\begin{aligned} F_{\alpha+1} &= F_\alpha \cup [f_{C_{\alpha+1}} \restriction Y_{\alpha+1}] \cup [f_{\alpha+1} \restriction Y_{\alpha+1} \setminus (C_{\alpha+1} \cup \mathrm{dom}(F_\alpha))] \in M[G_1], \\ F'_{\alpha+1} &= F'_\alpha \cup [f_{C_{\alpha+1}} \restriction Y'_{\alpha+1}] \cup [f'_{\alpha+1} \restriction Y'_{\alpha+1} \setminus (C_{\alpha+1} \cup \mathrm{dom}(F'_\alpha))] \in M[G_2]. \end{aligned}$$

First to check that these define functions: the domains of $f_{C_{\alpha+1}}$ and $F^{(\prime)}_\alpha$ may overlap, but only inside of M. By our IH, $F^{(\prime)}_\alpha \restriction M$ agrees with f_{C_α}. thus $f_{C_{\alpha+1}}$ and $F^{(\prime)}_\alpha$ agree.

By construction, $F_{\alpha+1}$ and $f_{C_{\alpha+1}}$ agree and $F'_{\alpha+1}$ and $f_{C_{\alpha+1}}$ agree, and $\mathrm{dom}(F_{\alpha+1}) \cap \mathrm{dom}(F'_{\alpha+1}) \subseteq \mathrm{dom}(f_{C_{\alpha+1}})$. Thus $F_{\alpha+1}$, $F'_{\alpha+1}$ and f_{c_α} agree and we have proven the IH for $k+1$.

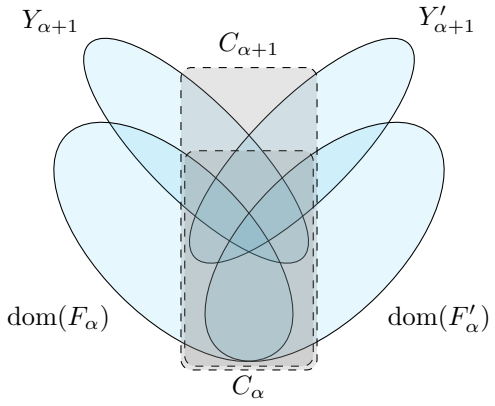


Figure 4.3: Representation of the successor step

Limit: Suppose that for all $\alpha < \lambda$ we have choice functions $F_\alpha \in M[G_1]$ and $F'_\alpha \in M[G_2]$ on $\bigcup_{\gamma\leq\alpha} Y_\gamma$ and $\bigcup_{\gamma\leq\alpha} Y_\gamma$ respectively, and a choice function $f_{C_\alpha} \in M$ on $C_\alpha \supseteq \mathrm{dom}(F_\alpha) \cap \mathrm{dom}(F'_\alpha)$ and F_α, F'_α and f_{C_α} agree.

Let $H_\lambda = \bigcup_{\alpha<\lambda} F_\alpha \in M[G_1]$ and $H'_\lambda = \bigcup_{\alpha<\lambda} F'_\alpha \in M[G_2]$ be choice functions on $\bigcup_{\gamma<\lambda} Y_\gamma$ and $\bigcup_{\gamma<\lambda} Y'_\gamma$ and let $h_\lambda = \bigcup_{\alpha<\lambda} f_{C_\gamma} \in M$ be a choice function on $C \supseteq \mathrm{dom}(H_\lambda) \cap \mathrm{dom}(H'_\lambda)$.

Now the maps $c_x \mapsto c_{y_{\lambda,x}}$ and $c'_x \mapsto c'_{y_{\lambda,x}}$ are defined by program P^F, parameter λ and choice functions H_λ and H'_λ and so again $Y_\lambda \in M[G_1]$ and $Y'_\lambda \in M[G_2]$. The same construction as from the successor case will now work, completing our induction.

There must be an $\alpha \in \mathrm{Ord}$ where the oracle is not called at all on any of the runs so that $Y_\alpha = \emptyset = Y'_\alpha$, since if there is not such an α, we can have a $\mathsf{PP}_{\mathsf{fin}}$-canonification F extending $\bigcup_{\alpha\in\mathrm{Ord}}(F_\alpha \cup F'_\alpha)$ for which there is a code

$c_x^{(\prime)}$ where $P^F(c_x^{(\prime)})$ calls the oracle |Ord|-many times, so it never halts. This is of course contradicting the fact that P^F must halt for all codes c_x and all $\mathsf{PP}_{\mathsf{fin}}$-canonifications F.

By Lemma 2.11, let F be a $\mathsf{PP}_{\mathsf{fin}}$-canonification that extends $F_\alpha \cup F'_\alpha$. Then by construction the full runs of $P^F(c_x^{(\prime)})$ indeed call the oracle on the $(y_{\gamma,x}^{(\prime)})_{\gamma<\alpha}$, and so since P^F was a program witnessing $\mathsf{PP} \leq_{\mathrm{OTM}} \mathsf{PP}_{\mathsf{fin}}$ we get $\mathit{decode}(P^F(c_x)) = \mathit{decode}(P^F(c'_x)) \in x$.

When P^F is run on all the $c_x \in c_X$, the used part of the oracle F is just $F_\alpha \in M[G_1]$, and since $M[G_1]$ is closed under OTM-computation the full runs of P^F are defined in $M[G_1]$. Then so is map $c_x \mapsto \big(x, \mathit{decode}(P^F(c_x))\big)$ and using Replacement, $Z = \{(x, P^F(c_x)) \mid c_x \in c_X\} \in M[G_1]$. Analogously, $Z' = \{\big(x, \mathit{decode}(P^F(c'_x))\big) \mid c'_x \in c'_X\} \in M[G_2]$. We had established $\mathit{decode}(P^F(c_x)) = \mathit{decode}(P^F(c'_x))$, hence $Z = Z'$. But then, by Lemma 2.35, $Z \in M[G_1] \cap M[G_2] = M$. Now Z is a choice function on X, while in M there was no such function. We conclude that $\mathsf{PP} \not\leq_{\mathrm{OTM}} \mathsf{PP}_{\mathsf{fin}}$. □

4.2.1 OTM-degrees

In Example 3.6 we mentioned the fact that $\mathsf{PP} \equiv_{\mathrm{OTM}} \mathsf{WO}$. The intuition behind this result follows from two observations: an OTM-program with a miracle command can use this oracle very often, and a PP-canonification is an global choice function, implying many other forms of choice.

Yet, Theorem 4.7 shows the first limit to what an OTM-program with a miracle command may achieve. We have mentioned how $\equiv_{\mathrm{OTM}}$ is an equivalence relation, meaning that we may think of the OTM-computability landscape as different OTM-degrees: the equivalence class of construction problems that are OTM-equivalent.

Definition 4.8. Let $R \subseteq V^2$ be a construction problem. We write $[R]_{\mathrm{OTM}}$ for the **OTM-degree of** R: the class of all construction problems $R' \subseteq V^2$ such that $R \equiv_{\mathrm{OTM}} R'$. Let $\leq$ be the partial order on OTM-degrees defined by $[R]_{\mathrm{OTM}} \leq [R']_{\mathrm{OTM}}$ if and only if $R \leq_{\mathrm{OTM}} R'$.

Corollary 4.9. Let 0 be the empty construction problem, such that $[0]_{\mathrm{OTM}}$ is the class of effectively OTM-computable problems. We have established $[0]_{\mathrm{OTM}} < [\mathsf{PP}_{\mathsf{fin}}]_{\mathrm{OTM}} < [\mathsf{PP}]_{\mathrm{OTM}}$.

Proof. The first inequality follows from the non-computability of $\mathsf{PP}_{\mathsf{fin}}$, see [3, Proposition 15], and the second from Theorem 4.7. □

There may be more OTM-degrees below $[\mathsf{PP}_{\mathsf{fin}}]_{\mathrm{OTM}}$, or degrees between $[\mathsf{PP}_{\mathsf{fin}}]_{\mathrm{OTM}}$ and $[\mathsf{PP}]_{\mathrm{OTM}}$, but we do not know any yet. These will all require We mention the following open question:

Question 3. Let MPP be the **Multiple Picking Principle** "for every set x, there is a finite set $y \subseteq x$". How does it OTM-compare to PP and $\mathsf{PP}_{\mathsf{fin}}$?

It is easily argued that $\mathsf{MPP} \not\leq_{\mathrm{OTM}} \mathsf{PP}_{\mathsf{fin}}$ since if it were computable by program P^F, with F a canonification of $\mathsf{PP}_{\mathsf{fin}}$, then also $\mathsf{PP} \leq_{\mathrm{OTM}} \mathsf{PP}_{\mathsf{fin}}$: Let Q^F execute P^F and when it outputs a finite set y, simply use the oracle to obtain a single element $F(y)$. However, we don't know if $\mathsf{PP}_{\mathsf{fin}} \leq_{\mathrm{OTM}} \mathsf{MPP}$.

Chapter 5

Reductions with miracle OTM-programs

In this chapter we will combine different notions of the previous chapters, mainly to show how these different notions may be made to fit a wider range of potential contexts. Before motivating which contexts these may be, we first show how we combine our different types of reductions.

Definition 5.1. Let $S \subseteq V^2$ be a construction problem. We define **S-OTM-computability** and **(strong) ordinal Weihrauch with S-oracle reducibility** ($\leq^S_{\mathrm{oW}}$ and $\leq^S_{\mathrm{soW}}$) in the same way as before, but with miracle OTM-programs P^F and Q^F, for any S-canonification F.

By definition a construction problem R is S-OTM-computable if and only if $R \leq_{\mathrm{OTM}} S$, so we have already considered S-OTM-computability. But S-Weihrauch reducibility is new, and will get some attention.

We will show that S-OTM-reducibility is not really of special interest either.

Proposition 5.2. Let $S, R, R' \subset V^2$ be construction problems, and define S-OTM-reducibility ($\leq^S_{\mathrm{OTM}}$) to be like OTM-reducibility, but with not just a `Miracle` tape, but also a `MiracleS` tape on which an S-oracle acts. Suppose that there are $x_S \in \mathrm{dom}(S)$ and $x_{R'} \in \mathrm{dom}(R')$ that are OTM-computable. Then $R \leq^S_{\mathrm{OTM}} R'$ if and only if $R \leq^{R'}_{\mathrm{OTM}} S$ if and only if $R \leq_{\mathrm{OTM}} \langle S, R' \rangle$.

Proof. The equivalence between $R \leq^S_{\mathrm{OTM}} R'$ and $R \leq^{R'}_{\mathrm{OTM}} S$ follows from the observation that a program witnessing the first is turned into a program witnessing the second, and vice versa, by switching the `Miracle` and the `MiracleS` tape everywhere.

To show that $R \leq_{\mathrm{OTM}} \langle S, R' \rangle$ implies $R \leq^S_{\mathrm{OTM}} R'$, let Q^G be a program witnessing the former and let F and F' be arbitrary S and R'-canonifications respectively. Define program $P^{F,F'}$ to be like Q^G but when the miracle command is used on some $\langle x, y \rangle$, it separately uses the S-miracle command on x and the other miracle command on y, pairing the results. This behavior of F and

F' may be extended to a full $\langle S, R\rangle$-canonification and so $P^{F,F'}$ will simulate Q^G for any R-instance and thus $P^{F,F'}$ witnesses $R \leq^S_{\mathrm{OTM}} R'$.

To show that $R \leq^S_{\mathrm{OTM}} R'$ implies $R \leq_{\mathrm{OTM}} \langle S, R'\rangle$, let $P^{F,F'}$ witness $R \leq^S_{\mathrm{OTM}} R'$ and let G be an arbitrary $\langle S, R'\rangle$-canonification. Define F to be the S-canonification satisfying $F(x) = \pi_1 \circ G(\langle x, x_{R'}\rangle)$ for all $x \in \mathrm{dom}(S)$, and similarly define $F'(x) = \pi_2 \circ G(\langle x_S, x\rangle)$ for all $x \in \mathrm{dom}(R')$. Now let Q^G be like $P^{F,F'}$, but whenever the S-miracle command is used on some x, it instead uses the miracle command on $\langle x, x_{R'}\rangle$ and when this returns some code for $G(\langle x, x_{R'}\rangle)$, it copies down only the first element, i.e. a code for $F(x)$. Similarly, when the other miracle command is used, Q^G uses $G(\langle x_S, x\rangle)$, and preserves only the code for $F'(x)$. Now Q^G is simulating $P^{F,F'}$ and so witnesses $R \leq_{\mathrm{OTM}} \langle S, R'\rangle$. □

In this chapter, Weihrauch with miracle reductions are used to say something about the relative computability of problems, when we do not yet know if an ordinal Weihrauch reduction exists. We use the fact that $\mathsf{PP} \not\leq_{\mathrm{OTM}} \mathsf{PP}_{\mathsf{fin}}$ from Theorem 4.7, since now a $\mathsf{PP}_{\mathsf{fin}}$-oracle will not immediately be enough to compute for instance WO.

5.1 The Compactness Theorem

We need to do some work before we can consider the effectivity of the Compactness Theorem. Before, we were only dealing with set theory, for which we have described a coding. We have not yet done this for first-order logic.

Definition 5.3. The **Compactness Theorem of First-Order Logic** is the following: "If Σ is a set of sentences such that for every finite subset i of Σ there is a model $\mathcal{M}_i \vDash i$, then there is a model $M \vDash \Sigma$."

The following is the construction problem: let $(x, y) \in \mathsf{COMP}(\mathsf{FOL})$ if and only if "If $x = \big(\mathcal{L}, \Sigma, \{\,(i, \mathcal{M}_i) \mid i \in \mathcal{P}^{<\omega}(\Sigma)\,\}\big)$ is such that Σ is a set of sentences in FO-language $\mathcal{L}$ and for all finite $i \subset \Sigma$ we have $\mathcal{M}_i \vDash i$, then $y = \mathcal{M}$ such that $\mathcal{M} \vDash \Sigma$."

We have to make some coding to represent the mentioned objects. To represent the predicates $\mathcal{R}$, variables $\mathcal{X}$ and function symbols $\mathcal{F}$ of $\mathcal{L} = \langle \mathcal{R}, \mathcal{X}, \mathcal{F}\rangle$ we may use tuples of the form $\langle i, k, x\rangle$ where $i = 0$ for predicates, $i = 1$ for variables and $i = 2$ for function symbols, k refers to the arity of the symbol, and x is the label/index to distinguish the unique symbol.

To code the sentences φ, we can then use a Gödel numbering $\ulcorner\varphi\urcorner$, and models are an interpretation function from the symbols to the models' domain (which can be other arbitrary sets).

Lemma 5.4. Let Σ be a finite set of sentences in first-order language $\mathcal{L}$, and $c_{\mathcal{L}}$ be a code for $\mathcal{L}$. Suppose that a finite $\mathcal{L}$-model for Σ exists. There is an OTM-program that computes a finite set of finite models for Σ, independent of the code for $c_{\mathcal{L}}$.

Remark: If $\mathcal{L}$ is well-ordered, it is possible to compute a single model for Σ, by enumerating $L[\mathcal{L}]$ and finding the first model for Σ. Yet, we did not assume $\mathcal{L}$ to be well-ordered; doing so would make it hard to compute instances of $\mathsf{COMP}(\mathsf{FOL})$.

Proof. Since Σ is a finite set, only a finite number of symbols appear in it. Let $\mathbf{S}$ be the symbols appearing in Σ and $\mathbf{S}^c$ the symbols not appearing in Σ. Let $\mathcal{L}|_{\mathbf{S}}$ be $\mathcal{L}$ restricted to $\mathbf{S}$. If $\mathcal{M}$ is an $\mathcal{L}$-structure that agrees on $\mathbf{S}$ with $\mathcal{L}|_{\mathbf{S}}$-structure $\mathcal{M}'$, then $\mathcal{M} \vDash \Sigma$ iff $\mathcal{M}' \vDash \Sigma$. By assumption, a finite $\mathcal{M} \vDash \Sigma$ exists. The model $\mathcal{M}|_{\mathbf{S}}$ is then also finite and $\mathcal{M}|_{\mathbf{S}} \vDash \Sigma$. Therefore a finite $\mathcal{L}|_{\mathbf{S}}$-model for Σ exists.

Let a program P do the following: first check if an $\mathcal{L}|_{\mathbf{S}}$-model exists of size $n = 1$, and keep increasing n until it finds a size for which we find a model. For any arbitrary enumeration of the symbols in $\mathbf{S}$ we can enumerate all possible interpretations of the symbols in $\mathbf{S}$ for domain $D = \{0, 1, \ldots, n-1\}$. For any of these interpretations, P can check if all sentences in Σ are true. If one of them models Σ, P will keep enumerating all of the remaining models of size n, collecting the ones that model Σ. If there is no model of size n, P continues with size $n+1$. At some point, P will find an n and a set of models $\mathcal{C}$.

To then extend all $\mathcal{M} \in \mathcal{C}$ to full $\mathcal{L}$-structures, we need to define how the constants, function symbols and relational symbols in $\mathbf{S}^c$ are interpreted. Thus, for every $\mathcal{M} \in \mathcal{C}$, define $\mathcal{M}^*$ to be $\mathcal{M}$ extended with every constant in $\mathbf{S}^c$ interpreted as 0, let every function in $\mathbf{S}^c$ for every arity k be interpreted as $f(\vec{a}) = 0$ for all $\vec{a} \in D^k$ and let every predicate symbol in $\mathbf{S}^c$ be interpreted as the empty set. Then P can run through the set of $\mathcal{M} \in \mathcal{C}$ and for every $\mathcal{M}$ compute model $\mathcal{M}^*$. When P is done, it has computed a finite set of models for Σ. □

Corollary 5.5. Let Σ be a finite set of sentences in first-order language $\mathcal{L}$ such that a finite $\mathcal{L}$-model for Σ exists. Then a finite $\mathcal{L}$-model for Σ is $\mathsf{PP}_{\mathsf{fin}}$-OTM-computable.

Now a program with access to a $\mathsf{PP}_{\mathsf{fin}}$-oracle can use this. The following proof ensures that these finite models exist.

Proposition 5.6. We have $\mathsf{BPI} \leq^{\mathsf{PP}_{\mathsf{fin}}}_{\mathrm{soW}} \mathsf{COMP}(\mathsf{FOL})$.

Proof. The proof in [12, Theorem 2.2.] is OTM-effective, if $\mathsf{PP}_{\mathsf{fin}}$ is computable.

Let F be any $\mathsf{PP}_{\mathsf{fin}}$-canonification and instance $X = (B, \leq)$ be a Boolean algebra. For a program P^F to create an instance of $\mathsf{COMP}(\mathsf{FOL})$, it first builds a language $\mathcal{L}$: for every element $u \in B$, program P^F adds a constant u to $\mathcal{L}$. Additionally, $\mathcal{L}$ has one unary predicate I, which may be coded as $\langle 0, 1, 0 \rangle$. Next, P^F computes the $\leq$-maximal elements of B: let 0^B and 1^B denote these constants.

Then P^F can compute Σ, a set of sentences such that a model $M \vDash \Sigma$

induces a prime ideal $\{u \in B \mid M \vDash I(u)\}$. That is, Σ contains the following:

$$I(0^B)$$
$$\neg I(1^B)$$
$$I(u) \vee I(-u) \text{ for every } u \in B$$
$$I(u_1) \wedge \ldots \wedge I(u_k) \to I(u_1) + \ldots + I(u_k) \text{ for all } u_1, \ldots, u_k \in B.$$

Every finite subset i of Σ contains a finite number of constants and then has a corresponding finite subalgebra $B|_i$ of $(B, \leq)$. Since $B|_i$ is finite, it has a prime ideal corresponding to a model of i. Then by Corollary 5.5, let P^F (using the $\mathsf{PP}_{\mathsf{fin}}$-canonification F) compute a model for every i. Then, combining this all, P^F returns $X = \big(\mathcal{L}, \Sigma, \langle\, (i, \mathcal{M}_i) \mid i \in \mathcal{P}^{<\omega}(\Sigma) \,\rangle\big)$.

Then let $\mathcal{M}$ be the model returned by the $\mathsf{COMP}(\mathsf{FOL})$-canonification. Now program Q^F may simply check for which of the constants u, $\mathcal{M} \vDash I(u)$. By design of Σ, $\{u \in B \vDash \mathcal{M} \vDash I(u)\}$ is a prime ideal on B. □

Proposition 5.7. We have $\mathsf{COMP}(\mathsf{FOL}) <_{\mathrm{soW}} \mathsf{WO}$.

Proof. The reduction $\mathsf{COMP}(\mathsf{FOL}) \leq_{\mathrm{soW}} \mathsf{WO}$ is a consequence of [3, Theorem 20.], and the other direction does not exist by Theorem 3.9, since [12, Theorems 7.1 & 7.8] shows a countable model of ZF where the Compactness Theorem holds, but not the Well-Ordering Theorem. □

Proposition 5.8. We have $\mathsf{UF} \not\leq_{\mathrm{oW}} \mathsf{COMP}(\mathsf{FOL})$.

Proof. Suppose for contradiction that $\mathsf{UF} \leq_{\mathrm{oW}} \mathsf{COMP}(\mathsf{FOL})$. Consider instance $\omega \in \mathrm{dom}(\mathsf{UF})$. For any UF-solution y to ω, we have $|y| = |\mathcal{P}(\omega)| > |\omega|$. Then by Theorem 3.10, there is an $x \in \mathrm{dom}(\mathsf{COMP}(\mathsf{FOL}))$ with $|\mathrm{tc}(\{x\})| \leq |\omega|$, such that $(x, y) \in \mathsf{COMP}(\mathsf{FOL})$ implies $|\mathrm{tc}(\{y\})| > \omega$.

Since x is an instance of $\mathsf{COMP}(\mathsf{FOL})$, $\mathcal{L} \in x$. We know $|\mathrm{tc}(\{x\})| \leq |\omega|$ so we also have $|\mathcal{L}| \leq |\omega|$. Then by the Löwenheim-Skolem Theorem, there must be a countable model $\mathcal{M}$ such that $(x, \mathcal{M}) \in \mathsf{COMP}(\mathsf{FOL})$. But we already established that $(x, y) \in \mathsf{COMP}(\mathsf{FOL})$ implies $|\mathrm{tc}(\{y\})| > \omega$, so we conclude that $\mathsf{UF} \not\leq \mathsf{COMP}(\mathsf{FOL})$. □

We have not found another reduction with $\mathsf{COMP}(\mathsf{FOL})$, leaving us to wonder the following:

Question 4. Is it the case $\mathsf{COMP}(\mathsf{FOL}) \leq_{\mathrm{soW}} \mathsf{UF}$, or $\mathsf{BPI} \leq_{\mathrm{soW}} \mathsf{COMP}(\mathsf{FOL})$, or $\mathsf{COMP}(\mathsf{FOL}) \leq_{\mathrm{soW}} \mathsf{BPI}$, or can it be proven that these reductions do not exist? In the latter case, are there natural construction problems S such that the Weihrauch with S-oracle reductions hold?

Chapter 6

Conclusion and Future Work

We have defined a framework of OTM-computability and several notions of reducibility to determine the (relative) effectivity of set theoretical statements. We have discussed the set theoretical assumptions and properties that were used to prove theorems establishing irreducibility and then expanded on the results in [3] by considering the relative computability of more choice principles.

We have not been able to consider choice principles that are topological or that center around cardinalities. It seems less straightforward to us to do this. Yet, this is needed in order to achieve a more complete understanding of the relative computability of well-known choice variants.

We have attempted to contribute to a better understanding of miracle OTM-reductions, and in doing so have been able to find the first known non-trivial OTM-irreducibility result: $\mathsf{PP} \not\leq_{\mathrm{OTM}} \mathsf{PP}_{\mathsf{fin}}$. We think that this opens a window to finding more of these irreducibilities. Also, it might be possible to generalize this and determine general conditions under which an OTM-reduction does not exist.

Finally, we introduced two new types of reductions. We have not fully studied these, but we think they could potentially be of use. The relation $\leq_{\alpha\text{-OTM}}$ may be a good tool lying between Weihrauch and OTM-reducibility, and whenever construction problem S is of a relative lower OTM-degree or if it unknown whether S is OTM-computable, ordinal Weihrauch with S-oracle reducibility may be a logical being able to study the Weihrauch lattice further.

These are the Weihrauch reductions mentioned in this thesis.

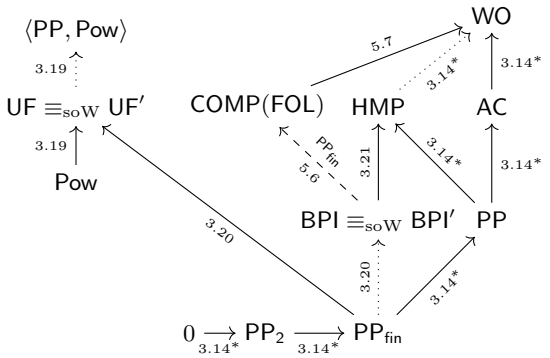


Figure 6.1: Graph of Weihrauch reductions. Dotted arrow from R to R': $R \leq_{\mathrm{sW}} R'$. Full arrow: $R <_{\mathrm{sW}} R'$. Numbers refer to propositions in this thesis, and an asterisk indicates a result in [3].

In the following graph, arrows represent miracle OTM-reductions instead. In the case of $\mathsf{PP}_{\mathsf{fin}} <_{\mathrm{OTM}} \mathsf{PP}$, the fact $\mathsf{PP}_{\mathsf{fin}} \leq_{\mathrm{OTM}} \mathsf{PP}$ is from [3].



Figure 6.2: Graph of miracle OTM-reductions. Numbers refer to propositions in this thesis, and an asterisk indicates a result in [3].

Bibliography

[1] Vasco Brattka and Guido Gherardi. "Weihrauch degrees, omniscience principles and weak computability". In: *OpenAccess Series in Informatics*. Vol. 11. 2009. DOI: 10.2178/jsl/1294170993.

[2] Vasco Brattka, Guido Gherardi, and Alberto Marcone. "The Bolzano-Weierstrass Theorem is the jump of Weak Ko{double acute}nig's Lemma". In: *Annals of Pure and Applied Logic* 163.6 (2012). ISSN: 01680072. DOI: 10.1016/j.apal.2011.10.006.

[3] Merlin Carl. "Effectivity and reducibility with ordinal Turing machines". In: *Computability* 10.4 (2021). ISSN: 22113576. DOI: 10.3233/COM-210307.

[4] Merlin Carl. "Generalized effective reducibility". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 9709. 2016. DOI: 10.1007/978-3-319-40189-8{_}23.

[5] Merlin Carl. *Ordinal Computability*. De Gruyter, Sept. 2019. ISBN: 9783110496154. DOI: 10.1515/9783110496154. URL: https://www.degruyter.com/document/doi/10.1515/9783110496154/html.

[6] Merlin Carl, Lorenzo Galeotti, and Robert Passmann. "Realisability for infinitary intuitionistic set theory". In: *Annals of Pure and Applied Logic* 174.6 (2023). ISSN: 01680072. DOI: 10.1016/j.apal.2023.103259.

[7] Keith J. Devlin. *Constructibility*. 2017. ISBN: 0-387-13258-9. DOI: 10.1017/9781316717219.

[8] Sy D. Friedman. *Fine Structure and Class Forcing*. 2000. DOI: 10.1515/9783110809114.

[9] Lorenzo Galeotti and Hugo Nobrega. "Towards computable analysis on the generalised real line". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 10307 LNCS. 2017. DOI: 10.1007/978-3-319-58741-7{_}24.

[10] Joel David Hamkins. *Mutually generics - MathOverflow*. URL: https://mathoverflow.net/questions/70211/mutually-generics (visited on 12/21/2023).

[11] Paul Howard and Jean Rubin. *Consequences of the Axiom of Choice*. American Mathematical Society, 1998. ISBN: 0821809776.

[12] Thomas J Jech. *The axiom of choice*. eng. Studies in logic and the foundations of mathematics, vol. 75. Amsterdam etc: North-Holland Publishing Company, 1973. ISBN: 0444104844.

[13] Thomas J. Jech. *Set Theory: The Third Millennium Edition, revised and expanded*. Springer, 2003, p. 621. ISBN: 3-540-44058-2.

[14] Akihiro Kanamori. *The Higher Infinite*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. ISBN: 978-3-540-88866-6. DOI: 10.1007/978-3-540-88867-3.

[15] Peter Koepke. "Turing Computations On Ordinals". In: *Bulletin of Symbolic Logic* 11.3 (2005). ISSN: 1079-8986. DOI: 10.2178/bsl/1122038993.

[16] Kenneth Kunen. "Set Theory: An introduction to independence proofs". In: *Elsevier Science Publishers B.V* (1980). ISSN: 00224812.

[17] Elliott Mendelson. *Introduction to Mathematical Logic*. Third Edition. 1987. ISBN: 0534066240.

[18] G. P. Monro. "On generic extensions without the axiom of choice". In: *Journal of Symbolic Logic* 48.1 (1983). ISSN: 0022-4812. DOI: 10.2307/2273318.

[19] John Myhill. "Constructive set theory". In: *Journal of Symbolic Logic* 40.3 (1975). ISSN: 0022-4812. DOI: 10.2307/2272159.

[20] Klaus Weihrauch. *Computable Analysis*. Springer Heidelberg, 2000. ISBN: 3540668179.

[21] Andrzej Zarach. "Unions of ZF–models which are themselves ZF–models". In: *Studies in Logic and the Foundations of Mathematics* 108.C (1982). ISSN: 0049237X. DOI: 10.1016/S0049-237X(09)70519-1.