Topics in Computability

by

Sapir Ben-Shahar

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Master of Science
in Mathematics.

Victoria University of Wellington
2025

Abstract

This thesis studies two topics in computability. The first is about computable metric and Polish spaces. We compare different notions of effective presentability and construct some spaces that are 'almost computable', in the sense that they do not have a computable presentation but they do have both left-c.e. and right-c.e. presentations. The second part studies c.e. Quasi-degrees (Q-degrees) and c.e. strong Quasi-degrees (sQ-degrees), which have interesting connections to algebra. We show that the c.e. sQ-degrees are not distributive, embed the lattice N_5 into them and show that no initial segment forms a lattice. We construct a non-computable c.e. set that has no c.e. simple set Q-below it. We also briefly study the relationship of sQ-degrees to wtt-degrees. Finally we show there is a minimal pair of sQ-degrees within the same Q-degree, and that if a degree is half of a minimal pair in the Q-degrees, it is also half of a minimal pair in the Turing degrees.

Acknowledgments

Thank you to my fantastic supervisors, Prof. Rod Downey and A/Prof. Sasha Melnikov, for suggesting fun problems to work on and for providing ongoing guidance and advice. I am grateful for your teaching of the relevant background, sharing your enthusiasm for the subject, and offering patience and feedback on my writing. Thank you to HT Koh, PhD student at Nanyang Technological University, for working with me on the second part of the Polish spaces and Polish groups project, and forcing me to write more formally and with more details. Thank you to Prof Mariya Soskova from University of Wisconsin-Madison who helped with the Q-degrees project.

Contents

Chapter 1

Background

1.1 General background

1.1.1 Early beginnings

Mathematicians have been interested in computing things for eons. Nearly four thousand years ago the Babylonians calculated $\sqrt{2}$ to be $1 + \frac{24}{60} + \frac{51}{60^2} + \frac{10}{60^3} = 1.414212963$, which is incredibly accurate: it only differs from $\sqrt{2}$ by about 0.0000006. Following this is a long history of mathematicians trying to compute important constants such as π and e explicitly, which eventually led to the formalisation of ideas such as limits, and the development of analysis by Cauchy and other analysts in the eighteenth and nineteenth centuries [27, 18].

Indeed, around the turn of the nineteenth century there was a growing interest in making mathematics more formal and rigorous, and finding mechanical (or effective) methods to solve problems and to determine whether logical statements were true or false. In 1900, Hilbert posed a list of unsolved problems which he thought should direct mathematical efforts. The second problem was about the axioms of arithmetic: *to prove that they are not contradictory, that is, that a definite number of logical steps based upon them can never lead to contradictory results.* The tenth problem was, *given* *a Diophantine equation with any number of unknown quantities and with rational integral numerical coefficients, to devise a process according to which it can be determined by a finite number of operations whether the equation is solvable in rational integers* [37]. Essentially, Hilbert was asking for an effective process, or an algorithm, to determine whether a Diophantine equation has integer solutions, given its integer coefficients.

While there wasn't yet any formal definition of what an *effective process* was, the mathematicians of the early 20th century did have a sharp intuitive notion for what an algorithmic process was. For instance, in 1912 Borel gave an informal definition of a computable real number: *We say that a number α is computable when, given any natural number n, we know how to obtain a rational number q that differs from α by less than $\frac{1}{n}$.* Although Borel's definition for 'obtaining' the rational was vague due to the lack of formalism around computation, he did say that the essential thing was for the operation (of obtaining the rational number) to be executable in a finite amount of time, by a sure and unambiguous method. The *Entscheidungsproblem* (the 'decision problem'), posed by Hilbert, was also an important and well-known problem. It asks for *a process which, given a logical expression, permits the determination of its validity.*

Another example from the early 20th century comes from Dehn, who analysed algorithmic questions about finitely presented groups. Dehn gave geometric algorithms for solving the word problem for certain kinds of finitely presented groups. The word problem in a group is to decide if two words, i.e. strings consisting of the generating elements, are equal in the group. Dehn noted that these methods were specific to certain classes of groups and did not apply in general. This prompted Dehn to articulate three questions, which significantly drove the study of what is now called combinatorial group theory [18]:

1. Is the word problem of every finitely presented group algorithmically decidable (solvable)?

2. Given x and y in a finitely presented group, can we algorithmically decide if x is conjugate to y?

3. Given two finitely presented groups can we algorithmically decide if they are isomorphic?

In fact, almost all of pre-20th century mathematics was fundamentally algorithmic, and proofs were often constructive. One notable exception to this was Hilbert's Basis Theorem from 1890, which proves that every algebraic set over a field can be described as the set of common roots of finitely many polynomial equations. Hilbert's proof did not actually compute this finite basis, but showed that the basis must exist. The non-constructive nature of this proof made it a rather controversial result at the time, as there was generally a lot of scepticism towards non-constructive proofs. Over time, acceptance for non-constructive proofs grew. For example, early editions of van der Waerden's books on modern algebra in the 1940s had algorithmic proofs of various results on rings and fields. But then in later editions of van der Waerden's books many of the algorithmic proofs were replaced by slicker, though less constructive proofs [18].

Then in 1931 Gödel proved that any consistent system of formal logic powerful enough to talk about arithmetic must include true statements that cannot be proven. He showed that such a system will have self-reference, and produced a formula P that says 'P has no proof'. Then if P is true, it has no proof. If P is false, that is, the negation of P is true, then P has a proof. But then that would contradict that P is false. So if the logical system is consistent, P must be true, and thus has no proof. In particular, the axioms of arithmetic cannot prove the consistency of arithmetic, giving a negative answer to Hilbert's second problem [31].

1.1.2 Computability

To answer Hilbert's tenth problem (and the many other questions involving effective processes), some more clarity was needed on the notion of an effective process to solve a problem, or as it was generally known at the time, 'effective calculability'. In 1936 Church proposed two equivalent definitions for effectively calculable functions, and further showed that not every problem was solvable (decidable) in this way. In particular, Church showed that the Entscheidungsproblem (Hilbert's decision problem) is unsolvable, that is, it is not effectively calculable [15].

The two notions that Church wrote about were the lambda calculus, due jointly to Church and Kleene, and recursive functions, due jointly to Jacques Herbrand and Gödel. These two very different notions of effective calculability are in fact equivalent, as was shown by Kleene, and also partly by Rosser and Church. Church argued that these definitions were both natural notions that captured the idea of effective calculability because for any function that was effectively calculable, there is an algorithm for calculating the function's values. Conversely, any function for which there is an algorithm to calculate its values will be effectively calculable under Church's definition [15].

In 1936 (with a correction added in 1937), Turing [75, 76] published a paper introducing Turing Machines, which he developed for the same purpose; to formalise the notion of effective calculability. The main purpose of these papers was to study computable numbers and functions, and to show that the Entscheidungsproblem is undecidable (among other applications). Turing showed that being Turing computable was equivalent to being lambda-computable and partial recursive, so in fact all of these definitions captured the same class of functions: the partial computable functions [75].

1.1.3 Non-computability and degrees

A particularly important undecidability result showed that there is no algorithm for determining whether an arbitrary Diophantine equation has an integer solution. That is, Hilbert's tenth problem is undecidable. This proof was only completed in 1973, in contrast to Church's much earlier result in 1936. Novikov in 1955 [59] and Boone in 1959 [9] showed that there are finitely presented groups whose word problem is undecidable, giving a negative answer to Dehn's first question. That is, there are c.e. presented groups (see section 1.2.3 for definitions) with no computable presentation. Computable presentations and c.e. presentations are two of the main notions of effective presentability for countable discrete algebraic structures, and this result shows that these main notions differ up to isomorphism for groups [18].

Adyan (in 1957) and Rabin (in 1958) showed that it is undecidable to tell if two given finitely presented groups are isomorphic [18]. In fact, the answer to all three of Dehn's questions turned out to be negative, and the techniques developed to answer these questions have been enormously influential in group theory. In 1958 Markov used Adyan's and Rabin's results to show that it is undecidable to tell whether two compact manifolds of dimension $n \geqslant 4$, given as simplicial complexes, are homeomorphic. The proof of this fact is accomplished by computably transforming a finite presentation of a group G into a simplex that is homeomorphic to an n-manifold whose fundamental group is G, so that two groups are isomorphic if and only if the simplices constructed from these groups are homeomorphic. That is, Markov reduced the isomorphism problem for group presentations to the homeomorphism problem for simplices representing n-manifolds. Then, because the isomorphism problem for group presentations is undecidable, so is the homeomorphism problem for n-manifolds [18].

Coding a known undecidable problem Q into another problem P in a computable way, that is, *reducing* the decidability of one problem to another, is a very useful technique. This shows that P is itself undecidable, because if P *was* decidable, the algorithm for P could be used in combination with the coding to compute the undecidable problem Q, which is impossible. Essentially, this says that P is *at least* as computationally complicated, or difficult to solve, as Q. More generally, two problems P_1 and P_2 can be compared in terms of their relative computational complexity, so $P_1 \leqslant P_2$ means that P_2 is at least is hard as P_1. That is, P_1 reduces to P_2, or P_2 computes P_1, in the sense that a method for solving P_2 would give a method for solving P_1.

Turing introduced the most general method of making this comparison, called *Turing reducibility*, denoted as $P_1 \leqslant_T P_2$. If both $P_1 \leqslant_T P_2$ and $P_2 \leqslant_T P_1$, we say P_1 and P_2 are *Turing equivalent*, and denote this by $P_1 \equiv_T P_2$. This relation, $\equiv_T$, is an equivalence relation on sets, and thus Turing reducibility leads to the concept of *Turing degrees*, which are the equivalence classes of $\equiv_T$. Then P_1 and P_2 have the same Turing degree if $P_1 \equiv_T P_2$, and P_2 has a higher degree than P_1 if $P_1 \leqslant_T P_2$, but $P_2 \nleqslant_T P_1$, and finally P_1 and P_2 have incomparable degrees if neither reduces to the other. When a Turing degree contains a c.e. set, we say it is a *c.e. Turing degree* (see Section 1.2.1 for definitions) [75].

The Turing degrees and c.e. Turing degrees have natural correspondences to other areas of mathematics, which means that studying these degree structures can give insight into other problems. For example, for any Turing degree there is a finitely generated group whose word problem has that degree. Further, if the Turing degree is a c.e. Turing degree, the same is true for a finitely presented group. Also, for each c.e. Turing degree there is a computable class of n-manifolds for $n \geqslant 4$ whose homeomorphism problem has that degree. With this correspondence we now know that, for example, the existence of infinitely many c.e. Turing degrees implies that there are infinitely many genuinely different word problems for finitely presented groups, and similarly for homeomorphism problems of n-manifolds ($n \geqslant 4$). That is, any result about the structure of the c.e. Turing degrees can immediately be translated into results in other areas of mathematics.

1.1.4 Post's Problem

In 1944, Post [68] examined the existing undecidability results in the literature. These were all proven in specific contexts, such as in groups (for example to answer Dehn's questions), or in first order logic (for the Entscheidungsproblem). Essentially, what all the undecidability proofs did at the time was encode Turing machines in some way into the specific context and objects under study. Implicitly, effectively generated undecidable sets such as the halting set (defined in Section 1.2.1) were being used in all of these early undecidability results. Post's idea was to abstract away from the specific context such as the algebra or logic. He suggested that by stripping away this additional structure and focusing solely on sets arising from Turing machine computations, we could study relative computational complexity with more clarity. This idea of abstracting the essential properties away from specific (applied) contexts is analogous to how linear algebra began as the study of linear transformations, and group theory began as the study of group actions on objects, and so on.

The idea of reduction is also widely used in computational complexity theory, especially in the theory of NP-completeness. In computational complexity theory, time constraints are added to the usual computability notions [30].

All the computably enumerable problems known at the time either had the same Turing degree as the halting set, or were computable. This prompted Post to ask in 1944 whether there were unsolvable (non-computable) problems that were computably enumerable, but had a lower degree than the halting set. That is, in the c.e. Turing degrees, Post was seeking a degree that was strictly between the lowest degree (the degree of the empty set, which contains all the computable sets and is denoted by $\mathbf{0}$), and the highest degree (the degree of the halting set, also called 'zero jump' and denoted by $\mathbf{0}'$). Specifically, Post was looking for a structural property of c.e. sets that would guarantee the sets are between $\mathbf{0}$ and $\mathbf{0}'$.

There were many attempts to solve Post's problem, and these drove the development of new tools and characterisations. These developments have had far-reaching implications in computability theory way beyond Post's problem. See Section 1.3 for more details on some of these attempts. Essentially, Post's idea of abstracting away from specific contexts was fundamentally important for the development of computability theory and the study of degree structures and relative computational complexity.

1.2 More technical background

Here we revisit some of the topics mentioned in the informal introduction in more detail, and introduce some formal definitions.

1.2.1 Computability, Computable Reals

Recall that the partial computable functions are those functions computed by a Turing machine. We can also think of the computability of a set by saying that a set is computable if its characteristic function is computable. Another important concept is that of computably enumerable (c.e.) sets, which are the domains of partial computable functions. That is, A is c.e. if for some e, $x \in A$ if and only if $\varphi_e(x)\downarrow$ (the e-th Turing machine halts on input x). This is one of many equivalent characterisations of c.e. sets. With c.e. sets, only *positive* information about the set is computable: when an element x belongs to the set, we will eventually find out as $\varphi_e(x)$ will at some point halt, and once we know the element is in the set this does not change. However, if the element is not in the set, $\varphi_e(x)$ will simply never converge, but at any finite stage of the computation we do not know whether the element is truly not in the set or if we just need to wait longer. A set is computable if and only if both it and its complement are computably enumerable.

A particularly important non-computable set is the *halting set*, first explicitly defined by Post in 1944 [68]:

$$K = \{e : \varphi_e(e)\downarrow\}$$

The halting set is famously not computable (i.e. the halting problem of deciding if a given Turing machine will halt on its own index is undecidable), although it is computably enumerable (c.e.). Roughly speaking, to enumerate K, run all the Turing machines on their own inputs and when $\varphi_e(e)\downarrow$, enumerate e into K. Of course, we would not be able to just run each machine in turn and wait for it to halt, else we may get stuck forever waiting for a given machine to halt. So a computable process to enumerate the halting set could for instance run the first s Turing machines on their own input for s many steps at stage s, and if any of them halt, enumerate the relevant e into K_s (K at stage s).

Among c.e. sets the halting set has the highest degree of unsolvability, that is, the highest c.e. Turing degree, in the sense that any other c.e. set is Turing below the halting set. We call sets with this property *Turing complete*, so the halting set K is Turing complete.

Returning to numbers, the focus of Turing's initial 1936/1937 paper, the intuition is that a computable number is one which a Turing machine could compute to arbitrary precision. There are a number of equivalent ways to formalise this, some of which effectivise classical ways of constructing real numbers. We say that a number α is a *computable real number* if one of the following equivalent conditions hold.

(i) There is a computable function f such that $|f(n) - \alpha| \leqslant 2^{-n}$. That is, there is a computable fast Cauchy sequence of rationals $(f(n))_{n\in\omega}$ that converges to α (note that this is essentially Borel's definition).

(ii) There is a computable sequence of closed intervals $(C_n)_{n\in\omega}$ whose intersection is α, that is $\bigcap_{C_n} C_n = \{\alpha\}$

(iii) The left cut of α, $\alpha^- = \{r \in \mathbb{Q} : r < \alpha\}$, and the right cut of α, $\alpha^+ = \{r \in \mathbb{Q} : r > \alpha\}$, are both computably enumerable.

(iv) The set of all open intervals with rational endpoints that contain α, $\{(a, b) : a, b \in \mathbb{Q}, a < \alpha < b\}$, is computably enumerable.

It is straightforward to see that these conditions are all equivalent. Given f as in (i), we can construct a computable sequence of closed sets by setting $C_n = [f(n)-2^{-n}, f(n)+2^{-n}]$. From a computable sequence of closed intervals $(C_n)_{n\in\omega}$, we can for each n, compute $[a, b] = \bigcap_{i=0}^{n} C_i$ and then enumerate $\{r \in \mathbb{Q} : r < a\}$ into α^- and $\{r \in \mathbb{Q} : r > b\}$ into α^+. From an enumeration of the left and right cuts, we can get an enumeration of open intervals $\bigcup_s \{(a, b) : a \in \alpha_s^-, b \in \alpha_s^+\}$ (that is, at stage s look at the enumerations of α^-, α^+ up to stage s and add all open intervals with one end point from α_s^- and the other endpoint from α_s^+, which guarantees that α is contained in the interval). Finally from an enumeration of open intervals we can get a convergent fast Cauchy sequence by enumerating the open intervals until we get an interval (a, b) with $b - a \leqslant 2^{-n}$ and then setting $f(n) = \frac{a+b}{2}$.

We can also take condition (iii) above and only require one of α^- and α^+ to be computably enumerable. Then if the left cut is c.e. we say that α is a *left-c.e. real*, and if the right cut is c.e. we call α a *right-c.e. real*. Now α is a computable real if and only if it is both a left-c.e. and a right-c.e. real. This is a bit reminiscent of how a set A is computable if and only if both A and the complement of A are computably enumerable: two 'opposite' enumerations give us a computable thing.

Since there are countably many computable functions (Turing machines), only countably many real numbers are computable. Turing showed that these include all real algebraic numbers and many transcendental numbers including π and e (which shouldn't be too surprising given people have been computing these numbers for such a long time) [75]. A particularly nice thing about the computable real numbers is that they form a field. This was stated in 1954 by Rice, although it was likely known earlier. Rice also showed that $\mathcal{E}(i)$ is algebraically closed, where $\mathcal{E}$ denotes the field of computable reals [70].

1.2.2 Computable Polish Spaces

The real numbers (strictly speaking, the reals in the interval $(0, 1)$) are a computable field in the sense that there is a single Turing machine which emulates addition on the reals, and similarly for multiplication. That is, there is a Turing functional Φ such that when Φ is given fast Cauchy sequences of rationals $(q_i)_{i\in\omega}$ and $(q'_i)_{i\in\omega}$ that converge to the reals x and y respectively, then Φ outputs a fast Cauchy sequence of rationals that converges to $x + y$, and similarly for multiplication. This notion of computability of a function on the reals (in this case addition and multiplication) is known as *Type II* computability and is often attributed to Kleene [44] (see Chapter 2 for more on this).

This notion and the notion for computable reals essentially rely on the fact that the rationals are a countable (computable) dense subset of the reals, so $\mathbb{R}$ is the completion of a computable, countable set (the rationals). This idea was extended to computability notions of separable Banach spaces, and to complete separable metric spaces, called *Polish spaces.* Ceitin [10] in 1959 and Moschovakis [57] in 1964 independently introduced the following notion for computable ('recursive') metric spaces.

Definition 1. A *computable presentation* of a Polish space M is given by a sequence $(x_i)_{i\in\omega}$ and a complete metric d such that $(x_i)_{i\in\omega}$ is dense in (M, d), and the distances $d(x_i, x_j)$ are uniformly computable reals in i and j.

Being uniformly computable means that there is a single procedure which, given i and j, will compute $d(x_i, x_j)$. The points x_i are usually called special points, rational points (in analogy to the rationals in $\mathbb{R}$) or ideal points. Many natural examples of computable Polish spaces come from functional analysis and topological algebra. As with computable reals, we can also weaken this notion slightly:

Definition 2. For a Polish space M, we say that a complete metric d and a sequence $(x_i)_{i\in\omega}$ that is dense in (M, d) give:

- a *right-c.e. presentation* of M if $\{r \in \mathbb{Q} : r > d(x_i, x_j)\}$ are uniformly c.e. in i, j (that is, the distances $d(x_i, x_j)$ are uniformly right-c.e. reals in i and j);
- a *left-c.e. presentation* of M if $\{r \in \mathbb{Q} : r < d(x_i, x_j)\}$ are uniformly c.e. in i, j (that is, the distances $d(x_i, x_j)$ are uniformly left-c.e. reals in i and j);

We will say that a Polish space M is *computable* if it has a computable presentation, and similarly it is *left-c.e. (right-c.e.)* if it has a left-c.e. (right-c.e.) presentation.

1.2.3 Algebraic structures

For discrete, countable algebraic structures, the following notion of computability was independently proposed by Rabin [69] in 1960, and Mal'cev [51] in 1961:

Definition 3. A discrete, countable algebraic structure A with finitely many operations and relations is *computably presented* if there is an algebraic structure B isomorphic to A whose elements form a computable set of natural numbers, and the operations and relations on B are computable (as functions and relations on the natural numbers).

This isomorphic copy B is called a *computable presentation* of A, or a *computable (isomorphic) copy* of A, and we say that A is *computable.* There are also stronger notions, such as primitive recursive presentations and punctual presentations, among many others, as well as weaker notions.

For instance, an algebraic structure is *computably enumerably (c.e) presented* if it is isomorphic to the factor of a computable structure by a c.e. congruence. A congruence is an equivalence relation that 'respects' the operations of the structure. That is, for any operation f, whenever $x_i \cong y_i$ we have that $f(x_0, \ldots, x_n) \cong f(y_0, \ldots, y_n)$. A standard example of a c.e. presented structure is the factor of a computable group G by a c.e. normal subgroup H, which are quite common in group theory. Now, a finite presentation of a group is also a c.e. presentation of that group, and groups with undecidable word problems cannot have computable presentations. Under these definitions, Dehn's first question is asking if every finitely presented group is computable (has a computable presentation). In particular, the question asks if c.e. presentability and computable presentability differ for finitely presented groups [18]. Thus the negative solution to Dehn's first question says that c.e and computable presentability do indeed differ for finitely presented groups.

It is important to note that there are several different traditions in computable mathematics, and so there are differences in terminology when it comes to computable presentations of spaces [18]. The closely related topics of effective descriptive set theory and classical combinatorial group theory also have their own terminology and notation. The most basic fundamental definitions in all of these turn out to be either equivalent or closely related, but sometimes the same term can correspond to non-equivalent definitions. For example, in combinatorial group theory 'recursively presented groups' refers to what we call 'c.e. presented groups' and 'recursively presented groups with solvable word problems' refers to what we call a 'computable group' (or a 'computably presented group'). In the literature, 'recursive' and 'computable' are often used interchangeably, which means that 'recursive group' could be referring to either of these two notions [18]. This is one of many examples of differences in terminology. To avoid confusion, we will only use the terminology that we described in our definitions, and in particular will avoid using the term 'recursive'.

1.2.4 Reducibilities

As well as the Turing degrees, there are a number of other degree structures arising from other reducibilities, and these have natural ties to various other areas of mathematics too. The different degree structures are also interesting to study in relation to one another, as sometimes results about one degree structure give insight into other degree structures. Some common reducibilities include the following.

Definition 4.

- A set A is *Turing reducible* to a set B, denoted $A \leqslant_T B$, if for some e,
$$\Phi_e^B = A.$$
That is, the Turing machine Φ_e computes A when given B as an oracle. This is the most general reducibility.

- A set A is *many-one reducible* (called *m-reducible*) to a set B, denoted $A \leqslant_m B$, if and only if there is a computable function f such that for all x,

$$x \in A \text{ if and only if } f(x) \in B.$$

Further, if f is a one-to-one function, then we say A is *one-one reducible* to B, written $A \leqslant_1 B$.

- A set A is *truth table reducible (tt-reducible)* to a set B, written $A \leqslant_{tt} B$, if and only if there is a computable function h such that $h(x)$ is a Boolean condition (a propositional formula built from the atomic formulas '$n \in X$') and $x \in A$ if and only if B satisfies the condition $h(x)$. Nerode [71] showed that $A \leqslant_{tt} B$ if and only if there is a Turing procedure Φ_e, total for all oracles, such that $\Phi_e^B = A$.

- A set A is *weak truth table reducible (wtt-reducible)* to a set B, denoted $A \leqslant_{wtt} B$, if and only if there is a computable function ψ and a Turing procedure Ψ such that

  $$x \in A \text{ if and only if } \Psi^B(x) = 1$$

  and additionally, for all y the use of the computation $\Psi^B(y)$ is bounded by $\psi(y)$. Note that wtt-reducibility is Turing reducibility but with a computable bound on the use.

Each of these reducibilities gives rise to its own degree structure. It is not too difficult to see that $\leqslant_m$ implies $\leqslant_{tt}$ implies $\leqslant_{wtt}$ implies $\leqslant_T$. Reducibilities end up being important in other areas of mathematics. For example, the c.e. m-degrees correspond to the word problems of finitely presented semigroups [60]. That is, for any c.e. m-degree there is a finitely presented semigroup whose word problem has that degree. Another example is that tt-degrees are fundamental in algorithmic randomness, because the totality of a tt-reduction Φ_e makes it appropriate for working with measures.

A reducibility that is of particular interest to us is Q-reducibility, which gained interest because of its connections to algebra and relevance to Post's Programme (see Section 1.3.2).

Definition 5. A set A is *Quasi-reducible* (*Q-reducible*) to a set B, written $A \leqslant_Q B$, if there is a computable function f such that for every x, $x \in A$ if and only if $W_{f(x)} \subseteq B$. Further, A is *strong Quasi-reducible* (*sQ-reducible*) to B, denoted $A \leqslant_{sQ} B$ if additionally there is a computable function h such that $\max\{z : z \in W_{f(x)}\} < h(x)$.

In general, Quasi-reducibility is not actually a reducibility (hence the name 'quasi'). This is because $W_{f(x)}$ may be infinite, requiring an infinite amount of information to determine whether $x \in A$, while a reducibility should only use a finite amount of information. On the other hand sQ-reducibility is always a reducibility, because the computable function h bounds the amount of information used in each computation (in particular, $|W_{f(x)}|$). For c.e. sets Q-reducibility *is* a reducibility, because we can always assume that $W_{f(x)}$ is finite.

To see this, suppose we have c.e. sets $A \leqslant_Q B$ via f. For each $W_{f(x)}$ we will construct a finite $W_{g(x)} \subseteq W_{f(x)}$, so that $A \leqslant_Q B$ via g. Simultaneously enumerate A, B and every $W_{f(x)}$. Each $W_{g(x)}$ will copy $W_{f(x)}$ (so we always have $W_{g(x)} \subseteq W_{f(x)}$), but we only put a new element (which is in $W_{f(x)}$) into $W_{g(x)}$ at stages $s+1$ where $x \notin A_{s+1}$ and $W_{g(x),s} \subseteq B_s$. That is, we put a new element into $W_{g(x)}$ if B contains all the elements that we have thus far put into $W_{g(x)}$, but x is not currently in A (which indicates there is something else in $W_{f(x)}$ which isn't currently in B). Now if $x \in A$, then we stop enumerating new elements into $W_{g(x)}$ at the first stage t where $x \in A_t$, and so $W_{g(x)}$ is finite. If $x \notin A$, then $W_{f(x)} \nsubseteq B$. Then eventually at a stage s some element $z \in W_{f(x)} \setminus B$ appears in $W_{f(x),s}$ and is enumerated into $W_{g(x),s}$, at which point we stop enumerating new elements into $W_{g(x)}$ (because from now on $W_{g(x),t} \nsubseteq B_t$, for all $t \geqslant s$) and so $W_{g(x)}$ remains finite.

That is, at every stage s, we either have that $x \in A_s$, or that there is a unique element $z \in W_{g(x),s+1} \setminus B_{s+1}$ and in particular if $W_{g(x),s+1} \neq W_{g(x),s}$, we must have that $W_{g(x),s} \subseteq B_{s+1}$ (since we required this in order to enumerate anything new into $W_{g(x)}$). In light of this, we make the following definition.

Definition 6. A total computable function $m(x, s)$ is *Q-like* if $m(x, s) = z$

with z being the unique element described above. So $m(x, s) \in B_s$ only if $x \in A$ and if $m(x, s+1) \neq m(x, s)$ then $m(x, s) \in B_{s+1}$.

Now $A \leqslant_Q B$ if and only if there is a Q-like function m such that $\lim_s m(x, s) = m(x)$ exists and $m(x) \in B$ iff $x \in A$. Furthermore, $A \leqslant_{sQ} B$ if for all s, $m(x, s) < h(x)$ (where $h(x)$ is the computable function from the definition of sQ-reduction).

Strong quasi-reducibility was introduced by Omanadze in 1991 [61]. Q-reducibility is a natural weakening of m-reducibility, replacing singletons with c.e. sets. For c.e. sets, $A \leqslant_Q B$ implies $A \leqslant_T B$, although in general for non-c.e. sets there is no simple relationship between $\leqslant_Q$ and $\leqslant_T$: for any set A, the set $A^Q = \{e : W_e \subseteq A\}$ is Q-equivalent to A, but A^Q is also a non-trivial index set and so by Rice's theorem, $K \leqslant_T A^Q$. In particular, the Q-degree of $\emptyset$ contains a set that is Turing above the halting set, while the Turing degree of $\emptyset$ contains only the computable sets. Notice also that sQ-reducibility is the analog of wtt-reducibility but for Q-reductions instead of Turing reductions (the use is computably bounded). In fact, we have that for c.e. sets, $A \leqslant_{sQ} B$ implies $A \leqslant_{wtt} B$.

As with the other reducibilities, the Q-degrees also have ties to algebra. Macintyre [50] showed that whenever the word problem of a group G was Turing below the word problem of a group H, then G must be a subgroup of every algebraically closed group of which H is a subgroup. The converse fails, but is true if we replace Turing reducibility with Q-reducibility, as shown by Belegradek [8]. That is, for any computably presented groups G and H, if G is a subgroup of every algebraically closed group of which H is a subgroup, then G's word problem must be Q-reducible to that of H. Thus any fact about the partial order of the c.e. Q-degrees has an immediate translation into one involving inclusion relations between the classes of finitely generated subgroups of algebraically closed groups. Since any countable algebraically closed group is determined up to isomorphism by the class of its finitely generated subgroups, this gives a natural relationship between a purely computability-theoretic notion and a purely algebraic one.

1.3 Post's Programme

1.3.1 First attempts: simple and hyper-simple sets

Post's programme was to find a thinness property of the c.e. sets which guaranteed incompleteness. It was motivated by the fact that *creative sets* have complements that are full of infinite c.e. sets, and this property guarantees m-incompleteness. The thought was that maybe a 'very thin' complement would work the same way for Turing incompleteness [68].

Creative sets are a large class of non-computable c.e. sets that includes the halting set. In fact, all creative sets are halting problems under some re-ordering of the partial computable functions. Every creative set contains (infinitely many) infinite c.e. sets in its complement, and a necessary and sufficient condition for a set S to be 1-complete (so every c.e. set is one-one reducible to S) is that S is infinite, and the complement $\overline{S}$ contains an infinite c.e. set. Thus in an effort to find a non-computable incomplete set, Post looked for sets that did not have this property:

Definition 7. A c.e. set A is *simple* if the complement $\overline{A}$ is infinite and contains no infinite c.e. sets. That is, if $|W_e| = \infty$ then $W_e \cap A \neq \emptyset$.

Notice that a simple set can't be computable, for if A is computable then $\overline{A}$ is computable, and thus $\overline{A}$ contains an infinite c.e. set, namely $\overline{A}$ itself. Further, a simple set is not 1-complete, since it does not contain any infinite c.e. sets in its complement. In fact, no creative set is one-one reducible to a simple set.

To see this, suppose we have a simple set A and a creative set C with $C \leqslant_1 A$, as witnessed by a one-to-one function f. Let g be a total computable function that enumerates W_e, one of the infinite c.e. sets that is contained in the complement $\overline{C}$. That is, $g(n) \notin C$ for every n, and so $f(g(n)) \notin A$. But now the infinite c.e. set $\{f(g(n)) : n \in \omega\}$ is entirely contained in the complement $\overline{A}$, contradicting that A is simple.

Having shown that simple sets exist, Post had two disjoint classes of non-computable sets. On the one hand, creative sets like the halting problem, that are 1-complete and have infinitely many infinite c.e. sets contained in their complement, and on the other hand simple sets, that are not 1-complete and do not have even a single infinite c.e. set in their complement. Post further showed that no creative set was wtt-reducible to a simple set. All this makes simple sets seem like a good candidate for a set that may give an intermediate degree. However, simple sets *can* in fact be truth table complete (and hence also Turing complete since $\leqslant_{tt}$ implies $\leqslant_T$). Post constructed for any creative set C a simple set S with $C \leqslant_{tt} S$. In particular, there is a simple set that has the same degree as K. Thus simple sets do not give the solution that Post was seeking of an intermediate degree.

To overcome this issue, Post next tried exploring sets which were simple in a stronger sense, replacing singletons with finite sets.

Definition 8. A c.e. set A is *hyper-simple* if $\overline{A}$ is infinite, and there is no infinite c.e. set of mutually exclusive finite sequences such that each sequence has at least one member in $\overline{A}$.

Another way of phrasing this definition is by considering disjoint canonical finite sets $D_z = \{x_1, \ldots, x_n\}$ given by some standard coding, for example $z = 2^{x_1}3^{x_2}\ldots p_n^{x_n}$. The exact coding is not important, the point is that we have a finite set and we know what it is from its index. For a computable function g_e, we call $V_e = \{D_{g_e(x)} : x \in \omega\}$ (with the property that the $D_{g_e(x)}$ are all disjoint) a *strong array.* Now A is hyper-simple if for every strong array V_e, there exists an x such that $D_{g_e(x)} \subseteq A$. That is, there is no strong array such that each finite set $D_{g_e(x)}$ has at least one member in $\overline{A}$. In particular, we don't have an infinite collection of canonical finite sets living in the complement $\overline{A}$.

The intuition for why this may be useful is as follows. Suppose we have a Turing reduction $\Gamma^A = X$, with A computing some set X. At some stage s we could have the computation on some element p be $\Gamma^{A_s}(p) = 0$, that is, $p \notin X_s$. But then if p later enters x, the computation needs to change to reflect this, and for this to happen, some element has to enter A below the use of the computation $\Gamma^{A_s}(p)$. This means that in order for the reduction to be correct, so for the computation to be able to recover when X changes, the reduction must be using things that are not currently in A to complete the computation $\Gamma^{A_s}(p)$. For if all the things which the computation used were already in A_s, then when X later changes, the reduction will be wrong. This is because even if the oracle A_s changes, that change won't affect the computation since the computation only uses things already in A at stage s and not any of the things which have changed. So if the reduction is correct, there must be a bunch of elements below the use of the computation that are not currently in A. These elements that are used in the computation look like a finite sequence, or a canonical finite set. The idea with a hyper-simple set is that there is at least one canonical finite set which is fully contained in A, which would correspond to a computation that can't change, or a tt-reduction that could fail. That is, the computational power of A is limited.

Indeed, Post showed that no hyper-simple set is truth table complete. However, there are still Turing complete hyper-simple sets. In fact, there are hyper-simple sets of every Turing degree. To construct a hyper-simple set with the same degree as a c.e. set B, we need a dump construction. This construction is described in Section 1.4.1.

Now if we apply the dump construction to $B = K$ we get a hyper-simple set that is Turing equivalent to the Halting set, and hence is complete. So hyper-simple sets also don't give the solution to Post's problem.

As an interesting aside, computability theory has another connection to classical mathematics (among many others not mentioned here) through hyper-simple sets. A consistent axiomatizable theory is not independently axiomatizable if and only if there is an enumeration $\{x_0, \ldots, x_k\}$ of it such that the set $\{n : x_{n+1}$ is deducible in first order logic from $x_0, \ldots, x_n\}$ is hyper-simple [60].

1.3.2 Hyper-hyper-simple and semirecursive sets

Post continued in his search for an intermediate degree with hyper-hyper-simple (hh-simple) sets, which now replace canonical finite sets with c.e. sets. The idea here is to capture the adaptive nature of Turing reductions in a notion akin to hyper-simplicity.

Definition 9. A set A is *hyper-hyper-simple (hh-simple)* if $\overline{A}$ is infinite and A meets all infinite weak arrays $V_e = \{W_{g_e(x)} : x \in \omega\}$ (where $W_{g_e(x)}$ are all disjoint). That is, there is an x for which $W_{g_e(x)} \subseteq A$.

When Post suggested hh-simple sets in [68], he did not know whether they even existed, but nevertheless hoped hh-simple sets were good candidates for an intermediate degree. Hh-simple sets are indeed interesting as potential candidates for Post's problem, because of a characterisation of hh-simple sets by Soloviev from 1974 [73]. This characterisation states that a co-infinite c.e. set is hh-simple if and only if it is not contained in any Q-complete set. In particular, hh-simple sets cannot be Q-complete themselves. This still leaves open the questions of whether hh-simple sets even exist, and if they do exist, whether they can be Turing complete.

Lachlan [48] proved another characterisation of hyper-hyper-simplicity in 1968: a set A is hh-simple if and only if its lattice of supersets $L^*(A)$ is a Boolean algebra. The lattice of supersets is defined as $L^*(A) = \{W_e : W_e \supseteq^* A\}$, under intersection and union. The relation $=^*$ and $\subseteq^*$ are equality and containment up to a finite difference. That is, if $A \subseteq^* W_e$, there are at most finitely many things that are in A but not W_e, and similarly with $A =^* B$, there are at most finitely many things in $A \setminus B$ and in $B \setminus A$. Note that $=^*$ is an equivalence relation.

With Lachlan's characterisation of hh-simple sets, we can consider those hh-simple sets whose lattice of supersets is the simplest possible Boolean algebra, the two element Boolean algebra. These are called *maximal sets*, and they have the property that for any W_e where $A \subseteq W_e$, either $W_e =^* A$ or $W_e =^* \omega$. That is, maximal sets have extremely thin complements: no c.e. set (that contains A) splits the complement $\overline{A}$, because if W_e contains A it either contains almost all of the complement of A (all but finitely much), or it only contains finitely much of $\overline{A}$. That is, W_e can never have infinitely many things from the complement of A while also leaving out infinitely many things. Maximal sets were first introduced by Myhill to try to solve Post's problem, following Post's intuition of making sets with very thin complements [60].

Friedberg [29] constructed a maximal set in 1958 using a priority argument, showing that hh-simple sets do indeed exist. We present a construction of a maximal set using the priority method in Section 1.4.3. The priority method was pioneered independently by Friedberg [28] and Muchnik [58] in order to solve Post's problem, and has since been adapted to solve many other problems. Friedberg and Muchnik did indeed each succeed in solving Post's problem, constructing non-computable c.e. sets each of which is not reducible to the other, and thus getting a set whose degree was strictly between $\mathbf{0}$ and $\mathbf{0}'$. However, this wasn't entirely in the spirit of Post's programme, which asked for a structural property that would guarantee incompleteness, and so various mathematicians continued looking for a structural property solution to Post's problem.

Now that we know maximal sets exist, all that's left is to determine if maximal sets, with their thin complements and Q-incompleteness, are also Turing incomplete. However, in 1963, Yates [77] constructed a maximal set with degree $\mathbf{0}'$.

The next attempt was to find a property that would reduce being Turing complete to being Q-complete, which could be combined with maximality to give a set of an intermediate degree. In order for a computation to change in a Turing reduction, something small (below the use of the computation) needs to enter the set which is used as the oracle in the computation. However, there could be many things below the use that could potentially enter the oracle, so we need to pay attention to a finite set of elements, and if any one of these elements enters, the computation might change. On the other hand with a Q-reduction, we only point at a single element (with the Q-like function), and if this single element enters then the computation might change. So to reduce being Turing complete to being Q-complete, we would need a property that turns questions about a finite set into questions about a single element. One property to consider is semirecursiveness [60].

Definition 10. A set A is *semirecursive* if there is a computable function $f(x, y)$ with the following two properties for all x and y:

$$(i) \quad f(x,y) = x \text{ or } f(x,y) = y;$$

$$(ii) \quad \text{if } x \in A \text{ or } y \in A, \text{ then } f(x,y) \in A$$

Semirecursiveness essentially picks out a preferred element between each pair of elements, in the sense that if x and y are not in the set A, and say $f(x, y) = x$, then if y enters A, x must also enter A. This is because by having y enter A, we now fulfill the premise in part (ii) of the definition, so $f(x, y) = x$ must also enter A. We can extend this to a finite set $\{x_0, \dots, x_n\} \subseteq \overline{A_s}$ by computing $f(x_i, x_j)$ for each pair in the set, and through that find the most preferred element x_k in the set, such that if *anything* in the set enters A (after stage s), x_k must also enter A.

Marchenkov showed in 1976 [53] that if A is a c.e. set, B is a semirecursive c.e. set and $A \leqslant_T B$, then $A \leqslant_Q B$. For c.e. sets we have that $\leqslant_Q$ implies $\leqslant_T$, so in fact this says that if B is semirecursive then $A \leqslant_T B$ if and only if $A \leqslant_Q B$. In particular, if A is not Q-complete then A also isn't Turing complete; so semirecursiveness is exactly the desired property. Marchenkov's result is not too difficult to see, for a sketch see Section 1.4.2.

Since hh-simple sets are not Q-complete, semirecursive hh-simple sets would provide the solution to Post's problem, because the hyper-hyper-simplicity ensures the set is not Q-complete, and the semirecursiveness lets us translate that into being not T-complete. All that remains is to construct such a set. Unfortunately, no hh-simple set is semirecursive. In fact, in 1992 Cholak, Downey and Stob [14] proved that no property of the lattice of supersets alone can guarantee incompleteness (recall that hh-simplicity is the same as the lattice of supersets being a Boolean algebra). Around the same time in 1991, Harrington and Soare [34] did find a first order property that guaranteed incompleteness and non-computability (any set satisfying this property is Turing incomplete and not computable), and further showed that there were c.e. sets that satisfied this property. This property is a 4 quantifier statement which was obtained from analysing how the 'automorphism machinery' fails when one tries to prove that all c.e. non-computable sets are automorphic to complete sets.

1.3.3 Equivalence relations

Another approach would be to try to loosen the definition of hyper-hyper-simplicity in a way that allows for a set to be semirecursive, while maintaining that the set is Q-incomplete. This approach replaces equality with a c.e. equivalence relation. In 1971, Ershov [22] introduced c.e. equivalence relations ('Positive equivalences'), encouraging their study and illustrating some of their potential. In particular, certain properties of c.e. equivalences have non-trivial consequences for sets closed under these equivalences, and many classical concepts about c.e. sets can naturally be extended to c.e. equivalences.

An equivalence relation $=_\eta$ is a *c.e. equivalence relation* if the set of η-equivalent pairs, $\{< x, y >: x =_\eta y\}$, is computably enumerable. This means that equivalence classes can grow over time, as we find out that more things are η-equivalent. We will consider only those c.e. equivalence relations that have infinitely many equivalence classes. It will soon become apparent that if there are only finitely many equivalence classes then the equivalence relation is not helpful for our purposes. For a c.e. equivalence relation $=_\eta$, we say a set A is *η-closed* if it consists entirely of η-equivalence classes, so the set doesn't split up any equivalence class. That is, for all x and y, if $x \in A$ and $x =_\eta y$ then $y \in A$. Further, A is called *η-finite* if it consists of only finitely many equivalence classes (so all η-finite sets are η-closed).

As Ershov [22] noted, we can relativise a large number of concepts to c.e. equivalence relations, for sets that are closed under said equivalences. In particular, we can use η-finiteness instead of finiteness to relative simplicity, hyper-simplicity, maximality and so on. This is why we want our equivalence relation to have infinitely many equivalence classes.

For example, a non-computable η-closed c.e. set A with $\overline{A}$ being η-infinite is *η-simple* if every η-closed c.e. subset of $\overline{A}$ is η-finite. A is *η-hyper-simple* if for every sequence of finite sets $\{F_{g_e(x)} : x \in \omega\}$ (for g_e computable) such that $[F_i]_\eta \cap [F_j]_\eta = \emptyset$ whenever $i \neq j$ (i.e. the equivalence classes of the finite sets are all disjoint), there is an x for which $[F_{g_e(x)}]_\eta \subseteq A$. A is *η-maximal* if for any c.e. set B that is η-closed and $B \supseteq A$, either $B \setminus A$ is η-finite or $\omega \setminus B$ is η-finite. Similarly we can naturally relativise many other notions, and a lot of properties are preserved when we do this, such as the fact that η-maximal sets are all η-hh-simple. However, not everything gets transferred over. For instance, η-simple sets are not necessarily non-computable if we don't include that in the definition, while all simple sets certainly are non-computable, without that needing to be part of the definition [22].

Most importantly, Marchenkov showed in 1976 [53] that η-hh-simple sets are all Q-incomplete. Once again we can combine this with semirecursiveness to get that all η-hh-simple sets are Turing incomplete. In 1973, Degtev constructed a c.e. equivalence relation η for which η-maximal semirecursive non-computable sets exist [60]. That is, the class of η-maximal semirecursive non-computable sets is not empty, and by Marchenkov's results, is Turing incomplete. At last, Post's problem has been solved using the methods suggested by Post's programme.

1.4 Proof Sketches

In this section we describe the constructions and proof sketches that were omitted from the previous section.

1.4.1 Dump construction

We describe the dump construction for making a hyper-simple set A that is Turing equivalent to a given c.e. set B.

Let $f(\omega) = B$ be a one-to-one computable enumeration of the c.e. set B, and construct A in stages with $A_0 = \emptyset$. Then let $\{a_{0,s}, \ldots, a_{s,s}\}$ be the first $s+1$ things in the complement of A_s, and let $A_{s+1} = A_s \cup \{a_{f(s),s}, \ldots, a_{s,s}\}$. That is, when $f(s)$ enters B (at stage s), we dump the $f(s)$-th thing in the complement of A into A, together with all things after, up until $a_{s,s}$. Then for $i < f(s)$, $a_{i,s+1} = a_{i,s}$, while $a_{f(s),s+1}, \ldots, a_{s+1,s+1}$ are the next elements in the complement of A_{s+1}.

Then $A \equiv_T B$ and A is hyper-simple. To compute A from B, go to a stage s where $B \upharpoonright n$ has stopped changing, that is, $B_s \upharpoonright n = B \upharpoonright n$. Now $A_s \upharpoonright a_{n,s} = A \upharpoonright a_{n,s}$, because the n-th thing in the complement of A at stage s, $a_{n,s}$, will only change if something below n enters B. Thus if nothing below n ever enters B hereafter, the first n things in the complement of A must have stopped changing too. Similarly, to compute B from A, wait for a stage s where $a_{n,s}$ is final, so $a_{n,t+1} = a_{n,t}$ for all $t \geqslant s$. That is, the first n things in the complement of A have stopped changing. Now $B_s \upharpoonright n = B \upharpoonright n$. To see that A is hyper-simple, suppose for a contradiction that we have a strong array V_e such that for every n, $D_{g_e(n)} \not\subseteq A$. Then we claim B is computable. To compute $B \upharpoonright z$, run the enumeration of V_e until we find an n such that for every $p \in D_{g_e(n)}$ which is not in A_s, $p > a_{z,s}$. We assume, as is standard, that everything is bounded by the stage number s (if this is not the case we can slow things down until it is so). In particular, $\max\{r : r \in D_{g_e(s)}\} < s \leqslant a_{s,s}$. We will at some stage s find our desired n because each canonical finite set $D_{g_e(n)}$ has elements that are not in A, and the complement of A at stage s, that is, the elements $a_{i,s}$, eventually stop changing. Further, the sets $D_{g_e(n)}$ are all disjoint, so we must have canonical finite sets with elements that are arbitrarily large, in particular greater than $a_{z,s}$. Now $B_s \upharpoonright z = B \upharpoonright z$, because if B were to change below z, say at $w < z$, then we would dump $a_{w,s}, \ldots, a_{z,s}, \ldots, a_{s,s}$ into A, and in so doing, make $D_{g_e(n)} \subseteq A$ which contradicts our assumption. Thus A must be hyper-simple.

1.4.2 Semirecursiveness

The following is a sketch of Marchenkov's result that if A is a c.e. set, B is a semirecursive c.e. set and $A \leqslant_T B$, then $A \leqslant_Q B$ [53].

Suppose B is semirecursive via f, and $A \leqslant_T B$ via Φ. To define a Q-like function $m(x, s)$, consider the simultaneous enumerations of A and B and the computations of Φ^B. In particular, at each stage s we want to keep track of how much of A_s agrees with the computation Φ^{B_s}. Thus we introduce the following function.

Definition 11. The *length of agreement* at stage s is $\ell(s) = \max\{z : \forall x \leqslant z,\ A_s \upharpoonright x = \Phi^{B_s} \upharpoonright x\}$

The length of agreement is the length on which the enumeration of A_s agrees with the current computation Φ^{B_s}. We will use various length of agreement functions throughout this thesis, and they always measure how much the relevant computations and/or enumerations agree with each other. The exact function will depend on the situation at hand, although we don't always explicitly write them out. In essence they are all the same as the length of agreement just defined, though they may include more sets/functions/Turing functionals. Now, assuming that $\Phi^B = A$, the length of agreement must go to infinity, that is $\lim_s \ell(s) = \infty$.

To define our Q-like function on an element x, we wait for the length of agreement to go above x, so $\ell(s) > x$, and compute $\Phi^{B_s}(x)$. If $\Phi^{B_s}(x) = 1$, then $x \in A_s$ and we can define $m(x, s)$ to be an element already in B. Otherwise, $\Phi^{B_s}(x) = 0$ and $x \notin A_s$. Consider pairwise all the elements below the use of this computation that are not in B_s. That is, elements $y, z \in \mathbb{N} \upharpoonright \varphi(x)[s] \setminus B_s$, where φ is the use function of Φ. If $\Phi^{B_s}(x)$ is to change and x is to enter A, then one of these elements below the use must enter B. Compute $f(y, z)$ for all such pairs, and in so doing determine which element y_0 is the most preferred element. Define $m(x, s) = y_0$. If something below the use enters B at stage $s' > s$, y_0 must enter B too, say at stage $t > s$. It is possible that at this stage the length of agreement has dropped below x, in which case we wait for a stage $t' \geqslant t$ where $\ell(t') > x$ again. If $x \in A_{t'}$, we don't need to do anything; m is correct and need never change again. Otherwise, m needs to retarget to an element not in $B_{t'}$ (that is, we define $m(x, t') \neq m(x, s)$ to be an element not in $B_{t'}$). To find this element, as before we compute $f(y, z)$ on all pairs $y, z \in \mathbb{N} \upharpoonright \varphi(x)[t'] \setminus B_{t'}$ and find the most preferred element (it is possible the use has changed, since we have a different computation now). Eventually we will reach a stage r where $B_r \upharpoonright \varphi(x)[r] = B \upharpoonright \varphi(x)[r]$ (i.e. B has stopped changing below the use of the computation, and so correspondingly the computation has stopped changing). Thus we know that $\lim_s m(x, s) = m(x)$ exists; either at some point x enters A and so m points at an element in B and never changes, or $x \notin A$ and $m(x, r) \notin B$ and $m(x) = m(x, r)$. Following this process for every x, we have a Q-reduction $A \leqslant_Q B$.

1.4.3 The priority method: maximal sets

The priority method which was developed independently by Friedberg [28] and Muchnik [58] to solve Post's problem is extremely useful, and has been used for many proofs about c.e. sets and degrees. Many results in this thesis use the priority method in various incarnations. Priority constructions are used for making sets with particular properties. Here we use the example of being maximal to illustrate the ideas and thought process behind a priority argument.

Theorem 12. *Maximal sets exist.*

Proof. The main idea of a priority argument is to break up a property, in this case maximality, into a countable sequence of *requirements*, so that if we satisfy every requirement during our construction, then our set will have the desired property. The individual requirements are easier to deal with than the property itself, and each one takes care of a small part of the property. For maximality, we need the following requirements to be satisfied by the set A we are constructing, in order for A to be maximal.

$$P_e : \text{ if } (W_e \supseteq A \text{ and } |W_e \setminus A| = \infty) \text{ then } W_e =^* \omega.$$

Each P_e requirement cares only about one set, W_e, and if W_e doesn't contain A then it is satisfied. If W_e does contain A but $W_e =^* A$, so there are only finitely many things in W_e that are not in A, then again P_e is met (satisfied). Otherwise, in order to satisfy P_e we need to ensure that $W_e =^* \omega$, and the only power we have to ensure this is to put things into A, since A is the set that we are constructing (then if W_e is to still contain A, it must also have these elements in it, which is how we can force it to be close to ω). Then if *every* requirement is met, the set A we are constructing will be maximal. The P requirements are called *positive* requirements because they want to put elements into A: when P_e sees there are lots of things in W_e that are in the complement of A, it will want to put things into A to ensure that either almost all of the complement is in W_e, or almost all of it avoids W_e.

We construct our set A in stages, and at each stage we may enumerate elements into A. Then provided each step in the construction is computable, this makes A computably enumerable. Because for each P_e we care about how much of the complement of A is in W_e, we will keep track of the complement of A by having markers $a_{n,s}$ for the n-th (smallest) thing in the complement of A_s. We want $\overline{A}$ to be infinite, so we don't want our markers for the elements in the complement to keep changing infinitely many times, which gives us another set of requirements to fulfill.

$$N_e : \lim_s a_{e,s} = a_e \text{ exists.}$$

These requirements are called *negative* requirements because they want to keep elements out of A: N_e wants to keep $a_{0,s}, \ldots, a_{e,s}$ out of A so that $\lim_s a_{e,s} = a_e$ exists, that is, the e-th thing in the complement of A stops changing.

In order to understand how we will meet all of our requirements, we first consider only P_0 in isolation.

Basic Strategy for $\mathbf{P_0}$ and $\mathbf{N_0}$: Recall that P_0 wants either almost all of the complement of A in W_0, or almost all of it outside of W_0. Suppose that at stage s, some element $a_{p,s}$ enters W_s (that is, an element that is currently in $\overline{A}$). One potential strategy would be to put $a_{p,s}$ into A_{s+1}. Certainly if we put into A everything from the complement which goes into W_0, then we will have $W_0 \setminus A = \emptyset$, and P_0 is satisfied. However, if we do this it is possible we will end up putting *everything* into A, which we certainly don't want! Instead what we can do is put $a_{0,s}, \ldots, a_{p-1,s}$ into A_{s+1}. Now $a_{0,s+1} = a_{p,s}$, and we say that the 0-*state* of $a_{0,s+1}$ has changed from (0) to (1). We use e-states to keep track of which elements in the complement are in which c.e. sets. In general, the e-state of an element a at stage s is a binary sequence with $e+1$-many elements, that indicates if a is or is not an element of the c.e. sets $W_{0,s}, W_{1,s}, \ldots, W_{e,s}$. That is, the 0-state of $a_{0,s}$ being (0) means that $a_{0,s} \notin W_{0,s}$, and the 0-state of $a_{0,s+1}$ being (1) means that $a_{0,s+1} \in W_{0,s+1}$. So initially, every $a_{i,s}$ is in 0-state (0). Now we want to keep $a_{0,s+1}$ out of A, so that $a_0 = \lim_s a_{0,s}$ exists N_0 is satisfied. At a later stage $t > s+1$, suppose we see $a_{n,t}$ enter W_t. We put $a_{1,t}, \ldots, a_{n-1,t}$ into A, so that $a_{1,t+1} = a_{n,t}$, and declare $a_{1,t+1}$ to be in 0-state (1). Whenever a new element a from the complement enters W_0 we do the same thing again: put all elements below a that have 0-state (0) into A, and a now has 0-state (1) and so will be kept out of A. Notice that under this scheme, P_0 changes $a_{e,s}$ at most $e+1$ many times.

There are two possibilities. First, that infinitely many things from the complement enter W_0. In this case we will have infinitely many things in the complement of A, all of which have 0-state (1). That is, they are all in W_0. Now P_0 is satisfied, because all of $\overline{A}$ is in W_0, so if W_0 contains A then $W_0 = \omega$. The other case is that this does not happen infinitely many times. That is, after some stage s, no element $a_{n,s}$ in the complement of A_s ever enters W_0. Then there are only finitely many things in $\overline{A}$ that are also in

W_0, namely those elements that entered W_0 before stage s. Everything else which is in $\overline{A}$ is also in $\overline{W_0}$, and so if $W_0 \supseteq A$, then $W_0 =^* A$ and P_0 is again satisfied. In this scenario, all but finitely many of the elements in $\overline{A}$ have 0-state (0).

Interactions between requirements. So far we have considered P_0 in isolation, as if it is the only requirement in existence. However, we also need to meet all other P requirements at the same time. Consider for example P_1. This requirement wants to have the same strategy as P_0, and do the same thing when it sees elements from the complement of A entering W_1. If we just let both requirements ignore each other and act as they wish, it is possible we will end up putting everything into A. For instance, suppose P_0 has seen $p-1$ elements from the complement enter W_0, so now $a_{0,s}, \ldots, a_{p-1,s}$ all have 0-state (1), and P_0 is happy to keep them out of A. Now suppose that $a_{p,s}$ enters W_1, and this is the first time something from the complement enters W_1. Then P_1 wants to put all of $a_{0,s}, \ldots, a_{p-1,s}$ into A, and make $a_{0,s+1} = a_{p,s}$. If this kind of thing keeps happening, alternating between P_0 and P_1, eventually everything will get put into A!

Priority. To overcome this issue, instead of just letting each requirement act independently and ignore all other requirements, we will assign a different *priority* to each requirement. We will say that P_i has *higher priority* than P_j if $i < j$. In this way, P_0 is the highest priority requirement, and so can act as described in the basic strategy, but any lower priority requirement has to take into consideration the actions of all higher priority requirements (of which there are finitely many). Also, we will only allow P_0 to change a_0, and only allow P_0 and P_1 to change a_1, and in general only P_i for $i \leqslant e$ can change a_e. Now only finitely many P_i can change each $a_{e,s}$, so if we ensure that each P_i changes $a_{e,s}$ at most finitely many times, then $\lim_s a_{e,s}$ exists and N_e is satisfied.

In order for P_e to keep track of what higher priority requirements are doing, the e-state of an element $a_{i,s}$ will be a tuple that keeps track of which W_n, $n \leqslant e$, $a_{i,s}$ is in. For example, if $a_{3,s}$ is in $W_{0,s}$ and $W_{3,s}$ but not in

$W_{1,s}$ nor $W_{2,s}$, its 3-state at stage s will be $(1, 0, 0, 1)$, its 2-state at stage s is $(1, 0, 0)$ and so on, and for $e > 3$, P_e is not allowed to change $a_{3,s}$ (it cannot put any of $a_{0,s}, \dots, a_{3,s}$ into A). Then the strategy for each P_e is to try to maximise the e-states of those elements a_i which P_e is allowed to change, so that almost all elements in $\overline{A}$ have the same e-state.

Since each P_e keeps track only of what higher priority requirements are doing, P_e is in a sense blind to the requirements below it, those with lower priority. This means that a lower priority requirement might have to take action multiple times for the same a_i. For example, suppose P_1 has seen 5 things go into W_1 (and thus far nothing has entered W_0), so P_1 has acted 5 times. As a result, at stage s, $a_{1,s}, \dots, a_{5,s}$ all have 1-state $(0, 1)$. Then perhaps at stage $t > s$, some large $a_{n,t}$ enters W_0, so P_0 takes action and puts $a_{0,t}, a_{1,t}, \dots a_{n-1,t}$ into A. Now $a_{1,t}, \dots, a_{5,t}$ are back to having 1-state $(0, 0)$, and so P_1 will want to take action for these elements again, should it see more things from the complement of A go into W_1. We say that P_1 has been *injured*, because the action of P_0 has destroyed the work of P_1, causing P_1 to need to start again.

The point of having a priority ordering is that we allow higher priority requirements to injure lower priority requirements, but not the other way around. If we don't have a priority ordering, and we didn't allow any requirement to injure any other requirement we would get stuck and won't be able to satisfy all the requirements, but if we allowed any requirement to injure any other requirement, we would once again be in trouble. For example, it could be that $a_{1,s}$ has 1-state $(1, 0)$, and then $a_{n,s}$ enters $W_{1,s}$. Now P_1 acts and puts $a_{1,s}, \dots, a_{n-1,s}$ into A, so now $a_{1,s}$ has 1-state $(0, 1)$. Then perhaps $a_{p,t}$, $t > s$ enters $W_{0,t}$ and P_0 takes action again, putting $a_{1,t}, \dots, a_{p-1,t}$ into A, turning the 1-state of $a_{1,t}$ back into $(1, 0)$. If this happens infinitely many times, we end up with everything except for a_0 in A! Having a priority ordering and only allowing higher priority requirements to injure lower priority requirements is a middle ground that lets us eventually satisfy every requirement, even though in the process of satisfying one requirement, we may cause another (lower priority) requirement to have to start over.

We can force requirements to respect higher priority requirements (i.e. not injure them) by insisting that a requirement P_e can only act to *increase* the e-state of an element $a_{i,s}$, where the e-states are ordered lexicographically (so for example with 2-states, $(0,0) <_L (0,1) <_L (1,0) <_L (1,1)$ and $(1,1)$ is the highest 2-state).

We say that P_e *requires attention* at stage s if there is an $a_{i,s} \in W_e$ and P_e can increase the e-state of some $a_{j,s}$, $e \leqslant j < i$, by making $a_{j,s+1} = a_{i,s}$ (by enumerating $a_{j,s}, \dots, a_{i-1,s}$ into A, as in the basic strategy).

Construction. We start with $A_0 = \emptyset$. At stage s, find the highest priority P_e that requires attention at stage s. Let j be the smallest such that P_e can increase the e-state of $a_{j,s}$, and let $a_{n,s} \in W_{e,s}$ be the (smallest) element by which P_e can do this. Enumerate $a_{j,s}, \dots, a_{n-1,s}$ into A_{s+1}. Repeat this until there are no more P_e that require attention at stage s.

Verification. We now check that our construction does indeed yield a maximal set, that is, all of our requirements are met.

Lemma 13. *All the N_e requirements are met.*

Proof. Each element $a_{e,s}$ in $\overline{A_s}$ can only be changed by P_i for $i \leqslant e$, so all changes to $a_{e,s}$ will be reflected by a change in the e-state of $a_{e,s}$. There are finitely many e-states, and the actions of P_i for $i \leqslant e$ will only ever increase the e-state of $a_{e,s}$. Thus the e-state of $a_{e,s}$, and consequently $a_{e,s}$ itself, only change finitely many times. Then $\lim_s a_{e,s}$ exists and N_e is satisfied. □

Lemma 14. *All the P_e requirements are met.*

Proof. Suppose not. Let e be the smallest such that P_e is not met. That is, $A \subseteq W_e$ and $|W_e \setminus A| = \infty$, but also $|\omega \setminus W_e| = \infty$. For this to happen, there are infinitely many elements in the complement of A that are in W_e, and infinitely many that are not. Let n be such that $a_n \notin W_e$. Go to a stage s_0 where all P_i for $i < e$ have finished acting on a_{j,s_0} for $j \leqslant n$, and $a_{n,s_0} = a_n$. There are infinitely many elements in $W_e \setminus A$, so there are infinitely many stages $s_1 > s_0$ where we see a new $a_{j,s_1} \in W_{e,s_1}$ for some $j > n$. If P_e requires attention at stage s_1, the smallest element a_{i,s_1} whose e-state can be increased by P_e's action cannot be a_{n,s_1}, nor can it be smaller than a_{n,s_1}, otherwise P_e will enumerate a_{n,s_1} into A_{s_1+1}, contradicting either that $W_e \supseteq A$ or that $a_n \notin W_e$. This means that a_{j,s_1} is not in W_{i,s_1} for some $i \leqslant e$ for which $a_{n,s_1} \in W_{i,s_1}$, so that shifting a_{j,s_1} down to a_{n,s_1+1} would decrease the e-state of a_n. Further, a_{j,s_1} never enters W_i, for if it does then at the stage t when it enters, P_e will act and enumerate a_{n,s_1} into A_{t+1}. If P_e does not require attention at stage s_1, that has to be because there is no element in $\overline{A_{s_1}}$ below a_{j,s_1} whose e-state can be increased by P_e. Since a_{n,s_1} has an e-state ending in 0 (so not the highest possible e-state), this means that again a_{j,s_1} is not in W_i for some $i \leqslant e$ for which $a_{n,s_1} \in W_i$.

Infinitely often we have an $a_{n,s} \in W_{i,s}$ for some $i \leqslant e$ and a larger $a_{j,s} \notin W_i$. There are only finitely many $i \leqslant e$, so for (at least) one of these W_i, there are infinitely many such elements. That is, there are infinitely many elements from the complement of A both in W_i and not in W_i, so in fact P_i is not met. This contradicts our choice of e, thus all P_e must have been satisfied. $\square$

All of the requirements have been satisfied, and so the constructed set A is indeed maximal. $\square$

The priority method is very useful and can be adapted to many different situations. In general, priority arguments can be much, much more complicated than this example, and they can also be simpler. It's possible in some priority arguments that the requirements don't interact with each other at all (do not cause injury to other requirements), and can each simply act independently. In this case all we need to do is reserve a part of the set or space that we are constructing for each requirement to freely take action in. It could also be that the requirements only need to take a finite number of actions before being satisfied forever. For more complicated priority arguments, various different techniques and models have been developed over the years to ensure all the requirements are met. For example, a tree of strategies can be used for more complicated priority requirements to have multiple versions of each requirement, when there are multiple ways a requirement can act depending on if a certain condition is met or not. Here, each version of a lower priority requirement 'guesses' how all higher priority requirements will act, and has a strategy based on these guesses. Then a requirement is met if some version of it on the tree is met. Another useful model is the pinball machine, which forces elements (that positive requirements want to enumerate) to pass through every higher priority negative requirement in turn before being enumerated. The priority method is used extensively throughout this thesis.

Chapter 2

Polish Groups

Sometimes, the combinatorics of specific structures cannot be separated from the problem at hand in the way Post suggested doing. Regardless, techniques that were developed for studying degree structures can still be used in applied contexts, for instance priority arguments. Questions about effective processes (i.e. computability) have been asked in many areas of mathematics, including analysis, algebra, logic, and so on.

For instance, a significant portion of Turing's 1936 paper [75] (about Turing machines) was focused on computable reals. Turing notes that *'although the subject of this paper is ostensibly the computable numbers, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth'* [75]. So while Turing focused on computable numbers, his techniques could be used for computable functions just as well. Around the 1950s this was picked up by Markov, Zaslavskii, Ceitin and others, and in fact nowadays Turing's notion of computable functions is more commonly known as Markov computability [18]. Much of classical elementary real analysis can be effectivized using Markov computability. In the mid 1950s Grzegorczyk and Lacombe laid out the foundations of computable analysis in detail. A lot of the work at this time was essentially on computable calculus [18].

Several different traditions and approaches arose independently in computable analysis. Banach and Mazur introduced sequential computability for functions on computable reals in 1937, which is more general than Markov computability. That is, all Markov computable functions are sequentially computable, but the converse is not always true (so some sequentially computable functions are not computable by a Turing machine) [18]. Another notion is Borel computability, also defined for functions on computable real numbers, which was shown to be equivalent to Markov computability in 1957 by Kriesel, Lacombe and Shoenfield [47] and in 1959 by Ceitin [10].

One key notion is typically attributed to Kleene [44] from 1952, although many other equivalent definitions were introduced around the same time. Kleene computable functions are not restricted to the computable reals only, and are equivalent to effectively continuous functions, one form of which was introduced by Grzegorczyk and Lacombe. Essentially, Borel computability corresponds to effective continuity restricted to the computable reals. On the other hand, Markov computable functions are not necessarily continuous, and so Markov computability is *not* equivalent to Kleene computability, as Kleene computable functions are necessarily continuous. Further, Specker [74] showed in 1949 that even continuous Markov computable functions are not necessarily Kleene computable.

Many other notions for computable real-valued functions, for example Lacombe-Grzegorczyk computability (introduced independently in 1955 by Lacombe and 1957 by Grzegorczyk) and uniformly computable functions (introduced by Caldwell and Pour-El in 1975) have also turned out to be equivalent to Kleene computability. With so many definitions arising in computable analysis, it was important to know which definitions were equivalent, which were stronger or weaker, and so on, in order to make sense of the existing literature on the subject.

Similarly in effective algebra, we saw that various notions for effective presentability of algebraic structures emerged by the early 1960s, with the most well-studied being c.e. presentations, co-c.e. presentations and computable presentations. As with computable functions, some key results early on to characterise and separate these notions in various classes of algebraic objects. For example, Mal'cev [52] in 1962 gave a characterisation of computable presentability for the groups G_S, where G_S is the subgroup of $(\mathbb{Q}, +)$ generated by $\{1, \frac{1}{p_i^n} : \langle i, n\rangle \in S\}$, and $\langle i, n\rangle \in S$ implies $\langle i, k\rangle \in S$ for all $k \leqslant n$. He found that G_S is computably presentable if and only if S is computably enumerable. As well as the Novikov and Boone result about undecidable word problems in finitely presented groups, another example in group theory comes from Khisamiev [42], who showed in 1986 that every c.e. presented torsion-free abelian group has a computable presentation. For Boolean algebras, some key examples include Feiner's result from around 1968 [23, 24] that there is a c.e. presented Boolean algebra that does not have a computable presentation (i.e. is not isomorphic to any computable Boolean Algebra), and the result of Downey and Jockusch from 1994 [16] that every low Boolean algebra has a computable presentation. The presentability of other standard classes have also been studied and characterised by Metakides and Nerode in 1979 [56], and many others [3, 21, 32].

Simultaneously and independently, multiple different notions of effective presentability for separable structures were being used by the early 1960s (recall the definitions by Ceitin and Moschovakis for Polish spaces). As with the rationals in the reals, there is a large class of topological spaces with computable countable dense subsets. Generally, many (though not all) of the natural separable spaces that arise classically in the literature have computable presentations in this sense. There was also some early work on computable metric spaces, for example by Ceitin in 1959 [10], but among separable structures, the theory around computable Banach spaces and computable compact spaces is more developed [18].

However, in contrast with effective analysis and algebra, the notions of effective presentability for separable spaces (that is, left-c.e. presentations, right-c.e. presentations and computable presentations) have only been compared very recently, in the last several years [35, 36, 49, 54]. For instance, it was shown in 2023 that there are right-c.e. Polish spaces not homeomorphic to any computable Polish space [6], and left-c.e. presented Polish spaces that are not computable Polish, up to homeomorphism [54]. In a similar vein, this chapter focuses on the following question.

Question 1. Does there exist a Polish space that is both left-c.e. and right-c.e. presentable but not computably presentable?

The question changes based on the type of isomorphism under which the structures are considered. For instance, one can study them up to quasi-isometry, isometry, bi-Lipschitz maps, homeomorphism, and so on. Considering the structures up to isometry, it is not too hard to construct a Polish space that is left-c.e. (or right-c.e.) but not computable. Simply take the space $[0, \alpha]$ with the Euclidean metric for some non-computable left-c.e. real α. The same can be done for the right-c.e. case by taking α to be a right-c.e. real that is not computable. Observe, however, that this would not suffice to answer Question 1 up to isometry, as a real number that is both left-c.e. and right-c.e. is computable. Instead, to answer the question up to isometry, we encode a Δ^0_2 set into a Polish space, and show that the space is both left-c.e. and right-c.e. presentable. Further, we show that this space is computably presentable if and only if the encoded Δ^0_2 set is limitwise monotonic. Since there are Δ^0_2 sets that are not limitwise monotonic [41, 43], in Theorem 16 we conclude that there is a space which answers Question 1 (up to isometry) in the affirmative. Moreover, Theorem 16 is witnessed by a discrete space. A totally disconnected example that works up to homeomorphism can be found in the recently submitted [46]. We seek an example which is neither discrete nor totally disconnected. Extending the technique in Theorem 16, we prove the following.

Theorem 21. *There is a compact connected Polish space that has a left-c.e. presentation and a right-c.e. presentation but does not have a computable presentation, up to isometry.*

Another related area of study is the recently emerged theory of effectively presented Polish groups. Following a similar pattern seen in effective algebra [45, 49], an effective presentation of a Polish group is defined to be an effective presentation of the Polish domain, upon which the (group) operations can be effectively computed. For instance, the domain of a Polish group can be computable, left-c.e. or right-c.e. Polish. Further, we need the operations to be effective in some sense. We examine this in detail in Section 2.2, where we see that two common notions for the effectiveness of the operations differ in general. Interestingly, the choice of these generally non-equivalent definition turns out to not affect our second main result. Recently, it was shown that there is a right-c.e. Polish group that is not topologically isomorphic to any computably presented Polish group [45]. We answer Question 1 for Polish groups up to topological isomorphism.

Theorem 37. *There is a Polish group G that has a left-c.e. presentation, a right-c.e. presentation, but no computable presentation, up to topological isomorphism.*

In order to prove Theorem 37, we develop techniques that allow us to reduce this question to a question about discrete groups. More specifically, we establish a one-to-one correspondence between topological and effective algebraic presentations of certain groups.

This chapter is organised as follows. Section 2.1 explores effective presentations of metric spaces and contains the proof of Theorem 21. Section 2.2 compares the notions of effectively continuous and sequentially computable operations on left-c.e. presented Polish groups, and establishes the aforementioned correspondence. Finally, Section 2.3 contains the proof of Theorem 37.

2.1 An Almost Computable Metric Space

To begin, we recall some definitions for presentations of Polish spaces.

Definition 15. Let $\mathcal{M} = (M, d)$ be a Polish space. $X = ((\alpha_i)_{i\in\mathbb{N}}, d)$ is a *computable presentation* of $\mathcal{M}$ if:

- $(\alpha_i)_{i\in\mathbb{N}}$ is dense in M, and

- $d(\alpha_i, \alpha_j)$ are reals uniformly computable in i, j.

Similarly X is a *left-c.e. presentation* (or lower semi-computable presentation) of $\mathcal{M}$ if $d(\alpha_i, \alpha_j)$ are uniformly left-c.e. reals, and it is a *right-c.e. presentation* (or upper semi-computable) if $d(\alpha_i, \alpha_j)$ are uniformly right-c.e. reals.

To this end, we view all our spaces up to isometry.

Theorem 16. *There exists a discrete Polish space that has a left-c.e. presentation and a right-c.e. presentation but does not have a computable presentation, up to isometry.*

Proof. The idea is to look at spaces with recognisable 'components' that we can use to ensure the space does not have a computable presentation. We take a metric with just two non-zero distances: one small distance for points within the same component, and one large distance for points in different components.

Let an *n-component* be a component with n points in it. Let a *discrete component space* be a discrete metric space with infinitely many 1-components and one n-component for infinitely many $n > 1$. Note that for each $n > 1$ there is at most one n-component. Define a metric for discrete component spaces:

$$d(x,y) = \begin{cases} 0 & \text{if } x = y; \\ 1 & \text{if } x \neq y \text{ and } x, y \text{ are within the same component;} \\ 2 & \text{if } x \neq y \text{ and } x, y \text{ are in different components.} \end{cases}$$

The idea is that we can 'code' a set $S \subseteq \mathbb{N}$ into a discrete component space $\mathcal{M}_S$ so that $\mathcal{M}_S$ has a computable presentation if, and only if, S is limitwise monotonic, as defined below:

Definition 17. [41, 43, 38] An infinite set $S \subseteq \mathbb{N}$ is said to be limitwise monotonic if

$$S = \operatorname{rng} \sup_y f(x,y)$$

for some total computable f where $\sup_y f(x, y)$ exists for all x.

The idea of the coding is that $n \in S$ iff $\mathcal{M}_S$ has an n-component. Further, if S is Δ^0_2 (but not necessarily limitwise monotonic) then $\mathcal{M}_S$ has a left-c.e. presentation and, via a different construction, a right-c.e. presentation. Since it is well-known [41, 43] that not every Δ^0_2 set is limitwise monotonic, the theorem follows.

Lemma 18. *A discrete component space $\mathcal{M}$ has a computable presentation if and only if the set $S_\mathcal{M} = \{n > 1 : \mathcal{M}$ has an n-component$\}$ is limitwise monotonic.*

Proof. Suppose $\mathcal{M} = (M, d)$ has a computable Polish presentation X. Then X is dense in M and $d(x_i, x_j)$ are uniformly computable reals for all $x_i, x_j \in X$. Because of the discrete component space metric, it suffices to compute $d(x_i, x_j)$ to within an accuracy of $\frac{1}{2}$ in order to know if $d(x_i, x_j)$ is $0, 1$ or 2. As such, when we say compute the distance $d(x_i, x_j)$, we mean compute it to within an accuracy of $\frac{1}{2}$, and from that conclude what the distance must be. Construct a function f witnessing that $S_\mathcal{M}$ is limitwise monotonic.
Construction.

Stage 0 Let $f(k, 0) = 1$ for all k. Declare all x_j to be unassigned.

Stage s Compute the distances $d(x_i, x_s)$ for all $i \leqslant s$. Then for each such i in turn, consider the following cases.

Case 1 Both x_i and x_s are unassigned, but $d(x_i, x_s) = 1$. Assign x_i and x_s to the k-th component where $k \in \omega$ is the least number not currently used to label a component. Define $f(k, s) = 2$.

Case 2 The element x_i is already assigned to the k-th component, x_s is unassigned and $d(x_i, x_s) = 1$. Assign x_s to the k-th component and define $f(k, s) = f(k, s - 1) + 1$.

Case 3 Either $d(x_i, x_s) = 1$ and both x_i and x_s are already assigned to a component, or $d(x_i, x_s) = 0$, or $d(x_i, x_s) = 2$. Do nothing.

For all $f(k,s)$ that are not already defined, let $f(k,s) = f(k,s-1)$.

Notice that as more points are computed in X, we may find new n-components and these can grow, but never shrink. Hence $f(x,s)$ is monotonically increasing in s. Since the k-th component corresponds to an n-component for some n, at some large enough stage s_k, all n points within this component will have shown up in the construction and been assigned to the k-th component, and so $f(k,s) = n$ for all stages $s \geqslant s_k$. Thus $\lim_s f(k,s) = \sup_s f(k,s) = n$, f is computable, and in particular $S_{\mathcal{M}} = \operatorname{rng}\sup_s f(k,s)$. Hence $S_{\mathcal{M}}$ is limitwise monotonic.

Now suppose $S = \operatorname{rng}\,\sup_y f(x,y)$ is an infinite limitwise monotonic set, so f is computable and the supremum exists for every x. It is known [40, 39] that if the range is infinite, we can ensure that f is injective. We can also assume that $f(i,0) = 0$ for all i. We construct a computable presentation $\mathcal{X} = ((x_i)_{i\in\omega}, d)$ for the discrete component space $\mathcal{M}$ that has $S_{\mathcal{M}} = S$. To uniformly compute the distances $d(x_i, x_j)$, wait for both x_i and x_j to show up in the construction, and from then on output either 1 or 2 as per the construction. For simplicity, $d(x_i, x_j) = 0$ iff $i = j$.

Construction.

Stage 0 For all i define $n_{i,0} = 0$ and say that x_i is unassigned.

Stage $s > 0$ For every $i \leqslant s$, if $f(i,s) > n_{i,s-1}$ then let $m = f(i,s) - n_{i,s-1}$, take the least k for which x_{2k} is unassigned and assign x_{2k}, $x_{2(k+1)}, \dots, x_{2(k+m-1)}$ to i (to the i-th component). If s is odd, assign x_s to -1 (this is to say it is not in an n-component for $n > 1$).

Define $d(x_i, x_j) = 1$ for each x_i, x_j that are assigned to the same component $k \neq -1$, and $d(x_i, x_j) = 2$ for x_i, x_j that are assigned to different components (including if one of x_i, x_j is assigned to -1), or if both x_i and x_j are assigned to -1. Finally, define $n_{i,s} = \max_{i\leqslant s} f(i,s)$.

Here the odd-numbered elements x_{2k+1} each form a 1-component, giving us countably many 1-components. Since f is injective and has infinite range, there is exactly one n-component per n in the range, which consists of points x_i with even indices. Thus we have constructed a computable presentation of $\mathcal{M}$, a discrete component space with $S_{\mathcal{M}} = S$. □

Lemma 19. *For any infinite Δ^0_2 set S, there is a left-c.e. discrete component space $\mathcal{M}$ such that $S_{\mathcal{M}} = S$.*

Proof. Fix f injective, computable, with $f(x, 0) = 0$ for all x, such that $S = \text{rng}(\lim_s f(x, s))$ and $\lim_s f(x, s)$ exists for every x. Construct a left-c.e. discrete component space $\mathcal{M}$ where $d(x_i, x_j) = 0$ iff $i = j$.
Construction.

Stage 0 Declare all x_i to be unassigned.

Stage s If s is odd then assign x_s to -1 (to say it is not in any n-component, it is a 1-component). For each $i \leqslant s$, consider the following cases.

Case 1 $f(i, s) = f(i, s-1)$. Do nothing

Case 2 $f(i, s) > f(i, s-1)$. Let $m = f(i, s) - f(i, s-1)$, take the least k for which x_{2k} is unassigned and assign $x_{2k}, x_{2(k+1)}, \ldots, x_{2(k+m-1)}$ to i, the i-th component.

Case 3 $f(i, s) < f(i, s-1)$. Let $m = f(i, s-1) - f(i, s)$, take the first m elements x_j that are currently assigned to i and re-assign them to -1.

Enumerate $\{r \in \mathbb{Q} : r < 1\}$ into the left cuts $d^-(x_i, x_j)$ for all $i \neq j$ such that x_i, x_j are assigned to the same component $k \neq -1$. Enumerate $\{r \in \mathbb{Q} : r < 2\}$ into the left cuts $d^-(x_i, x_j)$ for all x_i, x_j assigned to different components, or if one or both of x_i, x_j are assigned to -1.

Elements are only ever reassigned from i to -1, so distances only ever increase and hence this presentation of $\mathcal{M}$ is a left-c.e. presentation. Since f is injective, the space $\mathcal{M}$ is a discrete component space with $S_{\mathcal{M}} = S$. □

Lemma 20. *For any infinite Δ^0_2 set S, there is a right-c.e. discrete component space $\mathcal{M}$ such that $S_{\mathcal{M}} = S$.*

Proof. Fix f injective, computable, with $f(x,0) = 0$ for all x, such that $S = \operatorname{rng}(\lim_s f(x,s))$ and $\lim_s f(x,s)$ exists for every x. Construct a right-c.e. discrete component space $\mathcal{M}$.

Construction.

Stage 0 Declare all x_i to be unassigned.

Stage s If s is odd then assign x_s to -1 (to say it is not in any n-component, it is a 1-component). For each $i \leqslant s$, consider the following cases.

Case 1 $f(i,s) = f(i,s-1)$. Do nothing

Case 2 $f(i,s) > f(i,s-1)$. Let $m = f(i,s) - f(i,s-1)$, take the least k for which x_{2k} is unassigned and assign $x_{2k}, x_{2(k+1)}, \ldots, x_{2(k+m-1)}$ to i, the i-th component.

Case 3 $f(i,s) < f(i,s-1)$. For x_j that are assigned to i, re-assign x_j to -2 (to say it is no longer in any n-component). If $f(i,s) > 1$, then take the least k for which x_{2k} is unassigned and assign $x_{2k}, x_{2(k+1)}, \ldots, x_{2(k+f(i,s)-1)}$ to i.

Enumerate $\{r \in \mathbb{Q} : r > 1\}$ into the right cuts $d^+(x_i, x_j)$ for all $i \neq j$ where x_i, x_j are assigned to the same component $i > -1$. Enumerate $\{r \in \mathbb{Q} : r > 2\}$ into the right cuts $d^+(x_i, x_j)$ where x_i, x_j are assigned to different components (including if exactly one of x_i, x_j is assigned to -1 or to -2), or if both x_i, x_j are assigned to -1. Enumerate $\{r \in \mathbb{Q} : r > 0\}$ into the right cuts $d^+(x_i, x_j)$ where x_i, x_j are both assigned to -2.

Elements are only ever reassigned from i to -2, and all elements that are assigned to -2 are made into the same point, so distances only ever decrease and hence this presentation is a right-c.e. presentation of $\mathcal{M}$. Since f is injective, the space $\mathcal{M}$ is a discrete component space with $S_{\mathcal{M}} = S$. $\square$

To complete the proof of Theorem 16, take S to be Δ^0_2 not limitwise monotonic. Then consider the space $\mathcal{M}$ that has $S_{\mathcal{M}} = S$, which is unique up to isometry. Use this S in Lemma 19 and Lemma 20 to construct a left-c.e. and a right-c.e. presentation of $\mathcal{M}$. By Lemma 18, $\mathcal{M}$ is not computable (has no computable presentation) since S is not limitwise monotonic. □

Now we consider compact, connected spaces in order to show our main result.

Theorem 21. *There is a compact connected Polish space that has a left-c.e. presentation and a right-c.e. presentation but does not have a computable presentation, up to isometry.*

Proof. Our spaces here can each be viewed as a closed subset of the Hilbert cube. We think of points in terms of their coordinates. Then the distance between two points $x = (x_0, x_1, x_2, \ldots)$ and $y = (y_0, y_1, y_2, \ldots)$ is

$$d(x, y) = \sqrt{\sum_i (x_i - y_i)^2}.$$

The points will have at most two non-zero coordinates, so the sum has at most 4 (non-zero) terms in it and therefore is finite.

For an infinite set S, define a *component space* $\mathcal{M}_S$ as follows. The space $\mathcal{M}_S$ has a central point with branches coming out of it: each branch is a line segment along an axis (so points in a line segment have a single non-zero coordinate), perpendicular to all other branches. An *n-component* is a branch of length $\frac{4}{2^n}$ with a circle of radius $\frac{3}{4}\frac{1}{2^n}$ attached to the end. Each circle is in its own plane, perpendicular to all other branches and circles (so points on a circle have at most two non-zero coordinates). For each $n \in S$, the component space $\mathcal{M}_S$ has exactly one n-component. Additionally, for each $n \notin S$, $\mathcal{M}_S$ has an empty branch (a line segment with no circle on the end) of length $\frac{4}{2^n}$, and a branch of length $\frac{4}{2^n}$ with a larger circle of radius $\frac{1}{2^n}$ attached to the end. These bits of 'junk' will *not* be considered as n-components.

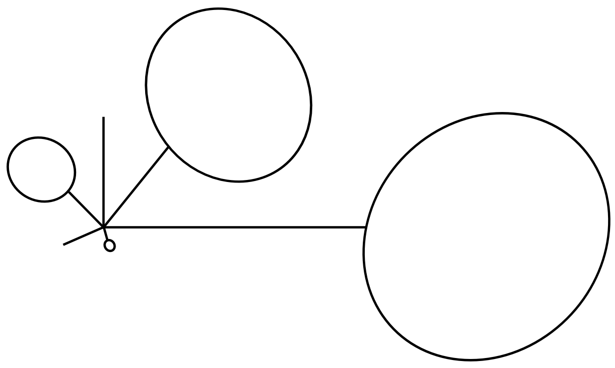


Figure 2.1: Example sketch of a component space. To accurately depict this, we would need 10 dimensions, since each circle and each line is in a plane perpendicular to all other circles and lines in the space.

The idea is that to get rid of n-components in the left-c.e. space, we expand the circle in the n-component to be of radius $\frac{1}{2^n}$. In the right-c.e. space, collapse it to a single point. Then to make the left-c.e. and right-c.e. constructions isomorphic, when we get rid of an n-component in the left-c.e. space we will introduce an empty branch of the same length, and in the right-c.e. space a branch with a circle of radius $\frac{1}{2^n}$.

For example, Figure 2.1 is a sketch of a component space, with a 1-component and a 4-component, but without a 2-component nor a 3-component (instead there is some 'junk'), and all other branches are not shown.

Lemma 22. *A circle of radius r can be unambiguously recognised in a computable presentation of a component space, by finding four points x_0, x_1, x_2, x_3 with distances $d(x_0, x_1) = d(x_1, x_2) = d(x_2, x_3) = d(x_3, x_0) = \sqrt{2}r$ and $d(x_0, x_2) = d(x_1, x_3) = 2r$, where the distances have been computed up to a high enough accuracy. That is, such points* must *come from a circle of radius r and not from any other part of the space.*

Proof. The concern here is that four such points might at first appear to be coming from a circle, but at a later stage we find out that in fact they are from elsewhere in the space, which would mean that there actually is no circle of radius r in the space. For instance, simply finding *three* points that are equidistant is not enough to determine that there is a circle of radius r in the space, because although they determine a unique circle in 2D Euclidean space, in our spaces they may in fact be located on 3 different branches, as in Figure 2.2.

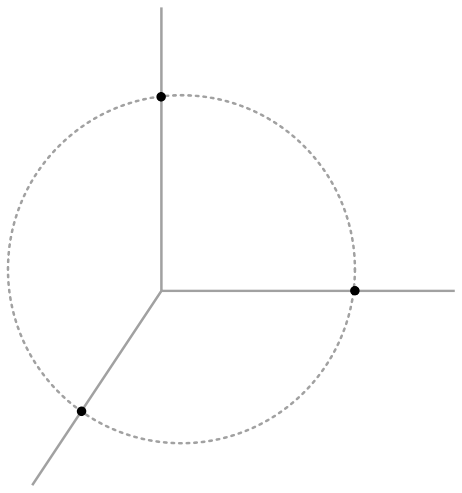


Figure 2.2: Three points that appear to be on a circle but are not from a circle

However, in a component space (although, of course, not in general), four points *are* sufficient. Think of a coordinate system for the component space where the origin is at the centre of the space and each point has at most two non-zero coordinates, one if it is on a branch and two if it is on a circle. That is, branches lie along axes and there is a second axis per component. Then the perpendicularity of components means that two points within different components cannot both have non-zero entries at the same coordinate. Let us consider the possible cases.

Case 1: There are 3 points that are within the same component

In this case, in order to have the distances described, the 4th point must be in the same component as the other points. This is because if we have 3 points, they lie on a plane, and a fourth point with the distances described is forced to lie within the same plane as the first 3 points. This can be checked by a simple calculation:

Consider the first 3 points x_0, x_1, x_2 (which are within the same component) under a new coordinate system, where x_1 is at the origin and x_0, x_2 are along axes. That is, $x_1 = (0,0,0)$, $x_0 = (\sqrt{2}r, 0, 0)$, $x_2 = (0, \sqrt{2}r, 0)$ and the fourth point $x_3 = (w, y, z)$ is to be determined. Then

$$d(x_0, x_3)^2 = w^2 - 2\sqrt{2}rw + 2r^2 + y^2 + z^2$$

$$= d(x_2, x_3)^2 = w^2 + y^2 - 2\sqrt{2}ry + 2r^2 + z^2$$

That is, $w = y$. Now $d(x_1, x_3)^2 = 4r^2 = w^2 + y^2 + z^2 = 2w^2 + z^2$ so $z^2 = 4r^2 - 2w^2$. Finally,

$$d(x_0, x_3)^2 = 2r^2 = w^2 - 2\sqrt{2}rw + 2r^2 + w^2 + 4r^2 - 2w^2$$

So $w = \sqrt{2}r$, and hence $z^2 = 0$. That is, the fourth point x_3 *must* be within the same plane as x_0, x_1, x_2 in order to have the distances described.

In this case, this means the fourth point must be within the same component as the other three. Now in the case that all four points are within the same component, they clearly must all come from the circle.

Case 2: There are 2 points within the same component

Suppose we have two points within one component and a third point in a different component. Consider the plane on which these three points lie. Similar to Case 1, the distances between the points force the fourth point to be within this same plane. Since we are in a component space, the fourth point can have at most two non-zero coordinates. By perpendicularity of the components, these coordinates cannot be a mix of the non-zero coordinates from the component of the first two points and the component of the third point. That is, the fourth point must be either in the same component as the first two points (in which case we have Case 1 again), or in the same component as the third point. In the latter, we have two points in one component and two points in a second component. If one of the points is at the origin, we are back to Case 1. Otherwise, let us consider the two subcases. For notational simplicity, we will write only the 4 potentially non-zero coordinates, so denote (w, y, u, v) the coordinates of a point, where the first two coordinates are the non-zero coordinates of one component and the second two coordinates are the non-zero coordinates of the other components.

Subcase 1: (at least) one point is on a branch

Let $x_0 = (w_0, y_0, 0, 0),\ x_1 = (0, 0, u_0, 0)$ and $x_2 = (0, 0, u_1, v_1)$ be the first three points, so x_1 is on a branch and y_0, v_1 may potentially be zero (so x_0 and x_2 may or may not be on branches). Then

$$\begin{aligned} d(x_0, x_1) &= \sqrt{w_0^2 + y_0^2 + u_0^2} \\ d(x_0, x_2) &= \sqrt{w_0^2 + y_0^2 + u_1^2 + v_1^2} \\ d(x_2, x_3) &= \sqrt{u_1^2 - 2u_0u_1 + u_0^2 + v_1^2} \end{aligned}$$

Now if $d(x_0, x_1) = d(x_1, x_2) = \sqrt{2}r$ and $d(x_0, x_2) = 2r$, then $w_0^2 + y_0^2 + u_0^2 = u_1^2 - 2u_0u_1 + u_0^2 + v_1^2$ and so

$$\begin{aligned} d(x_0, x_2)^2 = 4r^2 &= w_0^2 + y_0^2 + u_1^2 + v_1^2 \\ &= 2u_1^2 + 2v_1^2 - 2u_0u_1 \end{aligned}$$

But then

$$\begin{aligned} d(x_1, x_2)^2 = 2r^2 &= u_1^2 + v_1^2 - 2u_0u_1 + u_0^2 \\ &= u_1^2 + v_1^2 - u_0u_1 \end{aligned}$$

That is, $u_0u_1 = u_0^2$. That is, $u_1 = u_0$ and so $v_1 = 0$ because x_1 is on a branch and the circles are on the ends of the branches. But then $x_1 = x_2$, a contradiction since we assumed x_2 is a distinct point.

Similarly if $d(x_0, x_1) = d(x_0, x_2) = \sqrt{2}r$ and $d(x_1, x_2) = 2r$, a simple calculation gives rise to a contradiction.

Subcase 2: all points are on circles

Let x_0, x_1 be on the n^{th} circle and x_2, x_3 be on the m^{th} circle. These circles have radius at most $\frac{1}{2^n}$ and $\frac{1}{2^m}$ respectively (if the space does not have an n-component or m-component. If it does, the radii will be $\frac{3}{4}$ this). Then since x_0, x_1 are both on the same circle, $d(x_0, x_1) \leqslant \frac{2}{2^n}$. Further, x_0 and x_2 are on separate components and must each be on the circle part. That is, x_0 is at least $\frac{4}{2^n}$ (the length of the branch) away from the origin, and similarly x_2 is at least $\frac{4}{2^m}$ away from the origin. Thus by Pythagoras' theorem, $d(x_0, x_2) \geqslant 4\sqrt{\frac{1}{2^{2n}} + \frac{1}{2^{2m}}}$. Without loss of generality, assume $n \geqslant m$. Consider when $d(x_0, x_2) = d(x_0, x_1) = \sqrt{2}r$ and $d(x_0, x_3) = 2r$. Now we have a contradiction:

$$\frac{4\sqrt{2}}{2^n} \leqslant 4\sqrt{\frac{1}{2^{2n}} + \frac{1}{2^{2m}}} \leqslant d(x_0, x_2) = d(x_0, x_1) \leqslant \frac{2}{2^n}$$

The other case, where $d(x_0, x_2) = d(x_0, x_3) = \sqrt{2}r$ and $d(x_0, x_1) = 2r$ gives a similar contradiction. The point is that the lengths of the branches combined with the relative size of the circles make this case impossible.

Case 3: Each of the 4 points is in a different component

Consider the first three points x_0, x_1, x_2, which collectively have at most 6 non-zero coordinates, 2 per point, and consider the plane on which they lie. Any non-zero coordinate of any point on this plane must be one of the 6 coordinates aforementioned. As in Case 1, the fourth point must lie on this plane in order to satisfy the described distances, and so by perpendicularity of components it must be within the same component as one of the first three points. That is, in a component space, four points with the distances described cannot all be in different components.

This covers all the possible cases, and shows that 4 points in a component space with the described distances must indeed come from a circle of radius r and not anywhere else.

In a computable presentation, at each finite stage we do not know the distances *exactly*, so it may still be ambiguous where a point is in the space. For instance if we have 4 points whose distances at the current stage are known to within an accuracy of $2r$, or even to within an accuracy of $\frac{r}{2}$, and could have the described distances (as far as we know currently with this level of accuracy), we cannot conclude that they actually are from a circle of radius r, or of radius $\frac{3}{4}r$, etc. However, in component spaces, we have a discrete jump between the possible radii that circles can have, and the smaller the radius, the smaller this jump is. Because of this, if we compute the distances to a high enough accuracy and find the described distances, for instance up to $\frac{r}{100}$ (although of course a lower accuracy would do just fine too as long as it is not too low), it is clear that the points must be from a circle of radius r and not from anywhere else in the space nor from a circle of a different radius. The accuracy that we need to compute the distances to, in order to be certain that the points are what they appear to be, depends on the radius of the circle – the smaller the radius, the more accurate we need to be. $\square$

Lemma 23. *If a component space $\mathcal{M}$ has a computable presentation, then the set $S_{\mathcal{M}} = \{n \in \mathbb{N} : \mathcal{M}$ has an n-component$\}$ is computably enumerable.*

Proof. Suppose $\mathcal{M}$ is a component space with a computable presentation. Enumerate $S_{\mathcal{M}}$ as follows. At stage s, for each $n \leqslant s$ not already enumerated into $S_{\mathcal{M}}$, consider every quadruple of points x_i, x_j, x_k, x_l that have shown up in the computable construction of $\mathcal{M}$ by stage s. If the following two conditions hold for these points x_i, x_j, x_k, x_l, then enumerate n into $S_{\mathcal{M}}$:

1. The distances between x_i, x_j, x_k and x_l have been computed to an accuracy as described in the Lemma 22.

2. The distances are, to within this accuracy, $d(x_i, x_j) = d(x_j, x_k) = d(x_k, x_l) = d(x_l, x_i) = \sqrt{2}r$ and $d(x_i, x_k) = d(x_j, x_l) = 2r$ where $r = \frac{3}{4}\frac{1}{2^n}$.

Then if $\mathcal{M}$ has a circle of radius $\frac{3}{4}\frac{1}{2^n}$, at some stage there will be four points that satisfy the two conditions. When this happens, n gets enumerated into $S_{\mathcal{M}}$. Further, if at some stage n is enumerated, then by Lemma 22 the computable presentation must have a circle of radius $\frac{3}{4}\frac{1}{2^n}$. Thus the c.e. set $S_{\mathcal{M}}$ constructed is indeed $S_{\mathcal{M}} = \{n \in \mathbb{N} : \mathcal{M}$ has an n-component$\}$. $\square$

Lemma 24. *For any co-c.e. set S, there is a left-c.e. component space $\mathcal{M}$ with $S_{\mathcal{M}} = S$.*

Proof. Let S be an infinite co-c.e. set. Observe that in a left-c.e. presentation of a component space, we can start constructing what appears to be a circle of radius r on the end of a branch, and then at some stage s, increase the distances so that the circle now has a bigger radius. This can be done while still maintaining that the space is left-c.e. (this is not necessarily true for any arbitrary shape, but is for circles). For a point y on the circle, as in Figure 2.3, imagine a ray through y that starts where the circle meets the branch. Then the place where this ray hits the larger circle is where we send y to.

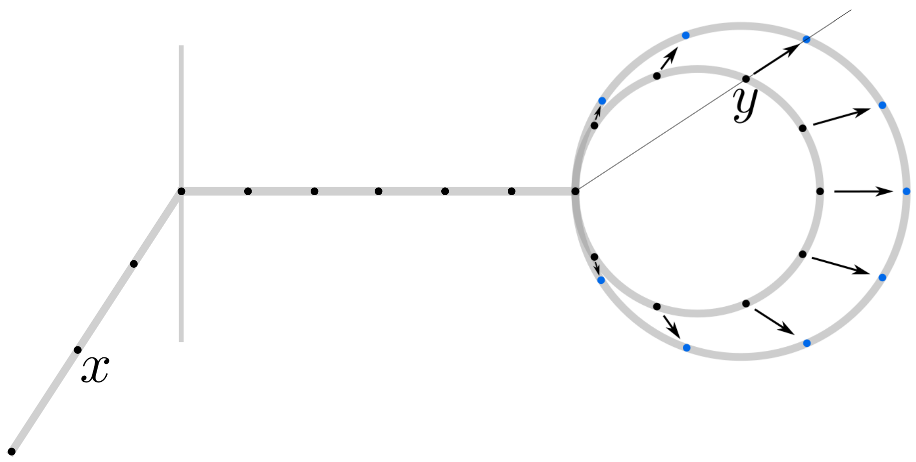


Figure 2.3: Expanding a circle at stage s in a left-c.e. manner

The points on the branch stay fixed, and so distances of any points within the component only get larger. For any point x anywhere else in the space, consider just the points x and any point y on the circle we are expanding. Each only has at most 2 non-zero coordinates, and these coordinates must be different since x and y are within different components. So the distance between them is $d(x, y) = \sqrt{x_0^2 + x_1^2 + y_0^2 + y_1^2}$, where x_0, x_1 are the (possibly) non-zero coordinates of x, and y_0, y_1 are the (possibly) non-zero coordinates of y. In expanding the circle, the absolute values of y_0 and y_1 can only increase, and hence $d(x, y)$ must increase too. That is, expanding a circle in a component space in this manner is a left-c.e. process.

Build the n-th component to occupy the coordinates $3n$ and $3n + 1$, so for any point in the component, these are the only two coordinates with (potentially) non-zero entries.

Construction.

Stage s Add two or three new points to the construction for each $n \leqslant s$ as follows, by enumerating the appropriate numbers into the left cuts of the distances between points.

Case 1 n is in S at stage s. Add a new point each to the circle and to the branch in the coordinates $3n$ and $3n + 1$, to continue constructing an n-component.

Case 2 n leaves the set S at stage s, so $n \in S_{s-1} \setminus S_s$. Expand the circle in the coordinates $3n$ and $3n + 1$ to have a radius of $\frac{1}{2^n}$ instead of $\frac{3}{4}\frac{1}{2^n}$, as in Figure 2.3. Add a new point to the expanded circle and a new point to the branch in the coordinates $3n$ and $3n + 1$.

Case 3 n left the set S at an earlier stage $t < s$. Add a point each to the expanded circle and to the branch in the coordinates $3n$ and $3n + 1$. Add a third point in the coordinate $3n + 2$, building an empty branch of length $\frac{4}{2^n}$ in this coordinate.

Then $\mathcal{M}$ has an n-component exactly for the n that are in S, so $S_{\mathcal{M}} = S$, and $\mathcal{M}$ has 'junk' for each n that is not in S. In particular, $\mathcal{M}$ is exactly the component space $\mathcal{M}_S$ built from the set S. □

Lemma 25. *For any co-c.e. set S, there is a right-c.e. component space $\mathcal{M}$ with $S_{\mathcal{M}} = S$.*

Proof. Let S be an infinite co-c.e. set. Observe that in a right-c.e. presentation of a component space, we can collapse a circle into a point in a right-c.e. manner.

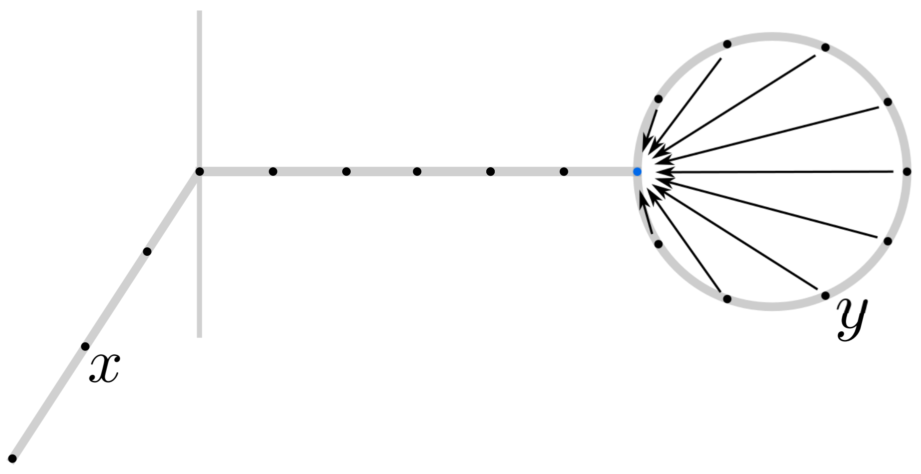


Figure 2.4: Collapsing a circle at stage s in a right-c.e. manner

As in Figure 2.4, distances of points within the component only get smaller. For any point x anywhere else in the space, consider just the points x and any point y on the circle we are collapsing. Again, each only has at most two non-zero coordinates which must be different. So the distance between them is $d(x, y) = \sqrt{x_0^2 + x_1^2 + y_0^2 + y_1^2}$, where x_0, x_1 are the (possibly) non-zero coordinates of x, and y_0, y_1 are the (possibly) non-zero coordinates of y. In collapsing the circle, the absolute values of y_0 and y_1 only decrease, and hence $d(x, y)$ must decrease too. As such, this collapse is a right-c.e. process.

Build the n-th component to occupy the coordinates $3n$ and $3n + 1$, so for any point in the component, these are the only two coordinates with (potentially) non-zero entries.

Construction.

Stage s Add one, two or three new points to the construction for each $n \leqslant s$ as follows, by enumerating the appropriate numbers into the right cuts of the distances between points.

Case 1 n is in S at stage s. Add a new point each to the circle and to the branch in the coordinates $3n$ and $3n+1$, to continue constructing an n-component.

Case 2 n leaves the set S at stage s, so $n \in S_{s-1} \setminus S_s$. Collapse the circle in the coordinates $3n$ and $3n+1$ onto the branch (which only occupies the coordinate $3n$), as in Figure 2.4. Add a new point to the now empty branch in the coordinate $3n$. Note that there are no longer any points with a non-zero entry in the coordinate $3n+1$.

Case 3 n left the set S at an earlier stage $t < s$. Add a point to the empty branch in coordinate $3n$. Add two more points in the coordinates $3n+1$ and $3n+2$, building a branch of length $\frac{4}{2^n}$ and a circle of radius $\frac{1}{2^n}$ on the end of the branch.

As in the previous lemma, $\mathcal{M}$ has an n-component exactly for the n that are in S and hence $S_{\mathcal{M}} = S$, and it has 'junk' for each n that is not in S. That is, once again we have constructed a presentation of the component space $\mathcal{M}_S$ built from the set S. □

To complete the proof of Theorem 21, take $S = \overline{K}$, the complement of the halting problem. Consider the component space $\mathcal{M}_S$. Construct a left-c.e. and a right-c.e. presentation of this space using Lemma 24 and Lemma 25. By Lemma 23, if $\mathcal{M}$ had a computable presentation then $\overline{K}$ would be computably enumerable, hence $\mathcal{M}$ cannot have a computable presentation. □

2.2 Sequential Computability and Effective Continuity

In this section we compare different notions of effective presentability for Polish groups. The notions differ in general, but for our main result it makes no difference which of the two non-equivalent notions we choose. To see that, we need to analyse these notions carefully.

To define an effective presentation of a Polish group, we require the domain and the group operation to be effective in some sense. For the domain, it is natural to consider the notions of left-c.e., right-c.e. and computable Polish metric presentations. As for the operations, one could use either effectively continuous or sequentially computable operations, as defined below.

Definition 26. Let X, Y be Polish spaces. A function $f : X \to Y$ is said to be *sequentially computable*, if there is a Turing functional Φ such that given a fast Cauchy name $(x_i)_{i\in\omega}$ for some $x \in X$, $\Phi\left((x_i)_{i\in\omega}\right)$ outputs a fast Cauchy name $(y_i)_{i\in\omega}$ for $f(x) = y \in Y$. Recall that a fast Cauchy name is a sequence $(x_i)_{i\in\omega}$ of special points such that for each $i \in \omega$, $d(x_i, x_{i+1}) \leq 2^{-i-1}$, or equivalently, $d(x_i, x_j) \leq 2^{-i}$ for all $j \geq i$.

Definition 27. Let $\mathcal{X}, \mathcal{Y}$ be topological spaces with effective listings of their basic open sets, each of which are nonempty, $(X_n)_{n\in\omega}$ and $(Y_n)_{n\in\omega}$, respectively. Then a function $f : \mathcal{X} \to \mathcal{Y}$ is said to be *effectively continuous* if there is some c.e. set W such that for each j, $f^{-1}(Y_j) = \bigcup_{i\in I} X_i$, where $I = \{i : (i, j) \in W\}$.

Note that we do not require $\mathcal{X}$ and $\mathcal{Y}$ to be computable topological spaces. In a computable topological space, the intersections of basic open sets is uniformly computably enumerable, and each basic open set must be nonempty; see [33]. It is well-known that basic open balls in a right-c.e. Polish space form a computable topological space. In a left-c.e. Polish space, we can list basic open balls, however, this will not necessarily induce a computable topological presentation of the space[1].

It is known that being sequentially computable is equivalent to being effectively continuous in right-c.e. Polish spaces [19]. We now prove that this equivalence fails for the group operations in left-c.e. Polish groups. In

[1]It has recently been shown in [7] that every Δ^0_2 Polish space is homeomorphic to a computable topological space.

the following lemmas we use the product metric on the domain of the group operation.

Lemma 28. *There is a Polish group with a left-c.e. presented domain for which the group operation and inverse operation are sequentially computable but not effectively continuous.*

Recall that a function being effectively continuous imply that given any basic open set Y, we may effectively enumerate its preimage $f^{-1}(Y)$. In particular, to enumerate N^y, assuming we have N^x for some $f(x) = y$, first enumerate the preimage $f^{-1}(Y)$ for each basic open set Y. Here the names N^x, N^y are sets containing exactly the codes of all basic open sets which contain x, y respectively. For each Y such that $f^{-1}(Y)$ enumerates some code for a basic open set also enumerated by N^x, we enumerate (the code of) Y into N^y. That is, if $f(x) = y$, and f is effectively continuous, then N^y is enumerable in N^x. We state this as a fact below.

Fact. If f is effectively continuous, $f(x) = y$, and N^x is c.e., then N^y is computably enumerable.

Proof of Lemma 28. Consider the discrete group $G = \bigoplus_{i\in\omega} \mathbb{Z}_3$ with elements represented as sequences consisting of $0, 1$, and 2, with cofinitely many 0s, using these as our special points $\{\alpha_i\}_{i\in\omega}$, fixing α_0 to be a non-identity element.

Now, consider the following metric on G, assuming w.l.o.g. that $i \leq j$.

$$d(\alpha_i, \alpha_j) = \begin{cases} 2 & \text{if } i = 0 \text{ and } j \in \emptyset' \setminus \{0\} \\ 0 & \text{if } i = j \\ 1 & \text{otherwise.} \end{cases}$$

Evidently, $d(\alpha_i, \alpha_j)$ is left-c.e.. Notice that both the group operation and the inverse operation are sequentially computable in $(G, d, +, ^{-1})$. This is because every Cauchy name for an element α_i must be constant from the second term onward, and will output i. Once we obtain i, we know exactly what element α_i is and so we can compute the index for α_i^{-1}. A similar argument holds for the group operation $+$. However, we claim that neither the group operation nor the inverse are effectively continuous.

To see this, notice first that α_0 is the only non-computable point (in the sense that N^{α_0} is not c.e.). Suppose for a contradiction that there exists a c.e. $W = N^{\alpha_0}$. This implies that the predicate '$\alpha_0 \in B_d(\alpha_i, 2)$' is c.e.. However, for each $i > 0$, by definition of d, $\alpha_0 \in B_d(\alpha_i, 2)$ iff $i \notin \emptyset'$. This would make the complement of $\emptyset'$ computably enumerable, a contradiction.

On the other hand, if $i \neq 0$, it is clear that for any $j \neq 0$, $\alpha_i \in B_d(\alpha_j, r)$ for any $r > 1$. By non-uniformly fixing whether or not $i \in \emptyset'$, we can then decide whether or not to include (the code of) $B_d(\alpha_0, r)$ in N^{α_i}. Thus, for each $i \neq 0$, N^{α_i} is c.e..

Since α_0 is not the group identity, we may conclude that there exists $i > 0$ such that $\alpha_i^{-1} = \alpha_0$ and there exists $j, k \neq 0$ such that $\alpha_j + \alpha_k = \alpha_0$. Now the inverse function, and the group operation sends α_i, and (α_j, α_k) respectively to α_0. By applying the Fact, neither the group operation nor the inverse operation may be effectively continuous, because α_0 is the only non-computable point in the sense described above. □

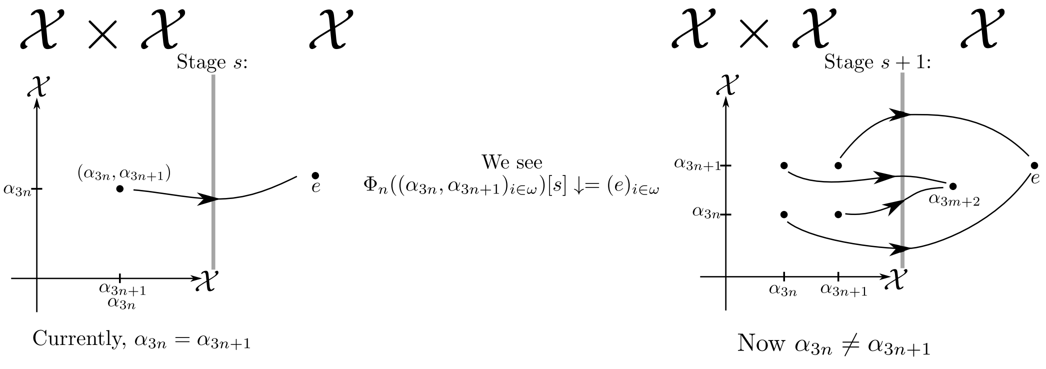
Lemma 29. *There is a Polish group with the following properties.*

- *The domain is left-c.e. presented.*
- *The group operation is effectively continuous but not sequentially computable.*
- *The inverse is both effectively continuous and sequentially computable.*

Proof. We construct a left-c.e. presentation of our space $\mathcal{X} = \bigoplus_{i\in\omega} \mathbb{Z}_2$ as follows. Let $\{\alpha_i\}_{i\in\omega} \cup \{e\}$ be the dense set of special points. The distinguished special point e will serve as the group identity. We satisfy the following requirements, where $(x, y) \in \mathcal{X} \times \mathcal{X}$.

$$R_n : (x_i, y_i)_{i\in\omega} \text{ is a Cauchy name for } (x, y) \text{ but } \Phi_n\left((x_i, y_i)_{i\in\omega}\right) \neq x + y.$$

During the construction, define a left-c.e. approximation of the standard uniform (distance 1) discrete metric on $\mathcal{X}$ by keeping the '=' relation co-c.e.. In particular, all points start at distance 0, and when two points are defined to be different, we increase the distance to 1. At the beginning of the construction, define e to be different from every other special point (increase the distance from 0 to 1).



$\mathcal{X} \times \mathcal{X}$
$\mathcal{X}$
Stage s:
$\mathcal{X}$
$(\alpha_{3n}, \alpha_{3n+1})$
α_{3n}
e
α_{3n+1}
α_{3n}
$\mathcal{X}$
Currently, $\alpha_{3n} = \alpha_{3n+1}$
We see
$\Phi_n((\alpha_{3n}, \alpha_{3n+1})_{i\in\omega})[s] \downarrow = (e)_{i\in\omega}$
$\mathcal{X} \times \mathcal{X}$
$\mathcal{X}$
Stage $s+1$:
$\mathcal{X}$
α_{3n+1}
α_{3n}
e
α_{3m+2}
α_{3n}
α_{3n+1}
$\mathcal{X}$
Now $\alpha_{3n} \neq \alpha_{3n+1}$

Figure 2.5: Sketch of the basic strategy

Strategy for R_n The gadget to satisfy R_n consists of the special points α_{3n} and α_{3n+1}. Define $\alpha_{3n}, \alpha_{3n+1}$ to be different from α_i for all $i \neq 3n, 3n+1$ and for the time being, keep $\alpha_{3n} = \alpha_{3n+1}$. Compute Φ_n on the constant name $(\alpha_{3n}, \alpha_{3n+1})_{i\in\omega}$. While waiting, we maintain that $\alpha_{3n} = \alpha_{3n+1}$. If Φ_n never converges, then R_n is met since Φ_n is not total. Otherwise, at some stage s it converges on the name $(\alpha_{3n}, \alpha_{3n+1})_{i\in\omega}$. Furthermore, since the metric we define on $\mathcal{X}$ is the standard discrete metric, we may further assume that $\Phi_n((\alpha_{3n}, \alpha_{3n+1})_{i\in\omega})$ produces a constant name[2]. We now consider the following possibilities.

- If $\Phi_n((\alpha_{3n}, \alpha_{3n+1})_{i\in\omega})[s] \downarrow$ and produces the constant name $(e)_{i\in\omega}$, then define $\alpha_{3n} \neq \alpha_{3n+1}$ and $\alpha_{3n}+\alpha_{3n+1} = \alpha_{3m+2}$, for some fresh m

[2]If the output is not constant from the second term onwards, then Φ_n immediately fails as it failed to give a Cauchy name despite getting a Cauchy name as input.

not yet seen in the construction, thus satisfying R_n since $\alpha_{3m+2} \neq e$.

- If $\Phi_n((\alpha_{3n}, \alpha_{3n+1})_{i\in\omega})[s] \downarrow$ and produces the constant name $(\alpha_j)_{i\in\omega}$ for some j, do nothing and then R_n must be satisfied as $\alpha_{3n} + \alpha_{3n+1} = e \neq \alpha_j$.

The special points $\alpha_{3n}, \alpha_{3n+1}$ for each n will act as the generators for our group, and we use α_{3n+2} to ensure our group operation $+$ is well-defined. To this end, during the construction, we maintain a set stage-wise, defined as $\Theta_s = \{\alpha_{3n} : n < s\} \cup \{\alpha_{3n+1} : n < s \wedge \alpha_{3n} \neq \alpha_{3n+1} \text{ at stage } s\}$. We think of each α_{3n+2} as being the sum of a collection of generators from Θ_s. Observe that since our group is abelian and all elements have order two, we can in fact think of each α_{3n+2} as some subset of Θ_s. While we need not explicitly define any sum consisting of α_{3n+1} while $\alpha_{3n} = \alpha_{3n+1}$, we shall artificially do so in order to ensure that the operation is effectively continuous. More specifically, if we define $\alpha_{3n} + \alpha_{i_0} + \alpha_{i_1} + \cdots + \alpha_{i_k} = \alpha_{3m+2}$ where $\alpha_{3n}, \alpha_{i_0}, \alpha_{i_1}, \ldots, \alpha_{i_k}$ are distinct elements of Θ_s and $\alpha_{3n+1} \notin \Theta_s$, then we define $\alpha_{3n+1} + \alpha_{i_0} + \alpha_{i_1} + \cdots + \alpha_{i_k} = \alpha_{3(m+1)+2}$. Evidently, while $\alpha_{3n} = \alpha_{3n+1}$, we should also keep $\alpha_{3m+2} = \alpha_{3(m+1)+2}$. However, once α_{3n} is defined to be different from α_{3n+1}, we need also define $\alpha_{3m+2} \neq \alpha_{3(m+1)+2}$. To facilitate this process in the description of the construction, for each n and each stage s, let $E_{n,s}$ denote the collection of m for which $\alpha_{3n} + \alpha_{i_0} + \alpha_{i_1} + \cdots + \alpha_{i_k}$ has been defined to be α_{3m+2} by stage s and $\alpha_{3n+1} \notin \Theta_s$.

We arrange the construction as follows. At a stage s, we say that R_n *requires attention* if Φ_n converges on the constant name $(\alpha_{3n}, \alpha_{3n+1})_{i\in\omega}$ and outputs a name for e. During the construction, we also maintain $\Psi(i,j)$ a left-c.e. approximation to the distance between α_i and α_j. At the beginning of the construction, let

$$\Psi(i,j)[0] = \begin{cases} 0 & \text{if } i = 3n \text{ and } j = 3n+1 \text{ for some } n \\ 0 & \text{if } i = 3n+2 \text{ and } j = 3m+2 \text{ for some } m, n \\ 1 & \text{otherwise.} \end{cases}$$

Unless stated during the construction, $\Psi(i,j)[s+1] = \Psi(i,j)[s]$. For convenience, once we have defined $\Psi(i,j)[s]$, we also let $\Psi(j,i)[s] = \Psi(i,j)[s]$.

Construction.

Stage $s \geq 0$ Let n be the least index such that R_n requires attention. Then define $\Psi(3n, 3n+1)[s] = 1$, and for each $m \in E_{n,s}$, define $\Psi(3m+2, 3(m+1)+2)[s] = 1$.

For each tuple $\langle \alpha_{i_0}, \alpha_{i_1}, \ldots, \alpha_{i_l} \rangle$ for some $l \leq s$ and $\alpha_{i_j} \in \Theta_s$ (distinct) such that the sum $\alpha_{i_0} + \alpha_{i_1} + \cdots + \alpha_{i_l}$ has yet to be defined, let the sum be α_{3m+2} for some fresh m. For each $i < 3m+2$, define $\Psi(i, 3m+2)[s] = 1$. Recall that if one of the $\alpha_{i_j} = \alpha_{3k}$ for some k where $\alpha_{3k+1} \notin \Theta_s$, then we would also define $\alpha_{i_0} + \alpha_{i_1} + \cdots + \alpha_{3k+1} + \cdots + \alpha_{i_l}$ to be $\alpha_{3(m+1)+2}$. If we do so, then define $\Psi(i, 3(m+1)+2)[s] = 1$ for each $i < 3m+2$.

If R_n never requires attention, then $\Phi_n((\alpha_{3n}, \alpha_{3n+1})_{i\in\omega})$ does not produce a name for the identity. This implies that $\lim_s \Psi(3n, 3n+1)[s] = 0$ and $\alpha_{3n} + \alpha_{3n+1} = e$. On the other hand, if R_n requires attention at some stage, then it is eventually the highest priority requirement which does so. At such a stage s, we would have attended to R_n by letting $\Psi(3n, 3n+1)[s] = 1$, thus making $\alpha_{3n} \neq \alpha_{3n+1}$ and therefore $\alpha_{3n} + \alpha_{3n+1} \neq e$. Since the strategies do not interact, it follows that each R_n is satisfied, so $+$ is not sequentially computable.

It is easy to see that $\Psi(i,j)$ is left-c.e. for each i, j. During the construction, we only ever define $\Psi(i,j) = 1$; either $\Psi(i,j)$ stays at 0 forever or it is increased to 1 and never again decreased. Observe also that given any i, j, $\lim_s \Psi(i,j)[s] \in \{0, 1\}$, and also that $\lim_s \Psi(i,j)[s] = 0$ iff $\alpha_i = \alpha_j$. Thus, the group constructed has a left-c.e. presented domain.

It remains to show that the operation $+$ as defined is effectively continuous. We define a c.e. set W as follows. Let α_k be given. For any α_i, α_j defined to be $\alpha_{i_0} + \alpha_{i_1} + \cdots + \alpha_{i_p}$ and $\alpha_{j_0} + \alpha_{j_1} + \cdots + \alpha_{j_q}$ where α_k has also been defined to be the sum of all the unique elements from $\{\alpha_{i_0}, \alpha_{i_1}, \ldots, \alpha_{i_p}, \alpha_{j_0}, \alpha_{j_1}, \ldots, \alpha_{j_q}\}$, enumerate $\langle B(\alpha_i, 1) \times B(\alpha_j, 1), B(\alpha_k, 1)\rangle$

into the c.e. set W. Also enumerate the elements $\langle B(\alpha_k, 1) \times B(e, 1), B(\alpha_k, 1)\rangle$ and $\langle B(e, 1) \times B(\alpha_k, 1), B(\alpha_k, 1)\rangle$ into W. This witnesses the effective continuity of $+$ since everything that sums to α_k will be covered by some α_i, α_j.

Finally, consider the inverse operation. In this example, the inverse is very simple (for every α we have $-\alpha = \alpha$), and so is clearly both sequentially computable and effectively continuous. □

Remark 30. *For a Polish group with a uniform isolating radius (as in our example), if the inverse is effectively continuous then it must also be sequentially computable. To elaborate, let r be the isolating radius. Given a sequence $(x_i)_{i\in\omega}$, enumerate the inverse preimages of $B(x_i, r)$ (which necessarily contains only x_i). This should produce only open sets of the form $B(-x_i, s)$ for some $0 < s \leq r$, as each element has a unique inverse. We may then simply output $(-x_i)_{i\in\omega}$ as given by these enumerations. If $(x_i)_{i\in\omega}$ is a Cauchy name for x, then the output is a Cauchy name for $-x$, thus making the inverse operation sequentially computable.*

We showed that the notions of effective continuity and sequential computability differ for Polish groups with left-c.e. presented domains, for a fixed presentation. However, if a discrete Polish group has a left-c.e. presented domain and the multiplication and inverse operations are sequentially computable, then there is some other topologically isomorphic presentation on which the operations are effectively continuous. This will be shown in Propositions 32 and 33. These propositions imply that in our next result, Theorem 37, it does not matter if we use sequential computability or effective continuity.

Definition 31. A discrete countable group G is

- *computable* if the domain of G is a computable set of natural numbers and the operations are computable functions on this set;
- *c.e. presented* if $G \cong H/K$, where H is a computable countable discrete group and $K \trianglelefteq H$ is c.e. as a subset of H;

- *co-c.e presented* if $G \cong H/K$, where H is a computable countable discrete group and $K \trianglelefteq H$ is co-c.e. as a subset of H.

Note that for the following propositions we do not assume our groups to be commutative. As such we use multiplicative notation.

Proposition 32. *Let G be a discrete countable group. The following are equivalent.*

1. *G is co-c.e. presentable.*
2. *There is a metric d such that (G, d) is a left-c.e. Polish space and the group operations are sequentially computable with respect to d.*

Proof. Let G be a discrete countable group with a left-c.e. Polish presentation $(\{\alpha_i\}_{i\in\omega}, d)$ on which the group operations are sequentially computable. Define a co-c.e. presentation of $F/H \cong G$ as follows. Let F be the free group generated by the special points $\{\alpha_i\}_{i\in\omega}$. It is clear that multiplication and the inverse operations are both computable in F and thus F/H. Given two words $u, v \in F$, the multiplication $u \cdot v$ is simply the concatenation uv and taking a simplification using the usual rules. The inverse of a word $u = u_0u_1 \dots u_n$ is simply $u_n^{-1}u_{n-1}^{-1} \dots u_0^{-1}$. The intention here is obviously to map each word $u = u_0u_1 \dots u_n$ to the product $u_0 \cdot u_1 \cdot \dots \cdot u_n \in G$. Rather than defining and proving that H is a co-c.e. subset of F, we instead show the equivalent condition that '$=$' is co-c.e..

Since G is a discrete countable group, then there must be some $r > 0$ such that $d(e, g) > r$ for any $g \neq e$. Non-uniformly fix a special point that represents e and also the isolating radius r. To tell if two words, $u = u_0u_1 \dots u_n$ and $w = w_0w_1 \dots w_m$ are equal, do the following. Since the group operations are assumed to be sequentially computable with respect to d, then we are able to produce a Cauchy name for

$$uw^{-1} = u_0 \cdot u_1 \cdot \dots \cdot u_n \cdot w_m^{-1} \cdot w_{m-1}^{-1} \cdot \dots \cdot w_0^{-1}.$$

Recall that each letter u_i and w_j are special points of G. We can use the constant names as the required input to produce a name for uw^{-1}. Let

$(\beta_i)_{i\in\omega}$ be a Cauchy name in d for uw^{-1}. Compute the name for uw^{-1} up to accuracy $r/2$; β_k such that $d(\beta_k, uw^{-1}) < r/2$. Note that d is not computable, however, since $(\beta_i)_{i\in\omega}$ is a name for uw^{-1}, then k can be found computably in r. Once such a β_k is found, enumerate the left cut of $d(\beta_k, e)$. Note that $d(\beta_k, e) > r/2$ iff $d(uw^{-1}, e) > r$. In other words, $d(\beta_k, e) > r/2$ iff $u \neq w$. Thus, it is co-c.e. to tell if '$u = w$'.

Now suppose that G is co-c.e. presentable. To construct a Polish presentation (G, d_1), let u, w be two words in $F/H \cong G$. Define the metric $d_1(u, w) = 0$ iff $uw^{-1} \in H$, and $d_1(u, w) = 1$ otherwise. Since the operations in F are computable and H is a co-c.e. subset, then d_1 is evidently a left-c.e. metric on G. Construct a left-c.e. presentation of (G, d_1) by letting every word $w \in F$ be a special point. It remains to check that the group operations are sequentially computable with respect to the presentation (F, d_1). Let the names of two points α, β be given, say $(\alpha_i)_{i\in\omega}$ and $(\beta_j)_{j\in\omega}$. Any two points in our presentation are either distance 1 apart, or must be equal, and for a Cauchy name of α we have that $d(\alpha_j, \alpha) < 2^{-j}$, so it must be that $\alpha_1 = \alpha$, and similarly $\beta_1 = \beta$. As such, given two names, define the output of the operation to be the constant name $(\alpha_1\beta_1)_{n\in\omega}$. This is a Cauchy name for $\alpha\beta$, and so the operation is sequentially computable. A similar argument can be applied for the inverse operation. Therefore, if G is co-c.e. presentable, then (G, d_1) is a left-c.e. Polish space on which the group operations are sequentially computable. □

Proposition 33. *If a discrete countable group has a co-c.e. presentation, then its domain has a left-c.e. Polish presentation on which the group operations are effectively continuous.*

Proof. Suppose that G has a co-c.e. presentation, F/H. Define a left-c.e. Polish presentation of (G, d_1) on which the group operations are effectively continuous as follows. Let each word of F be a special point of the left-c.e. Polish presentation. As before, define $d_1(u, w) = 0$ if $u = w$ and 1 otherwise. Since the relation '=' is co-c.e., the metric d_1 defined on the words of F is a left-c.e. metric. (Note that this is the same left-c.e. Polish presentation as constructed in the Proposition 32.)

Let $B(w, r)$ be given. To enumerate the preimage of $B(w, r)$ under $^{-1}$, do the following. If $r \geq 1$, then we enumerate every basic open ball $B(u, q)$ for each word $u \in F$ and each $q \in \mathbb{Q}^+$. If $r < 1$, then using the assumption that the group operations are computable, compute w^{-1} (in F) and enumerate the ball $B(w^{-1}, r)$ for each $r < 1$ into the preimage of $B(w, r)$ under $^{-1}$. Note that this gives exactly the desired preimage.

In order to show that $\cdot$ is effectively continuous, consider the function $f(x, y) = x \cdot y^{-1}$. Let $B(w, r)$ be given. For each word $u \in F$, compute $f(w, u)$. By the assumption that the group operations are computable, it follows that f is also computable. Once $f(w, u) \downarrow= v$, then enumerate the open ball $B(v, p) \times B(u, q)$ for each rational $0 < p, q < 1$ into the preimage of $B(w, r)$ under $\cdot$. We now check that this procedure produces the preimage. If $B(v, p) \times B(u, q)$ is ever enumerated into the preimage, then it must be that $f(w, u) = w \cdot u^{-1} = v$ which gives $w = v \cdot u$. That is, every ball enumerated must be contained in the preimage. Now let v, u be such that $v \cdot u = w$. We note here that given the words w and u, the word given by $w \cdot u^{-1}$ need not literally be v. However, it must give some word v' which is equivalent to v. Then the procedure must have enumerated the ball $B(v', p) \times B(u, q)$ for each rational $0 < p, q < 1$ into the preimage of $B(w, r)$. In other words, $(v, u) = (v', u)$ is contained in the preimage. □

Since we are dealing with discrete groups, and constructing c.e. and co-c.e. presentations for these groups, for our second main result it does not matter whether we choose sequential computability or effective continuity. In the following definition, we take all our groups and spaces to be discrete and countable.

Definition 34. A Polish group $(G, \cdot, ^{-1})$ is *computably presented* if the domain is a computable Polish space and the operations are effectively continuous. Similarly, if the domain is a left-c.e. presentable (right-c.e. presentable)

Polish space, and the operations are effectively continuous, then we say that the Polish group is left-c.e. (right-c.e.) presented. We call such Polish groups computable, left-c.e. or right-c.e. respectively.

In order to show our second main result in the next section, we recall a couple of lemmas from [45].

Lemma 35. *(Lemma 6.1 in [45]) A discrete countable group is computably presentable iff it is a computably presented Polish group.*

Lemma 36. *(Lemma 6.2 in [45]) A discrete countable group is c.e. presentable iff it is a right-c.e. presented Polish group.*

2.3 An Almost Computable Polish Group

As discussed in the previous section, the following result holds regardless of whether we require the operations to be sequentially computable or effectively continuous. This is because the group G that we construct is discrete and has both c.e. and co-c.e. presentations.

Theorem 37. *There is a Polish group G that has a left-c.e. presentation, a right-c.e. presentation, but no computable presentation up to topological isomorphism.*

Proof. We start by showing that a particular kind of group constructed from any Δ^0_2 set S always has a co-c.e. presentation and a c.e. presentation, and then by taking an infinite S that is Δ^0_2 but not limitwise monotonic we get such a group that has no computable presentation. In the following lemmas, we fix p to be a prime number.

Lemma 38. *A group $G_S \cong \bigoplus_{n \in S} \mathbb{Z}_{p^n} \oplus \bigoplus_{\omega} \mathbb{Z}$ has a c.e. presentation for any Δ^0_2 set S.*

Proof. We have that S is Δ^0_2 so $S = \operatorname{rng}(\lim_s f(x,s))$ for f computable and the limit exists for every x. We construct a c.e. presentation of $G_S = H/K$.

Take $H = \bigoplus_\omega \mathbb{Z}$. In order to turn the i^{th} copy of $\mathbb{Z}$ in our group into a copy of $\mathbb{Z}_{p^n}$, we will enumerate elements into K of the form $(0, \ldots, 0, mp^n, 0, \ldots)$, $m \in \mathbb{Z}$ where the term mp^n is in the i^{th} entry and all other entries are 0. This corresponds to turning the i^{th} copy of $\mathbb{Z}$ into $\mathbb{Z}_{p^n}$ and leaving all other groups in the direct sum unaffected. We also need to ensure that at the end we have infinitely many copies of $\mathbb{Z}$ so that our presentation is a presentation of G_S. To do this, we will keep all even indexed groups in the product as copies of $\mathbb{Z}$, and encode S into the odd indices. Fix some enumeration of S such that for all $t \in \omega$, at most one element either enters or leaves S at stage t.

Construction.

Stage 0 Let $H = \bigoplus_\omega \mathbb{Z}$. Enumerate the all-zero sequence into K.

Stage t For each element currently in K, say the elements currently in K are $x_1, x_2, \ldots, x_m$, enumerate $\sum_{i=1}^m \sigma(i)(x_i)$ into K, where σ runs over all binary strings of length m. Then consider the following cases.

Case 1 If n enters S at stage t, we turn the $(2t)^{th}$ copy of $\mathbb{Z}$ into $\mathbb{Z}_{p^n}$. That is, for all $s \geq 0$, at stage $t+s$, enumerate $(0, 0, \ldots, (s+1)p^n, 0, \ldots)$ and $(0, 0, \ldots, -(s+1)p^n, 0, \ldots)$ into K, where the only non-zero entry is at index $2t$.

Case 2 If n leaves S at stage t, let t_0 be the stage at which n was last enumerated into S, and turn the $(2t_0)^{th}$ group in the direct sum into the trivial group. That is, for all $s \geq 0$, at stage $t+s$, we enumerate for each $1 \leq i < p^n$ $(0, 0, \ldots, i + sp^n, 0, \ldots)$, and $(0, 0, \ldots, -(i + sp^n), 0, \ldots)$ into K.

By construction, H is a computable group, and K is computably enumerable. To see that K is a subgroup, notice that the identity of H is enumerated into K at stage 0, and all finite sums of elements are enumerated eventually, making it closed under the group operation. Finally, since the inverse of an element x is always enumerated at the same stage as x, K is also closed under inverses. Thus K is a subgroup, and since H is abelian, then H/K is c.e. presentable. Furthermore, since S is Δ^0_2, $n \in S$ iff $\exists T \in \omega$ s.t. $\forall t \geq T$, $n \in S_t$. Then pick the least such T and notice that the $2T^{th}$ group in the direct sum is isomorphic to $\mathbb{Z}_{p^n}$ as every multiple of p^n is eventually enumerated into K (at the $2T^{th}$ coordinate) by maintaining the procedure in case 1. Similarly, if $n \notin S$, then after some stage T, n will not show up again in S_t for all $t \geqslant T$, and so no new copy of $\mathbb{Z}_{p^n}$ will be created (of course if n never entered S in the first place, no copy was created to begin with). Also see that each time n leaves S at some stage t, the copy of $\mathbb{Z}_{p^n}$ being maintained is then turned into the trivial group as in case 2 of the above construction. Finally, since only even-numbered copies of $\mathbb{Z}$ are turned into $\mathbb{Z}_{p^n}$ or the trivial group, there will still be infinitely many copies of $\mathbb{Z}$ that are left untouched. $\square$

Lemma 39. *A group $G_S \cong \bigoplus_{n\in S} \mathbb{Z}_{p^n} \oplus \bigoplus_\omega \mathbb{Z}$ is co-c.e. presentable for any Δ^0_2 set S.*

Proof. We have that S is Δ^0_2 so $S = \text{rng}(\lim_s f(x,s))$ where the limit exists for every x and f is computable. We construct a co-c.e. presentation of G_S by constructing an H and K and take $G_S = H/K$. Initially, we take $K = H$ and throughout we will remove elements from K (K is co-c.e.). In order to turn the i^{th} copy of $\mathbb{Z}$ in our group into a copy of $\mathbb{Z}_{p^n}$, we will enumerate elements of $H = \bigoplus_\omega \mathbb{Z}$ that do not have a multiple of p^n in the i^{th} entry into $H \setminus K$. We also need to ensure that at the end we have infinitely many copies of $\mathbb{Z}$ so that it is isomorphic to G_S. To do this, we will build all even indexed groups in the product as copies of $\mathbb{Z}$, and encode S into the odd indices. Fix some enumeration of S such that for each $t \in \omega$ at most one element either enters or leaves S at stage t.

Construction.

Stage 0 Begin with $H = \bigoplus_\omega \mathbb{Z}$. We enumerate elements into $H \setminus K$. Since H is a direct sum, our elements are infinite sequences with finitely many non-zero entries.

Stage t Enumerate sequences $(j_0, 0, j_1, 0, \dots, 0, j_t, 0, 0, 0, \dots)$ into $H \setminus K$ where $\langle j_i \rangle_{0 \leq i \leq t}$ are all possible t-tuples which are not identically 0 containing entries at most t. Take all finite sums of distinct elements $x_1, x_2, \dots, x_m$ currently in $H \setminus K$, such that if $i \neq j$, then x_i and x_j do not both have a non-zero entry at the same index for any index of the sequence. This is to construct the copies of $\mathbb{Z}$, ensuring we do not enumerate the all-zero sequence into $H \setminus K$. Then consider the following cases.

Case 1 If n enters S at stage t, we turn the $(2t+1)^{th}$ group into $\mathbb{Z}_{p^n}$. That is, for all $s \geq 0$, at stage $t+s$, enumerate $(0, 0, \dots, i + sp^n, 0, \dots)$ and $(0, 0, \dots, -(i + sp^n), 0, \dots)$ for all $1 \leq i < p^n$ into $H \setminus K$ where the only non-zero entry is at index $2t+1$.

Case 2 If n leaves S at stage t, let t_0 be the stage at which n was last enumerated into S, and turn the $(2t_0+1)^{th}$ group in the direct sum into a copy of $\mathbb{Z}$. That is, for all $s \geq 0$, at stage $t+s$, enumerate $(0, 0, \dots, (s+1)p^n, 0 \dots)$, and $(0, 0, \dots, -(s+1)p^n, 0 \dots)$ into $H \setminus K$, where the non-zero entry is at the $(2t_0+1)^{th}$ index.

Again, H is a computable group, and $H \setminus K$ is computably enumerable. If $n \in S$, then take the largest stage t such that $n \in S_t \setminus S_{t-1}$, that is, the last stage t at which n entered S. Up until now, no elements with a non-zero entry at index $2t+1$ have been enumerated into $H \setminus K$, so up until now the $(2t+1)^{th}$ coordinate of the structure just looks like the trivial group. From this stage on, by case 1 of the construction, we enumerate every element with non-zero entries in index $2t+1$, where the entries are not multiples of p^n, into $H \setminus K$. During the summation step at each stage t, because we do not add elements that share a non-zero entry in the same index, we cannot create the all-zero sequence or an element $(\dots, mp^n, \dots)$ for some $m \neq 0$, with mp^n in the $(2t+1)^{th}$ entry. In this way we create a copy of $\mathbb{Z}_{p^n}$ in the $(2t+1)^{th}$ coordinate. If n enters S at stage t' and later leaves S, then case 1 continues to enumerate the elements that would make the $(2t'+1)^{th}$ coordinate into a copy of $\mathbb{Z}_{p^n}$, but now case 2 enumerates the remaining elements (the multiples of p^n), turning the $(2t'+1)^{th}$ coordinate into a copy of $\mathbb{Z}$.

The only entries that are not enumerated into $H \setminus K$ are the all-zero sequence and elements of the form $(\dots, mp^n, \dots)$ where the entry mp^n corresponds to the location of a $\mathbb{Z}_{p^n}$ group in H/K (and sums of such elements). Hence K contains only and all such elements, making K a subgroup and H/K co-c.e. presentable.

All even indices in the direct sum correspond to copies of $\mathbb{Z}$ and some of the odd indices are the trivial group, some are $\mathbb{Z}$, and infinitely are various $\mathbb{Z}_{p^n}$ for different n. In particular, if $n \in S$ then the $(2t+1)$-th index has a $\mathbb{Z}_{p^n}$, for t the last stage at which n was enumerated into S. Furthermore, if $n \notin S$, then there will not be any copy of $\mathbb{Z}_{p^n}$ in H/K. Thus $H/K \cong G_S$, so G_S is co-c.e. presentable. $\square$

Remark 40. *Note that Lemmas 38 and 39 also hold for Σ^0_2 sets.*

Proposition 41. *There is an abelian group G_S that has both c.e. and co-c.e. presentations but no computable presentation.*

Proof. Consider $G_S \cong \bigoplus_{n\in S} \mathbb{Z}_{p^n} \oplus \bigoplus_{\omega} \mathbb{Z}$ where S is an infinite Δ^0_2 set that is not limitwise monotonic and p is a prime. It is known [55, 41] that $\bigoplus_{n\in S} \mathbb{Z}_{p^n}$ has a computable presentation iff S is limitwise monotonic. By taking S not limitwise monotonic, we know that G_S is not computable. Using Lemmas 38 and 39, G_S has a c.e. presentation and a co-c.e. presentation. $\square$

To complete the proof of Theorem 37, take G_S from Proposition 41, that is, a group with both c.e. and co-c.e. presentations but no computable presentation. Apply Lemma 35 to G_S to obtain that it has no computable Polish presentation, Lemma 36 to get that it has a right-c.e. Polish presentation, and finally Proposition 33 to obtain that G_S has a left-c.e. Polish presentation. To obtain the same result but with a definition that uses sequentially computable operations instead of effectively continuous operations, apply Proposition 32 instead of Proposition 33 to get that G_S is left-c.e. with sequentially computable operations. As discussed, the other two Lemmas are still applicable when using sequentially computable operations. Thus we obtain a Polish group G_S with a left-c.e. presentation, a right-c.e. presentation but no computable presentation, up to topological isomorphism. $\square$

Chapter 3

Q-degrees

3.1 Existing literature

The structure of the Q-degrees has been studied by Omanadze [66, 61, 62] since the 1970s, as well as Downey, LaForte and Nies [17], Batyrshin [4, 5, 1], Soloviev [73], Arslanov [2, 1], and others. There are also a number of variants of Q-reducibility that have emerged, and seem to be gaining more interest [11, 66, 67, 65, 63, 13, 12]. Omanadze wrote a comprehensive survey of results about the Q-degrees in 2003 [62]. We will give a brief outline of some of the work done on Q-degrees to date.

Earlier studies on the Q-degrees were mostly concerned with structural problems, with few results comparing Q-reducibility to other reducibilities. For instance, Omanadze studied nowhere simple sets. A c.e. set A is *nowhere simple* if for every c.e. set B with $B \setminus A$ infinite, there is an infinite c.e. set $W \subseteq B \setminus A$. Shore proved in 1978 [72] that every c.e. Turing degree contains a nowhere simple set. In 1987 Omanadze proved that every non-computable c.e. Turing degree contains a c.e. set whose Q-degree contains neither simple nor nowhere simple sets. He also showed that any c.e. set that is Q-reducible to a nowhere simple set is itself nowhere simple. As a result, every c.e. set contained in the Q-degree of a nowhere simple set is nowhere simple. That is, a c.e. Q-degree consists either entirely of nowhere simple sets, or does not contain any nowhere simple sets. Further, to obtain a set that contains neither simple nor nowhere simple sets, one can take $A \oplus B$ where A is not a nowhere simple set and B is a nowhere simple set, and the Q-degrees of A and B are incomparable. Much more recently, in 2019, Omanadze [64] showed that every non-computable c.e. wtt-degree contains a c.e. set A such that the Q-degree of A contains neither simple nor nowhere simple sets.

A number of other structural problems were also studied, as well as various complexity problems for the c.e. Q-degrees. These are discussed in Omanadze's survey [62]. Then in 1984 Omanadze studied maximal sets, and the relationship between m-degrees and Q-degrees of maximal sets. He found that the Q-degree of a maximal set is not the least upper bound of any pair of incomparable Q-degrees, and that among all m-degrees contained in the Q-degree of a maximal set there is a least m-degree. Further, the Q-degree of a maximal set does not contain any c.e. semirecursive sets. Complexity problems of c.e. sets have also been studied. In particular, Blum and Marques introduced the notions of subcreative and effectively speedable sets in 1973. A number of interesting results about these sets and Q-reducibility have been shown, including by Gill and Morris in 1974 and by Omanadze in 1987 and 1988 [62].

A particularly interesting result is that of Fischer and Ambos-Spies in 1987 [26], who showed that the c.e. Q-degrees are not distributive, and also are not a lattice. This is in contrast to the wtt-degrees which form a dense, distributive upper semilattice. The c.e. Q- and sQ-degrees are also dense, with the sQ-degrees density result being shown by Omanadze in 1991 [61]. The result for the density of the Q-degrees was obtained by Downey, LaForte and Nies in 1998 [17], and required a much more difficult proof than the sQ-degrees density result.

Since the 1990s, more studies have been concerned with the relationship of the Q- and sQ-degrees to other strong reducibilities, though there are still relatively few results about this. For example, in 1991 Omanadze showed that every non-computable c.e. wtt-degree contains infinitely many pairwise sQ-incomparable c.e. sets. Along a similar vein, he showed in 1995 that if A, B are c.e. sets, where $A \equiv_{sQ} B$ and $A <_m B$, then the sQ-degree of A contains infinitely many pairwise m-incomparable c.e. sets. This is in some sense opposite to the notion of contiguity, which has classically been studied for wtt-degrees within Turing degrees. We say that a c.e. Turing degree is *contiguous* if it contains a single c.e. wtt-degree (that is, all the sets in the Turing degree are also wtt-equivalent), and that a c.e. Q-degree is contiguous if it contains a single sQ-degree. In 1994 Omanadze showed that a c.e. Turing degree $\mathbf{a}$ is contiguous if and only if all semirecursive sets contained in $\mathbf{a}$ are sQ-equivalent. That is, there are interesting interactions between the Q- and sQ-degrees and the wtt- and Turing degrees. Around the same time Omanadze also showed that there is a c.e. set which is simultaneously Q- and wtt-complete but not sQ-complete.

In 1998 Downey, LaForte and Nies [17] studied the Q-degrees and obtained a number of interesting results. For example, they showed that there is a non-computable c.e. set A and a c.e. set B with $A \equiv_T B$ such that A and B form a minimal pair in the Q-degrees. They also showed that the elementary theory of the upper semilattice of Q-degrees has an undecidable first order theory, amongst other results.

As well as c.e. Q-degrees, n-c.e. Q-degrees have also been studied, for example by Arslanov and Omanadze in 2007 [2]. They proved that if n is even, then there is an n-c.e. set whose Q-degree does not bound any Q-degree of a non-computable c.e. set; but if n is odd, then every properly n-c.e. set Q-computes a non-computable c.e. set. They also discuss density: they show that the Q-degrees of properly n-c.e. sets are dense in the Q-degrees of c.e. sets, but that the Q-degrees of c.e. sets are not dense in the Q-degrees of d.c.e. sets. Arslanov, Batyrshin and Omanadze in 2008 [1] studied the distribution of incomparable Q-degrees, and minimal pairs. In particular, they showed that there is a c.e. Q-degree that is not half of a minimal pair.

Isolated and non-isolated degrees where studied in 2009 by Batyrshin [4]. A degree $\mathbf{d}$ is called *isolated from below* if there is a c.e. degree $\mathbf{b} <_Q \mathbf{d}$ such that for all c.e. degrees $\mathbf{a}$, if $\mathbf{a} \leqslant_Q \mathbf{d}$, then $\mathbf{a} \leqslant_Q \mathbf{b}$. Otherwise, it is called *non-isolated from below.* Similarly, $\mathbf{d}$ is called *isolated from above* if there is a c.e. degree $\mathbf{b} >_Q \mathbf{d}$ such that for all c.e. degrees $\mathbf{a}$, if $\mathbf{a} \leqslant_Q \mathbf{d}$, then $\mathbf{b} \leqslant_Q \mathbf{a}$, otherwise $\mathbf{d}$ is called *non-isolated from above.* Batyrshin proved that non-isolated from below 2-c.e. Q-degrees are dense in the structure of c.e. Q-degrees, and that below any c.e. Q-degree there is a 2-c.e. Q-degree, which is non-isolated from below and from above.

Batyrshin [5] also studied the relationship between Q- and m-reducibilities in 2014. He proved that there exists a non-computable and m-incomplete c.e. set B such that for any c.e. set A, $A \leqslant_Q B$ implies $A \leqslant_m B$. Such a degree is called *m-topped*, so Batyrshin showed the existence of a non-computable and m-incomplete m-topped Q-degree. He also showed that for any c.e. non-computable set A there exists a c.e. set B such that $A \leqslant_Q B$ but $A \nleqslant_m B$, and that for any simple set B there is a c.e. set A such that $A \leqslant_Q B$ but $A \nleqslant_m B$. It follows that the Q-degree of any simple set contains infinitely many c.e. m-degrees.

We note that the structure of the Q-degrees and their relationship to other reducibilities has not been studied nearly as extensively as some of the other reducibilities and degree structures. In this chapter we study the structure of the Q- and sQ-degrees and their relationship to other reducibilities.

3.1.1 Our results

Since $\leqslant_{sQ}$ implies $\leqslant_{wtt}$, we might have anticipated there to be similarities between the wtt-degrees and the sQ-degrees. However, in this chapter we show that the c.e. sQ-degrees are not distributive, and that we can embed the lattice N_5 into them. We show that the c.e. sQ- and Q-degrees don't have any initial segments that form a lattice. In contrast, the c.e. wtt-degrees do have initial segments that form lattices, as Fischer showed in 1986 [25]. This is particularly surprising because the proof for the wtt-degrees seems to mainly rely on the fact that the uses are computably bounded, which we also have in the sQ-degrees. The difference comes in permitting: in our argument we need to use a more complicated ‘permitting’ technique customized to work in the sQ-degrees, because permitting in general fails badly. In particular, we show that there is a non-computable c.e. set with no c.e. simple set Q-below it. We show this directly through a construction, although it is also implied by Omanadze’s result about nowhere simple sets.

Studying the relationship between wtt-degrees and sQ-degrees, we show that the sQ-degree of a semirecursive set A is maximal among the sQ-degrees of c.e. sets of the same wtt-degree as A, and that if $A \equiv_{wtt} B$ and the infimum of A and B exists in the Q- or sQ-degrees, it will have the same wtt-degree as A.

We also study minimal pairs and half minimal pairs, and show that there is a minimal pair of sQ-degrees within the same Q-degree. Finally, we prove that if the degree of $\mathbf{b}$ is half of a minimal pair in the Q-degrees, it is also half of a minimal pair in the Turing degrees.

3.2 Lattices

Theorem 42. *The c.e. Q- and sQ-degrees are not distributive.*

Proof. We show this for sQ-reductions and note that it works for Q too. We build c.e. sets A_1, A_2 and $B \leqslant_{sQ} A_1 \oplus A_2$ via q and meet requirements

R_e : If $U_e \oplus V_e \leq_{sQ} B$ via m_e and $B \leq_{sQ} U_e \oplus V_e$ via n_e then for all i, $R_{e,i}$.

$$R_{e,i}: \ U_e \not\leq_{sQ} A_1 \text{ via } \varphi_i \text{ or } V_e \not\leq_{sQ} A_2 \text{ via } \psi_i.$$

Note that these $R_{e,i}$ requirements suffice, because the pair $V_e \oplus U_e$ will show up as some other $R_{e',i'}$ requirements, with $U_{e'} = V_e$ and $V_{e'} = U_e$, so we will not have that $U_e \leqslant_{sQ} A_2$ and $V_e \leqslant_{sQ} A_1$.

We use a tree and meet R_e at nodes τ during expansion stages, where the length of agreement $\ell(e, s)$ between $U_e \oplus V_e$ and B via m_e and n_e increases. The tree is sculpted so that we have all the $R_{\tau,i}$-nodes σ devoted to meeting $R_{e,i}$ below $\tau\widehat{\ }\infty$. At a $\tau\widehat{\ }\infty$ stage we activate the subrequirements $R_{\tau,i}$, and these $R_{\tau,i}$-nodes σ have two outcomes, $1 <_L 0$, with 1 meaning we have digonalized against φ_i or ψ_i.

Basic Strategy for $R_{\tau,i}$. Pick a fresh follower x targeted for B and set $q(x,s) = 2a$ for some a, so x is pointing at A_1, and wait for a stage $t \geqslant s$ where $\ell(e,t) > x$. Now at stage t we know that $n_e(x,t)$ is defined, and if x enters B, $n_e(x,t)$ must enter U_e (if $n_e(x,t)$ is pointing at an element of U_e) or V_e (if it is pointing at V_e).

Suppose that $n_e(x,t)$ is pointing at U_e. Now we wait for the length of agreement $\ell(e,i,s')$ associated with $R_{e,i}$ (between U_e and A_1 via φ_i and between V_e and A_2 via ψ_i) to go above $n_e(x,s')$, so that $\varphi_i(n_e(x,s'),s')$ must be defined and pointing at an element that is not currently in A_1. If $\varphi_i(n_e(x,s'),s') \neq q(x,s') = 2a$, then we win by putting x into B, $q(x,s')$ into A_1 and keeping $\varphi_i(n_e(x,s'),s')$ out of A_1. Otherwise if $\varphi_i(n_e(x,s'),s') = q(x,s')$, then we put $q(x,s')$ into $A_{1,s'+1}$ and define $q(x,s'+1)$ to be something new and odd, that is, targeted for A_2. Now if φ_i is to recover, at the next expansionary stage $t' > s'$ it will have had to retarget, so $\varphi_i(n_e(x,s'),t')$ is an element not currently in A_1. Then we win by putting x into B and $q(x,t')$ into A_2, and restraining $\varphi_i(n_e(x,s'),t')$ out of A_1. Then $R_{\tau,i}$ is met. If instead $n_e(x,t)$ is pointing at V_e, then when the length of agreement $\ell(e,i,s') > n_e(x,s')$, we have that $\psi_i(n_e(x,s'),s')$ is defined and pointing at an element not currently in A_2. This element cannot possibly be $q(x,s')$ (which is targeted for A_1), and so we treat this like the first case above: put x into B and $q(x,s')$ into A_1 and restrain $\psi_i(n_e(x,s'),s')$ out of A_2. Now $R_{\tau,i}$ is met.

Definition 43. We say a requirement $R_{\tau,i}$ at node σ *requires attention* at stage s if s is a σ-stage (so is e-expansionary), $R_{\tau,i}$ has not been met yet and one of the following holds.

(i) $R_{\tau,i}$ has no follower.

(ii) $R_{\tau,i}$ has a follower x and $\ell(e,s) > x$ and $\ell(e,i,s) > n_e(x,s)$.

Construction. At stage s, compute TP_s. Initialise all requirements R_τ where $\tau \not\leqslant_L TP_s$. Find the highest priority requirement $R_{\tau,i}$ that requires attention at stage s. If $R_{\tau,i}$ has no follower, assign a fresh follower x and define $q(x,s) = 2a$ (so targeted for A_1) for some fresh number a. Otherwise, $R_{\tau,i}$ already has a follower x, $n_e(x,s)$ is defined and targeted at U_e (or at V_e), and $\varphi_i(n_e(x,s))$ (or $\psi_i(n_e(x,s))$ in the case that n_e is targeted at V_e) is defined and pointing at A_1. If $\varphi_i(n_e(x,s),s) = q(x,s)$, then we put $q(x,s)$ into $A_{1,s+1}$ and define $q(x,s+1)$ to be something new and odd, so targeted for A_2. Otherwise, $\varphi_i(n_e(x,s),s) \neq q(x,s)$ (respectively, $\psi_i(n_e(x,s),s) \neq q(x,s)$), then we put x into B, $q(x,s)$ into A_1 or A_2 respectively, whichever it is targeted for (depending on if we have taken action for this requirement already and retargeted $q(x)$ to A_2 or not), and restrain $\varphi_i(n_e(x,s),s)$ (respectively, $\psi_i(n_e(x,s),s)$) out of A_1. Initialise all lower priority requirements.

Verification. Let the true path TP be the leftmost path visited infinitely often. We show that all R_e are met. Let τ be the node on the true path assigned to R_e. If $\tau\hat{\ }f$ is on the true path there are only finitely many e-expansionary stages, so $U_e \oplus V_e \not\equiv_{sQ} B$ and hence R_e is met. Suppose $\tau\hat{\ }\infty$ is on the true path. Then we claim all $R_{\tau,i}$ are met, and thus R_e is met. Go to a stage s after which TP_t, $t \geqslant s$ is never to the left of τ, so R_τ has priority. Such a stage exists because each of the finitely many requirements above R_τ requires attention and acts a finite number of times before being met. Go to the first stage $t \geqslant s$ where all requirements above $R_{\tau,i}$ have finished acting, so now $R_{\tau,i}$ has priority and requires attention. Then $R_{\tau,i}$ receives a fresh follower x at stage t, and this follower is never initialised. Since there are infinitely many expansionary stages, eventually we will have $\ell(e,s) > x$ and so $n_e(x,s)$ is defined. Now monitor the length of agreement $\ell(e,i,t)$, $t \geqslant s$. If $\ell(e,i,t)$ is never bigger than $n_e(x,s)$, then $R_{\tau,i}$ is met because φ_i or ψ_i got stuck. Otherwise, at some stage $t > s$ we have $\ell(e,i,t) > n_e(x,s)$, and at this stage $R_{\tau,i}$ again is the highest priority requirement that requires attention. Now as per the basic strategy, either $R_{\tau,i}$ is satisfied at stage t, or it will be satisfied at the next e-expansionary stage and does not require attention again. □

Theorem 44. *N_5 embeds into the c.e. Q- and sQ-degrees.*

Proof. We build c.e. sets A, B and $B \oplus C$ forming the relevant degrees. The negative requirements are

$$N_e^1 : \Phi_e^A = \Psi_e^B = f \text{ implies } f \text{ is computable.}$$

$$N_e^2 : \Phi_e^A = \Psi_e^{B \oplus C} = f \text{ implies } f \text{ is computable.}$$

These are standard minimal pair requirements that monitor the length of agreement $\ell_1(e, s) = \max\{z : \forall x \leqslant z, \Phi_e^{A_s}(x) \downarrow = \Psi_e^{B_s}(x)\}$ for N_e^1, and similarly for N_e^2 we have the length of agreement $\ell_2(e, s)$ between A and $B \oplus C$ via Φ_e and Ψ_e. We also have positive requirements

$$P_e : \neg(C \leqslant_Q B \text{ via } m_e).$$

$$Q_e : \overline{A} \neq W_e.$$

$$R_e : \overline{B} \neq W_e.$$

Finally, we need the overall *coding* requirements

$$A \oplus B \leqslant_Q A \oplus (B \oplus C) \text{ via } q_1.$$

$$A \oplus B \geqslant_Q A \oplus (B \oplus C) \text{ via } q_2.$$

Note that simply making q_1 the identity will satisfy our first coding requirement. The Q_e, R_e are met by standard Friedberg strategies cooperating with minimal pair requirements. The minimal pair requirements monitor the length of agreement between A and B (respectively, $B \oplus C$), and only let elements enter one side at a time. That is, during an expansion stage things below the length of agreement are allowed to enter one side only, say A, and the other side (B or $B \oplus C$) is restrained to preserve the current computation. Then at the next expansionary stage A must have recovered its computation and at this stage we can again put elements into one of A or

B (respectively, $B \oplus C$), while restraining the other. The Friedberg strategies pick a follower x, targeted for A in the case of Q_e, and targeted for B in the case of R_e and wait for x to enter W_e. If x enters W_e, the strategy wants to put x into A (respectively, B). Enumerating x into A (or B) is only allowed at an expansion stage where A (or B) is not restrained by the minimal pair requirements. Here we are only putting elements into A and B, so the second coding requirement can use the identity for the Q-reduction q_2 on these elements.

Finally to satisfy the P_e requirements, consider P_e at some node σ. We pick a fresh follower x targeted for C, and a fresh trace $q_2(x, s)$ targeted for A in the coding of $A \oplus (B \oplus C)$ into $A \oplus B$. Now we wait for $m_e(x, s) \notin B$ to be defined. At the next σ-stage t, we would want to put $q_2(x, s)$ into A and x into C while restraining $m(x, s)$ out of B. However, because of the minimal pair requirements we cannot put elements both into A and C simultaneously. We don't know if σ's guess is correct: it is possible that this is the last time we ever visit σ, in which case if we only put one of $q_2(x, s)$ and x into A or C, our q_2 reduction fails since we never reach another σ-stage to correct it (by putting the other element, x or $q_2(x, s)$, into C or A respectively). To overcome this, we put $q_2(x, s)$ into A, and pick any small number targeted for B that is different to $m_e(x, s)$, and define $q_2(x, t)$ to be that number. Then at the next σ-stage we put x and $q_2(x, t)$ into C and B respectively. For elements in A and B, we can set q_2 to be the identity.

We use a tree and assign all of our requirements to their own nodes: N_e^1 and N_e^2 requirements on nodes τ with outcomes $\infty <_L f$ (with the infinite outcome, as usual, indicating that the relevant length of agreement has increased since the last expansionary stage), and P_e, Q_e and R_e requirements on nodes σ with outcomes $1 <_L 0$, with 1 indicating that the requirement has enumerated a follower x into its target set.

Definition 45. We say that a P_e requirement at node σ *requires attention* at stage s if s is a σ-stage, and one of the following holds.

(i) P_e has no follower.

(ii) P_e has a follower x and $m_e(x,s)$ is defined, $m_e(x,s) \notin B_s$, and $q_2(x,s) \notin A_s \oplus B_s$.

We say that a Q_e (respectively, R_e) requirement at a node σ *requires attention* at stage s if s is a σ-stage, $W_{e,s} \cap A_s = \emptyset$ (respectively, $W_{e,s} \cap B_s = \emptyset$), and one of the following holds.

(i) Q_e (respectively, R_e) has no follower.

(ii) Q_e (respectively, R_e) has a follower x and $x \in W_{e,s}$.

Construction. At stage s, compute TP_s and initialise all requirements at nodes $\tau \nleq_L TP_s$. Find P_σ, R_σ or Q_σ with highest priority that requires attention at stage s. Initialise all requirements at nodes $\tau \nleq_L \sigma$. If the requirement is a Q_σ or R_σ requirement that has no follower, assign it a fresh follower x. If it is a Q_σ or R_σ requirement with follower x, enumerate x into its target set. If it is a P_σ requirement with no follower, assign it a fresh follower x and define x's trace $q_2(x,s) \notin A_s$ to be a fresh number targeted for A. If it is a P_σ requirement with follower x and $q_2(x,s)$ is targeted for A, enumerate $q_2(x,s)$ into A and define $q_2(x,s+1)$ to be any smaller number targeted for B that is not $m_e(x,s)$ (and of course is not in B_s nor a follower for any higher priority requirement). Finally if it is a P_σ requirement with follower x and $q_2(x,s)$ is targeted for B, enumerate x and $q_2(x,s)$ into C and B respectively.

Coherence. As with a standard minimal pair argument, we want to 'nest' our N requirements, so a lower priority N requirement guessing the infinite outcome for some higher priority N' requirement will only consider stages that are N'-expansionary (for instance in its computation of the length of agreement). That way every N-expansionary stage is also an N'-expansionary stage. Then all the N requirements drop their restraint simultaneously, allowing us to enumerate something into either A or into $B \oplus C$ without breaking a restraint.

Verification. Let the true path TP be the leftmost path visited infinitely often.

Lemma 46. *All the P_e, Q_e and R_e requirements have versions that are met, and each P_σ, Q_σ and R_σ acts at most finitely often.*

Proof. We prove this by simultaneous induction on e for P_e and for Q_e, the same argument as for Q_e applies to R_e.

Let P_e be assigned to a node $\sigma \subset TP$. Go to the least σ-stage s_0 where for all $\tau <_L \sigma$ and $s \geqslant s_0$, if $\tau \not\subset \sigma$ then s is not a τ-stage, and any Q, R or P requirement assigned to τ will not act at stage s. Now P_σ requires attention and has priority at stage s_0 so it receives a fresh follower x and defines $q_2(x, s_0)$ to be a fresh number targeted for A. If m_e does not converge on x or if it does but $m_e(x, s) \in B_s$, then P_σ never requires attention again and P_e is met. Otherwise, at the next σ-stage $s_1 > s_0$ (after $m_e(x, s)$ is defined), P_σ again has priority and requires attention. Now $m_e(x, s_1) \notin B_{s_1}$ is defined. Notice that if $m_e(x, s_1)$ is a follower for a higher priority positive requirement, this requirement will never enumerate $m_e(x, s_1)$, and if it is a follower for a lower priority positive requirement, this lower priority requirement is initialised at stage s_1 and its follower is thus cancelled. At stage s_1 P_σ enumerates $q_2(x, s_1) = q_2(x, s_0)$ into A_{s_1+1}, and defines $q_2(x, s_1+1)$ to be a small number different to $m_e(x, s_1)$ targeted for B. At the next σ-stage $s_2 > s_1$, P_σ again has priority and requires attention. Any followers assigned to lower priority positive requirements between stages s_1 and s_2 are fresh big numbers and so are different to $m_e(x, s_1)$ (and are initialised if they were not enumerated before stage s_2), and so $m_e(x, s_2) = m_e(x, s_1) \notin B_{s_2}$. Now P_σ enumerates x into C_{s_2+1} and $q_2(x, s_2)$ into B_{s_2+1}. Since any new follower assigned to a positive requirement hereafter will be fresh, $m_e(x, s_2)$ never gets enumerated, and so P_e is met and P_σ never requires attention again. Note also that P_σ has acted only finitely many times.

Now let Q_e be assigned to a node ρ, with $\sigma \subset \rho \subset TP$. At stage s_2, Q_ρ is initialised, and at stage s_2+1, if Q_e is not already met (i.e. if $W_{e,s_2+1} \cap A_{s_2+1} = \emptyset$) then Q_e requires attention and has highest priority, and so receives a fresh follower x. Note that similar to stage s_0, we also have that for all $\tau <_L \rho$ and $s \geqslant s_2$, if $\tau \not\subset \rho$ then s is not a τ-stage, and any Q, R, or P

requirement assigned to τ does not act at stage s. As such, the follower x is never initialised. If x never enters W_e then P_e is satisfied. Otherwise, if x enters $W_{e,s}$ then at the next ρ-stage $s_3 > s, s_2$, Q_ρ requires attention and has priority, and enumerates x into A_{s_3+1}. Thus P_e is satisfied and P_ρ has acted only finitely often. The same argument now applies to the version of R_e on the true path (which is immediately after Q_ρ). □

Lemma 47. *All the N_e^1 and N_e^2 requirements have versions that are met.*

Proof. We prove this by induction on e for N_e^2, the argument for N_e^1 is similar. Let N_e^2 be assigned to a node $\tau \subset TP$. Go to a stage s_0 where for all $\sigma <_L \tau$ and $s \geqslant s_0$, if $\sigma \not\subset \tau$ then s is not a σ-stage, and any Q, R or P requirement assigned to σ will not act at stage s. That is, we are never again to the left of τ and all higher priority positive requirements have finished acting. If $\tau\widehat{\ }f \subset TP$ then N_e^2 is met, since $\lim_s \ell_2(e,s) < \infty$ so $\Phi_e^A \neq \Psi_e^{B\oplus C}$.

Suppose then that $\ell_2(e,s)$ goes to infinity, so $\tau\widehat{\ }\infty \subset TP$. We claim $\Phi_e^A = \Psi_e^{B\oplus C}$ is computable. To compute $\Phi_e^A(x)$, go to a τ-stage $s > s_0$ where $\ell_2(e,s) > x$. Now $\Phi_e^A(x)[s] = \Phi_e^A(x)$. To see this, note that at stage s of the construction we initialise all requirements at nodes $\sigma \not\leqslant_L TP_s$, and that any new followers assigned will be fresh (large) numbers, and so will not be able to affect the computation of $\Phi_e^A(x)[s]$. Then the only followers that could have been appointed before stage s and not been initialised, and so could enter A, B or C after stage s are followers appointed to requirements at nodes σ where $\tau\widehat{\ }\infty \subset \sigma$. These followers can only be enumerated at $\tau\widehat{\ }\infty$-stages $s' > s_0$, and at each such stage at most one follower is enumerated (or in the case of a P_e requirement, a follower in C and a trace in B). As a result, at any $\tau\widehat{\ }\infty$-stage $s' > s_0$, at most one of A or $B \oplus C$ can change below the use of $\Phi_e^A(x)[s]$, and no further changes occur before the next $\tau\widehat{\ }\infty$-stage. Thus $\Phi_e^A(x)[s] = \Psi_e^{B\oplus C}(x)[s] = \Phi_e^A(x) = \Psi_e^{B\oplus C}(x)$. Note that these are Q-reductions (so we could have cast all this in terms of Q-like functions), although we haven't used anything specific to Q-reductions here and so we could equally make this work for Turing functionals. □

This shows that all our positive and negative requirements are met. The mapping q_1 that takes each element in $A \oplus B$ to itself in $A \oplus (B \oplus C)$ clearly satisfies the first coding requirement. Similarly for elements x in A and B the mapping $q_2(x) = x$ is a valid Q-reduction. For elements x in C, when $q_2(x, s)$ is first defined it is a fresh number in A, and so is not a follower for any other positive requirement. Further, any followers later appointed to positive requirements and targeted for A will be big numbers and so will be different to $q_2(x, s)$. Thus the only requirement that can enumerate $q_2(x, s)$ and x is the P_e requirement for which x is a follower. By the two actions that P_e may take, clearly $\lim_s q_2(x, s) = q_2(x) \in A \oplus B$ iff $x \in C$, and so overall q_2 is a valid Q-reduction witnessing our second coding requirement.

Finally, notice that all of the Q-reductions we constructed are also sQ-reductions, so this result holds in the sQ-degrees as well as the Q-degrees. □

Theorem 48. *There is no nontrivial initial segment in the c.e. (s)Q-degrees that is a lattice.*

Proof. We prove the theorem for Q-degrees and note that the proof also works for sQ-degrees.

Fix a non-computable c.e. set C. We will build A and B below C so that A and B do not have an infimum. We satisfy the requirements

$$R_e : \text{ If } U_e \leq_Q A \text{ via } m_e \text{ and } U_e \leq_Q B \text{ via } n_e$$

then there is a c.e. set $Z \leq_Q A, B$ via q_A and q_B so that for all i, $R_{e,i}$.

$$R_{e,i} : \; Z \not\leq_Q U_e \text{ via } p_i.$$

First consider how to build A and B (without thinking about C). We use a tree and meet R_e at nodes τ during expansion stages, where the length of agreement between $U_e, m_e^{-1}(A)$ and $n_e^{-1}(B)$ increases. At such a $\tau\widehat{\ }\infty$ stage we will build $Z = Z_\tau$. We also need to build Q-reductions q_A and q_B to witness $Z \leqslant_Q A, B$ respectively. The tree is sculpted so that we have the $R_{\tau,i}$-nodes σ devoted to meeting $R_{e,i}$ below $\tau\widehat{\ }\infty$. At σ we have two outcomes $1 <_L 0$, with 1 meaning diagonalization.

Definition 49. A follower x for a requirement $R_{e,i}$ is *realized* if $p_i(x)$ is defined, below the length of agreement, and not in U_e.

Basic Strategy for $R_{\tau,i}$. Assume the length of agreement $\ell(e,s)$ goes to infinity, so R_τ has infinitely many expansionary stages. At a σ stage we pick a fresh follower x targeted for Z, and note that at the next $\tau^\frown\infty$ stage s we will define $q_A(x)[s], q_B(x)[s] = x$. Now $R_{\tau,i}$ wants to use x to diagonalize against p_i and show that p_i is not a Q-reduction witnessing $Z \leqslant_Q U_e$. So $R_{\tau,i}$ waits for a stage s' where $p_i(x)[s']$ is defined. If this never happens, p_i is not total and so cannot be a Q-reduction. If $p_i(x) \in U_e[s']$, then $R_{\tau,i}$ simply keeps x out of Z forever, and succeeds.

Otherwise, at some stage $t > s$ we see that $p_i(x)[t] \leqslant \ell(e,t)$ is defined and not in $U_{e,t}$. Now $R_{\tau,i}$ wants to enumerate x into Z while keeping $p_i(x)$ out of U_e, and at the same time preserving all the Q-reductions. So $R_{\tau,i}$ asserts control of $A_t \upharpoonright t$ and $B_t \upharpoonright t$ to preserve this current computation, using the convention that the stage number t bounds everything. In particular, we are keeping $m_e(p_i(x))$ and $n_e(p_i(x))$ out of A and B respectively, and this forces $p_i(x)$ (which is below the length of agreement) to stay out of U_e.

At the next $\tau^\frown\infty$ stage $v > t$, we finish the stage by putting $x = q_A(x)$ into A_{v+1}. We do nothing else, continuing to preserve the B-side until the next $\tau^\frown\infty$ stage r. At this stage we need to redefine $q_A(x)$. Note that because we froze the B side and r is a $\tau^\frown\infty$ stage, we have that $p_i(x)[r] = p_i(x)[t] \notin U_{e,r}$ and if $m_e(p_i(x))[t]$ entered A (say if it was equal to $q_A(x)$), it must have retargeted in order for the length to have returned, and so the new value $m_e(p_i(x))[r]$ is not in A. Now pick some $a \neq m_e(p_i(x))$ and redefine $q_A(x)[r] = a$.

At the next σ-stage, we put a into A, $q_B(x) = x$ into B and x into Z, while keeping $m_e(p_i(x))$ out of A. This forces $p_i(x)$ to remain out of U_e while x has entered Z, and this disagreement is preserved. Thus we have diagonalized against p_i.

Note that here we did not actually need to change the value of $q_A(x)$ if it already differed from $m_e(p_i(x))$ and could have enumerated x into B and

Z at stage r before, however, once we incorporate permitting we may not be able to enumerate $q_B(x)$ in B immediately, and so this allows for a delay.

Now we need to incorporate permitting from a non-computable c.e. set C.

Basic Strategy for $R_{\tau,i}$, with C-permitting. We will build $A \leqslant_Q C$ via c_A and $B \leqslant_Q C$ via c_B. From our strategy above, we can put our follower x into A if $c_A(x)$ enters C. Knowing that C is non-computable only means that C has to change late, but we have no control over which elements enter C. So $R_{\tau,i}$ will need many followers, one of which will eventually get C-permission. The $R_{\tau,i}$ requirement will work in cycles, and at cycle n it will have followers $x_{k,j}, x^1_{k,j}, x^2_{k,j}$ for each possible $k, j \leqslant n$ configuration. Then the follower $x_{k,j}$ will succeed in diagonalizing against p_i if k enters C and later j enters C. Here $x_{k,j}$ is our initial follower, then once we put $x_{k,j}$ in (with permission), it is possible that $m_e(p_i(x_{k,j}))$ retargets. That is why we have two more followers $x^1_{k,j}$ and $x^2_{k,j}$. If $m_e(p_i(x_{k,j}))$ retargets to a follower, we still have another follower to enumerate instead (also with permission), while letting us keep $p_i(x_{k,j})$ out of U_e.

Initially, we set $c_A(x_{k,j}) = k$ and $c_B(x_{k,j}) = j$, and $q_A(x_{k,j}) = q_B(x_{k,j}) = x_{k,j}$. Then wait until all followers $x_{k,j}$ are realized, that is, $p_i(x_{k,j})$ is defined, below the length of agreement and is pointing at an element not in U_e. Now the strategy waits for permission from C. If $k \leqslant n$ enters C then we enumerate in A all $q_A(x_{k,j}) = x_{k,j}$ for every j, while preserving B, and reset $c_A(x^1_{k,j}) = c_A(x^2_{k,j}) = j$. Now we wait again for the next $\tau\widehat{\ }\infty$ expansionary stage v, so the length of agreement has returned, and reset $q_A(x_{k,j})$ to be one of $x^1_{k,j}$ or $x^2_{k,j}$, which is not $m_e(p_i(x_{k,j}))[v]$. Should j ever enter C, we can enumerate $q_B(x_{k,j}) = x_{k,j}$ into B, $q_A(x_{k,j})$ into A and $x_{k,j}$ into Z, and thus meet $R_{e,i}$ with the disagreement that this creates. While we are waiting for permission from C (either for the first or second time), we start a new cycle $n + 1$ and define a new batch of fresh followers.

Notice that our requirements are pretty much independent of one another. Each follower is a follower for a single requirement only, and when it is assigned it is a fresh big number, and so is bigger than any existing computations that are being preserved. The only interaction that can happen is if some $m_e(p_i(x_{k,j}))$ from requirement $R_{e,i}$ retargets to a follower w of a different requirement $R_{e',j}$. This is not a problem, for if $R_{e',j}$ has lower priority than $R_{e,i}$, this follower w is initialised, and if $R_{e',j}$ has higher priority and at some stage wants to enumerate w, then $R_{e,i}$ will be initialised. Further, each requirement only acts a finite number of times, and so each requirement can only be initialised finitely many times by the finitely many requirements above it.

Definition 50. We say a requirement $R_{\tau,i}$ at node σ *requires attention* at stage s if s is a σ-stage (and thus a $\tau^\frown\infty$ expansionary stage), $R_{\tau,i}$ has not diagonalized against p_i yet, and one of the following holds.

(i) $R_{\tau,i}$ is in cycle n waiting for C-permission, and some $k \leqslant n$ enters C_s.

(ii) $R_{\tau,i}$ is in (or after) cycle n and has previously received a C-permission k and acted for it, so $q_A(x_{k,j}) \in A$ for all $j \leqslant n$ and $q_A(x_{k,j})[s]$ is one of $x^1_{k,j}, x^2_{k,j}$, and now some $j \leqslant n$ enters C_s.

(iii) $R_{\tau,i}$ is in cycle n, all existing $R_{\tau,i}$ followers have been realised and the strategy is waiting for permission from C.

(iv) $R_{\tau,i}$ does not have any followers.

Define TP_s to be the unique σ of length s with s a σ-stage.

Construction. Fix some small $c \notin C$. At stage s, compute TP_s. Initialise all requirements R_τ where $\tau \not\leqslant_L TP_s$. Find the highest priority requirement $R_{\tau,i}$ that requires attention at stage s and act for it according to the case by which it requires attention.

(i) $R_{\tau,i}$ has received C-permission k, i.e. k has entered C_s. Enumerate in A all $q_A(x_{k,j})$ for every j, and assert control of $B_s \restriction s$ to preserve the current computations. Reset $c_A(x^1_{k,j}) = c_A(x^2_{k,j}) = j$. For each j, reset $c_B(x_{j,k}) = c \notin C$. These will not be enumerated because they did not get C-permission j first. At the next $\tau^\frown\infty$ expansionary stage v, reset $q_A(x_{k,j})$ to be one of $x^1_{k,j}$ or $x^2_{k,j}$ which is not $m_e(p_i(x_{k,j}))[v]$.

(ii) $R_{\tau,i}$ has received a second C-permission j after having received C-permission k. Enumerate $q_B(x_{k,j})[s] = x_{k,j}$ into B, $q_A(x_{k,j})[s]$ into A, and $x_{k,j}$ into Z, keeping $m_e(p_i(x_{k,j}))[s]$ out of A. Then for whichever one of $x^1_{k,j}$ or $x^2_{k,j}$ that is not $q_A(x_{k,j})[s]$, redefine $c_A(x^i_{k,j}) = c$. And for the other one, $x^{i'}_{k,j} = q_A(x_{k,j})[s]$, retarget $q_A(x^{i'}_{k,j}) = x^i_{k,j}$. For each k, reset $c_A(x_{j,k}) = c_A(x^1_{j,k}) = c_A(x^2_{j,k}) = c$. These will not be enumerated because they received permissions in the wrong order.

(iii) $R_{\tau,i}$ is in cycle n waiting for C-permission, and all existing $R_{\tau,i}$ followers have been realised. Assuming $n + 1 \notin C$, begin cycle $n + 1$ (otherwise begin cycle $n + 2$ instead) by defining fresh new followers $x_{n+1,j}, x^1_{n+1,j}, x^2_{n+1,j}$ and $x_{k,n+1}, x^1_{k,n+1}, x^2_{k,n+1}$ for each $k, j < n + 1$ that are not already in C. For every new triple of followers $x_{k,j}, x^1_{k,j}, x^2_{k,j}$, define $c_A(x_{k,j}) = c_A(x^1_{k,j}) = c_A(x^2_{k,j}) = k$, $c_B(x_{k,j}) = j$, $c_B(x^1_{k,j}) = c_B(x^2_{k,j}) = c$, and $q_A(x_{k,j}) = x_{k,j}$, $q_B(x_{k,j}) = x_{k,j}$ and similarly for $x^1_{k,j}$ and $x^2_{k,j}$ (point them at themselves in A and B via q_A, q_B).

(iv) $R_{\tau,i}$ has no followers. Begin the first cycle and define the first batch of followers similar to case (*iii*) above.

End of Construction.
Verification. Let the true path TP be the leftmost path visited infinitely often.

Lemma 51. *Every R_e is met. That is, if $U_e \leqslant_Q A$ via m_e and $U_e \leqslant_Q B$ via n_e, then all $R_{e,i}$ are met below some τ node assigned to R_e, and q_A, q_B are valid Q-reductions.*

Proof. Suppose R_e is at a node τ on the true path, and R_τ does not have infinitely many expansionary stages, that is, $\tau^\frown f$ is on the true path TP. Then $\ell(e, s)$ gets stuck and there must be a disagreement somewhere. That is, either $U_e \not\leqslant_Q A$ via m_e, or $U_e \not\leqslant_Q B$ via n_e, and so R_e is met. Notice that in this case it is possible that q_A is not a valid Q-reduction, say if $q_A(x_{k,j})$ is enumerated into A after a first k permission, but there are no further $\tau^\frown\infty$ expansionary stages and so $q_A(x_{k,j})$ is not retargeted and $x_{k,j}$ is not enumerated into Z. However, because the premise of R_e failed to hold, we do not need to build Z or q_A or q_B, so this does not matter.

Suppose R_τ does have infinitely many expansionary stages, so $\tau^\frown\infty$ is on the true path. Go to a stage s after which TP_t, $t \geqslant s$ is never to the left of τ, so R_τ has priority (and is not initialised again). Then for each $R_{\tau,i}$ below τ, we claim $R_{\tau,i}$ is met. There are infinitely many τ expansionary stages, so $R_{\tau,i}$ must eventually receive followers and begin its cycles. Should $R_{\tau,i}$ at any point have a follower x for which p_i is never defined on, or is defined and $p_i(x) \in U_e$, then as discussed in the basic strategy, $R_{\tau,i}$ succeeds by simply doing nothing. In this case x is never enumerated, p_i is either not a Q-reduction or is not valid (it is wrong on x), x is never realised and so $R_{\tau,i}$ succeeds (and stops at some cycle n).

Otherwise, if every follower of $R_{\tau,i}$ gets realised, then $R_{\tau,i}$ eventually has a successful cycle, because C is non-computable. To see this, suppose that we have infinitely many cycles with no success. There are two cases, either infinitely many cycles get one permission but never get the second permission, or only finitely many cycles get one permission. No cycle gets a second permission. Then we claim C is computable. In the first case, to compute $C(i)$, wait for a cycle $k > i$ and one C-permission j for a follower from cycle k. Such a k exists because there are infinitely many cycles that get one permission. From now on $C(i)$ will not change because otherwise cycle k gets a second permission and succeeds. In the second case, to compute $C(i)$ wait for a cycle $k > i$ when there are no more permissions. Then $C(i)$ again does not change from now on, because otherwise cycle k will get a permission.

Thus because C is non-computable, some cycle must eventually succeed and $R_{\tau,i}$ is met.

Further, notice that q_A and q_B are always valid, regardless of if some $x_{k,j}$ gets no permissions, one permission, or two permissions. If $x_{k,j}$ gets no permissions or a j permission but not a k permission, then none of $x_{k,j}, x^1_{k,j}$ or $x^2_{k,j}$ enter A, B or Z, and q_A and q_B on each of these simply points at itself, i.e. $q_A(x_{k,j}) = x_{k,j}$, and so on. If $x_{k,j}$ gets a k permission but not a j permission, then only $x_{k,j} = q_A(x_{k,j})$ enters A, no other element enters and $q_A(x_{k,j})$ is retargeted to one of $x^1_{k,j}$ or $x^2_{k,j}$, because there are infinitely many τ expansionary stages and so at the next one, q_A recovers, and q_A, q_B on the rest of the followers remains correct. Finally if both a k and a j permission are received at stages s and $t > s$ respectively, then $q_A(x_{k,j})[s] = x_{k,j}$ and $q_A(x_{k,j})[t]$ enter A, and $x_{k,j} = q_B(x_{k,j})$ enters both Z and B maintaining that q_B is a valid Q-reduction, and finally one of $q_A(x^1_{k,j})$ or $q_A(x^2_{k,j})$ gets retargeted and thus maintains that q_A is a valid Q-reduction. Hence R_e is met. $\square$

Notice also that c_A and c_B are valid Q-reductions. At each stage in the construction, if any element $c_A(x)$ (or $c_B(x)$) enters C then either the corresponding element x enters A (or B) immediately, or $c_A(x)$ (respectively, $c_B(x)$) immediately retargets to an element not currently in C. Thus the reductions are always consistent, that is, valid Q-reductions, independently of (and without knowing) whether any individual $\ell(\tau, s)$ goes to infinity or gets stuck.

Finally, notice that q_A, q_B, c_A, c_B are clearly also sQ-reductions. For $q_A(x_{k,j}) \leqslant \max\{x_{k,j}, x^1_{k,j}, x^2_{k,j}\}$ and $q_B(x_{k,j}) = x_{k,j}$, and similarly for $x^1_{k,j}$ and $x^2_{k,j}$. And $c_A(x_{k,j}), c_B(x_{k,j}) \leqslant \max\{c, k, j\}$, similarly for $x^1_{k,j}$ and $x^2_{k,j}$. $\square$

3.3 sQ-degrees and wtt-degrees

In this section we look at the relationship between the sQ-degrees and the wtt-degrees. Despite the fact that the sQ-degrees are non-distributive, we can sometimes transfer results from the wtt-degrees to the Q-degrees through the sQ-degrees.

Lemma 52. *If* $\mathbf{a}$ *is a c.e. wtt-degree and* $A \in \mathbf{a}$ *is a c.e. semirecursive set, then* $W \leqslant_{wtt} A$ *implies that* $W \leqslant_{sQ} A$*, for any c.e. set* W*.*

Proof. Suppose A is semirecursive via $f(x, y)$, and that $\Gamma^A = W$ is the wtt-procedure with use function γ. Define the length of agreement $\ell(s) = \max\{x : \forall y \leqslant x,\ \Gamma^{A_s}(y) = W_s(y)\}$ and maximum length of agreement $m\ell(s) = \max\{\ell(t) : t < s\}$. We call a stage s *expansionary* if $\ell(s) > m\ell(s)$. Suppose that the wtt-procedure Γ has been accelerated so that every stage is expansionary. We define a Q-like function $m(x, s)$ for $x \leqslant \ell(s)$ as follows.

Suppose at stage s we are defining m on x for the first time and $\Gamma^{A_s}(x) = 0$ (say if x is below the length of agreement for the first time), or that $m(x, s-1) \in A_s$ but $\Gamma^{A_s}(x) = 0$. In order for $\Gamma^{A_s}(x)$ to change and thus for x to enter W, (at least) one element below the use $\gamma(x)$ must enter A. Compute $f(x, y)$ for $x, y \in \mathbb{N} \restriction \gamma(x) \setminus A_s$ and use this to determine the most preferred element z, which must enter A if any other element(s) below $\gamma(x)$ enter. This element exists because whenever $f(x, y) = x$, x is preferred over y (and conversely if $f(x, y) = y$), because if y enters A, x must also enter A. Now set $m(x, s) = z$.

If instead $m(x, s-1) \notin A_s$ is defined and $\Gamma^{A_s}(x) = 0$, set $m(x, s) = m(x, s-1)$.

Otherwise, $\Gamma^{A_s}(x) = 1$ (so $x \in W_s$). If this is the first time we are defining m on x, let $m(x, s) = x_0$ for some fixed $x_0 \in A$. If not, $m(x, s-1)$ is defined, so set $m(x, s) = m(x, s-1)$. Now either $m(x, s-1) \in A_s$ already, or some other small number entered A_s and caused $\Gamma^{A_s}(x)$ to change, but $m(x, s-1)$ hasn't entered A_s. However, because $m(x, s-1)$ is a preferred element, it must enter A at some stage t and we will have $\Gamma^A_t(x) = \Gamma^A_s(x) = 1$, since every stage is expansionary.

Then $m(x) = \lim_s m(x, s)$ exists and $m(x) \in A$ if and only if $x \in W$, and furthermore $m(x) \leqslant \max\{\gamma(x), x_0\}$. Thus we have an sQ reduction $W \leqslant_{sQ} A$. □

Theorem 53. *If* $A \leqslant_{wtt} B$ *are c.e., then there exists* $D \equiv_{wtt} A$ *with* $D \leqslant_{sQ} A, B$*. Hence if* $A \equiv_{wtt} B$ *then if the infimum of* A *and* B *in the Q-degrees or sQ-degrees exists, it will have the same wtt-degree as* A*.*

Proof. Suppose that $\Gamma^B = A$ is the wtt-procedure with use function γ, and this is accelerated so that every stage is expansionary, via $\ell(s)$.

Construction: For each i we define a set $P_{i,j} = \{x_{i,j} : 0 \leqslant j \leqslant \gamma(i)\}$. At stage $s+1$ we put $x_{i,j}$ into D if one of the following occurs.

1. Both $i \in A_{s+1} \backslash A_s$ and $j \in B_{s+1} \backslash B_s$. That is, i and j both enter between stages s and $s+1$, simultaneously changing A and the computation of Γ.

2. $i \in A_s$, $j \in B_{s+1} \setminus B_s$ and $\Gamma^{B_s}(i) \neq \Gamma^{B_{s+1}}(i) = 1$ for the first time. That is, i was already in A at stage s but $\Gamma^{B_s}(i) = 0$. Note that this means $i > \ell(s)$. Then $j \leqslant \gamma(i)$ entered B_{s+1} making the computation match. If there is no $i' < i$ causing a disagreement, now $i \leqslant \ell(s+1)$.

3. $i \in A_{s+1} \setminus A_s$ but $B_s \upharpoonright \gamma(i) = B_{s+1} \upharpoonright \gamma(i)$, and $\Gamma^{B_s}(i) = 1$. Further, $j \leqslant \gamma(i)$ is the last such j to have entered B. That is, i enters making the computations match, but B did not change below the use $\gamma(i)$ since the computation was already correct. Again we have $i > \ell(s)$ and if there is no $i' < i$ causing a disagreement, then $i \leqslant \ell(s+1)$.

End of Construction.

Then $D \leqslant_{sQ} A, B$. To see this, construct Q-like functions m and n, with m targeted for A and n for B.

Define $m(x_{i,j}, 0) = i$ and fix $a \notin A$. Unless otherwise stated, we keep $m(z, s+1) = m(z, s)$.

Suppose $i \in A_{s+1} \setminus A_s$. If $j \in B_{s+1} \setminus B_s$, define $m(x_{i,k}, s+1) = a$ for each $k \neq j$, and keep $m(x_{i,j}, s+1) = m(x_{i,j}, s) = i$. If instead $B_{s+1} \upharpoonright \gamma(i) = B_s \upharpoonright \gamma(i)$ and $\Gamma^{B_s}(i) = 1$, look at the enumeration of B up to stage $s+1$ and let j be the last $j \leqslant \gamma(i)$ to enter B during this time. Let $m(x_{i,k}, s+1) = a$ for each $k \neq j$, and keep $m(x_{i,j}, s+1) = m(x_{i,j}, s) = i$.

Now suppose $j \leqslant \gamma(i)$ enters B at stage $s+1$ and i is already in A_s. If $\Gamma^{B_{s+1}}(i) = 0$, define $m(x_{i,j}, s+1) = a$. If $\Gamma^{B_{s+1}}(i) = 1 \neq \Gamma^{B_s}$ and $m(x_{i,k}, s) = k$, define $m(x_{i,k}, s+1) = a$ for each $k \neq j$ and keep $m(x_{i,j}, s+1) = m(x_{i,j}, s) = i$.

It is not difficult to verify that $x_{i,j} \in D$ iff $\lim_s(m(x_{i,j}, s)) = m(x_{i,j}) \in A$. Further, $m(x_{i,j}) \leqslant \max\{i, a\}$ and so this is an sQ-reduction, thus $D \leqslant_{sQ} A$.

Now for our second reduction, define $n(x_{i,j}, 0) = j$ and fix $b \notin B$. Unless otherwise stated, we keep $n(z, s+1) = n(z, s)$.

Suppose $j \in B_{s+1} \setminus B_s$. If $i \in A_{s+1} \setminus A_s$, let $n(x_{i',j}, s+1) = b$ for all $i' \neq i$, and keep $n(x_{i,j}, s+1) = n(x_{i,j}, s) = j$. If $i \notin A_s$, define $n(x_{i,j}, s+1) = b$. Finally if $i \in A_s$ and $\Gamma^{B_{s+1}}(i) = 1 \neq \Gamma^{B_s}(i)$ and $n(x_{i,k}, s) = k$, let $n(x_{i,k}, s+1) = b$ for all $k \neq j$ and keep $n(x_{i,j}, s+1) = j$.

Then $x_{i,j} \in D$ iff $\lim_s(n(x_{i,j}, s)) = n(x_{i,j}) \in B$, and $n(x_{i,j}) \leqslant \max\{\gamma(i), b\}$. Thus we have an sQ-reduction $D \leqslant_{sQ} B$.

Finally, since $D \leqslant_{sQ} A$ we have $D \leqslant_{wtt} A$, so to see that $A \equiv_{wtt} D$ we need to show that $A \leqslant_{wtt} D$. To decide if $i \in A$, wait for a stage s where $i \leqslant \ell(s)$ and $D_s \restriction \max\{x : x \in P_{i,j}\} = D \restriction \max\{x : x \in P_{i,j}\}$. Then $i \in A$ iff $i \in A_s$, since every stage being expansionary implies that if i enters A_t at a later stage t, some $j \leqslant \gamma(i)$ must also enter B at the same stage. □

3.4 Simple sets and Q-degrees

Theorem 54. *There exists a c.e. set B such that B is not computable and no c.e. $V_e \leq_Q B$ is simple.*

Proof. We construct a c.e. set B to meet

$$P_e : \overline{B} \neq W_e$$

Additionally, we must meet the requirements

$$R_e : \text{ if } V_e \leqslant_Q B \text{ via } m_e \text{ then } V_e \text{ is not simple}$$

To meet the P_e requirements we use a standard Friedberg-Muchnik strategy. That is, we pick a follower x_e and wait for x_e to show up in W_e. If x_e never enters W_e, then $x_e \notin (W_e \cup B)$, hence $\overline{B} \neq W_e$ and P_e is satisfied. If x_e enters W_e at stage s then we put x_e into B and again P_e is satisfied.

For the R_e requirements, if $V_e \leqslant_Q B$ via m_e and V_e is not computable, then R_e will construct a c.e. set A_e that witnesses that V_e is not simple. That is, $|A_e| = \infty$ but $A_e \cap V_e = \emptyset$.

To do this, R_e uses the reduction m_e to put elements $x \notin V_e$ into A_e by restraining $m_e(x, s)$ out of B. Then x cannot enter V_e, provided x is below the *length of agreement function* at stage s:

$$\ell(e, s) = \max\{x : \forall y \leqslant x,\ y \in V_{e,s} \text{ iff } m_e(y, s) \in B_s\}$$

We also define the *maximum length of agreement function* to be:

$$m\ell(e, s) = \max\{\ell(e, t) : t < s\}$$

Define $m\ell(e, 0) = 0$. If the length of agreement gets stuck, that is $\lim_s \ell(e, s) < \infty$, then $V_e \not\leqslant_Q B$ and R_e is satisfied. Suppose $\ell(e, s)$ goes to infinity, so R_e wants to build an infinite set A_e where $A_e \cap V_e = \emptyset$. It is possible that $|\overline{V_e}| < \infty$, in which case R_e cannot build an infinite A_e, but is still satisfied since V_e is not simple. Another possibility is that m_e only points to a finite number of elements not in B. In this case $|\overline{V_e}|$ could be infinite, but V_e is computable and therefore not simple. Because of these two computable outcomes, as well as requiring the length of agreement to increase, we also require there to be new elements $m_e(x, s) \notin B$ for which $x \leqslant \ell(e, s)$ in order for s to be an expansionary stage. This will allow R_e to act and put an element into A_e at every expansionary stage. Define $q(e, s)$ to be the number of distinct elements $m_e(x, s)$ that are not in B_s, whose x is below the length of agreement. That is,

$$q(e, s) = |\{m_e(x, s) : x \leqslant \ell(e, s), m_e(x, s) \notin B_s\}|$$

In order for s to be an e-expansionary stage, we require $q(e, s)$ to have increased by at least 2 since the last e-expansionary stage, which will allow $P_{e'}$ requirements below R_e to pick a follower that is respected. For the first e-expansionary stage we require $q(e, s) \geqslant e + 2$.

The Priority Tree. We define the tree $\mathcal{T}$ inductively on $|\sigma|$. If $|\sigma|$ is even then σ has 3 children, $\sigma\hat{\ }\infty, \sigma\hat{\ }c$ and $\sigma\hat{\ }f$. If $|\sigma|$ is odd then σ has 2 children, $\sigma\hat{\ }0$ and $\sigma\hat{\ }1$. We assign R_e to σ iff $|\sigma| = 2e$, and assign P_e to σ iff $|\sigma| = 2e+1$. Write M_σ for the version of requirement M at guess σ and use lexicographic ordering with $\infty <_L c <_L f$ and $0 <_L 1$. Then M_σ has higher priority than M_τ if $\sigma <_L \tau$.

Consider some P_σ below the infinite outcome of R_τ, that is, $\tau\hat{\ }\infty \subseteq \sigma$. Let $2e = |\tau|$ and $2e' + 1 = |\sigma|$. When P_σ is assigned a follower, it is assigned an element that is not currently restrained by R_τ. At each subsequent τ-expansionary stage, R_τ sees at least 2 new elements, and only a single new follower may be appointed. Thus even if P_σ's follower has been seen by R_τ or is one of the elements that R_τ sees at some subsequent expansionary stage, R_τ always sees enough elements that it can pick a non-follower to restrain. In particular, P_σ's follower never gets restrained by R_τ. A similar thing happens when we have more requirements, say

R_{τ_1} and P_{σ_1} where $\tau\hat{\ }\infty \subseteq \sigma\hat{\ }1 \subseteq \tau_1\hat{\ }\infty \subseteq \sigma_1$. Each time R_τ and R_{τ_1} have an expansionary stage, they leave at least one unrestrained element that they have seen. Thus even if all the unrestrained but seen elements are assigned as followers, at the next expansionary stage both R_τ and R_{τ_1} will be able to restrain an element that is not a follower. This is true even if R_τ has restrained every element that R_{τ_1} has seen, or if all the elements seen by R_τ are restrained by R_{τ_1}.

Now suppose $\tau <_L \sigma$ but $\tau \not\subseteq \sigma$, so R_τ is on a branch further to the left than P_σ. When P_σ is appointed a follower at stage s, R_τ has restrained some number of elements, and P_σ's follower is not allowed to be any of these elements. If R_τ is visited again before P_σ has acted, then P_σ is initialised. If P_σ acts and puts its follower in B, R_τ could not have been visited between when P_σ's follower was appointed and when the follower is enumerated, and so R_τ could not have later restrained that follower. If R_τ is visited again, this follower will not be one of the elements it considers (since it is now in B), and thus P_σ does not cause any problems for R_τ. Also, if $\sigma <_L \tau$ and P_σ has a follower, R_τ can restrain this follower, since if the follower is enumerated then R_τ is initialised.

Definition 55. We define the notions of σ-stage and σ-expansionary by induction on $|\sigma|$.

(i) Every stage s is a λ-stage.

(ii) Suppose s is a σ-stage with $|\sigma| = 2e$. Then say s is σ-expansionary if $\ell(e, s) > m\ell(e, s)$ and $q(e, s) \geqslant n(e, s)$ (we will define $n(e, s)$ during the construction) and declare s to be a $\sigma\widehat{\ }\infty$-stage.

If $\ell(e, s) > m\ell(e, s)$ but $q(e, s) < n(e, s)$, declare s to be a $\sigma\widehat{\ }c$-stage.

If $\ell(e, s) \leqslant m\ell(e, s)$ then declare s to be a $\sigma\widehat{\ }f$-stage.

(iii) Suppose s is a σ-stage with $|\sigma| = 2e + 1$. If no P_τ where $|\tau| = |\sigma|$ has acted yet and put a follower into B (including P_σ itself), then declare s to be a $\sigma\widehat{\ }1$-stage. If some P_τ, $|\tau| = |\sigma|$ has already acted, or P_σ acts at stage s then declare s to be a $\sigma\widehat{\ }0$ stage.

Define TP_s to be the unique σ of length s with s a σ-stage.

We say that P_σ requires attention at stage s if $W_{e,s} \cap B = \emptyset$, s is a σ-stage and either:

(i) Currently P_σ has no follower; or

(ii) P_σ has a follower $x \in W_{e,s}$.

Construction. Define $n(e, 0) = e + 2$ for every e.

At stage s, compute TP_s. Initialise all M_τ where $\tau \not\leqslant_L TP_s$. Run through substages t for $t \leqslant s$ as follows.

- If $t = 2e$ and s is σ-expansionary for $\sigma \subseteq TP_s$, $|\sigma| = t$, act for R_σ as follows. Consider all $m_e(x, s) \notin B_s$ where $x \leqslant \ell(e, s)$ which R_σ has not yet restrained (at previous expansionary stages). If one of these $m_e(x, s)$ is already restrained by some R_τ, restrain it. Otherwise, restrain an $m_e(x, s)$ that is not a follower for any P_σ where $\sigma \not\leqslant_L \tau$.

Claim 1. Such an $m_e(x, s)$ exists.

Enumerate the corresponding x into $A_{\sigma,s+1}$ and define $n(e, s+1) = n(e, s) + 2$.

- If $t = 2e + 1$, consider P_σ where $\sigma \subseteq TP_s$ and $|\sigma| = t$. If P_σ requires attention and does not have a follower, then appoint the smallest unrestrained x that is not currently a follower for any $P_{\sigma'}$ as the follower for P_σ. Define $n(e, s+1) = n(e, s)$ for all e where $n(e, s+1)$ is not already defined and end stage s.

  Claim 2. The element x following P_σ is never restrained by any higher priority R_τ.

  If P_σ requires attention and already has a follower, enumerate the follower into B_{s+1}, initialise all requirements M_τ for $\tau \not\leq_L \sigma$, define $n(e, s+1) = n(e, s)$ for all e where $n(e, s+1)$ is not yet defined, and end stage s.

End of Construction.
Verification. Let TP be the leftmost path visited infinitely often.

Lemma 56. *Truth of outcome lemma: all R_e and P_e requirements are met.*

Proof. Suppose $\xi \subseteq TP$. We will prove the following by simultaneous induction on $|\xi|$:

(i) If $|\xi| = 2e$ and $\xi\widehat{\ }\infty \subseteq TP$ then

  (a) $|A_\xi| = \infty$ and $A_\xi \cap V_e = \emptyset$ and thus R_e is satisfied,

  (b) If s is the d^{th} ξ-expansionary stage, there is at least one element $m_e(x, s) \notin B_s$ which is not a follower for any P_σ, $\sigma \not\leq_L \xi$ and R_ξ can restrain this element. That is, Claim 1 holds.

(ii) If $|\xi| = 2e$ and $\xi\widehat{\ }c \subseteq TP$ then V_e is computable.

(iii) If $|\xi| = 2e$ and $\xi\widehat{\ }f \subseteq TP$ then $V_e \not\leq_Q B$.

(iv) If $|\xi| = 2e + 1$ and $\xi^\frown 1 \subseteq TP$ then P_e is satisfied.

(v) If $|\xi| = 2e + 1$ and $\xi^\frown 0 \subseteq TP$ then some P_σ with $|\sigma| = |\xi|$ (potentially $\sigma = \xi$) enumerates its follower into B, satisfying P_e. In particular, P_ξ's follower and the followers of P_σ, $|\sigma| = |\xi|$ are not restrained by any R_τ, $\tau \leqslant_L \xi$, and thus Claim 2 holds.

Notice that once some P_σ (where $|\sigma| = 2e + 1$) acts and puts a follower in B, then P_e is satisfied and no other version of P_e ever requires attention again.

Base case: $\xi = \lambda$. Here we are considering R_0.

Suppose $f \subset TP$. Since TP is the leftmost path visited infinitely often, this means that after some stage s_0, $\ell(0, s) < m\ell(0, s_0)$ for all $s > s_0$. This can only happen if one of the following occurs. There is a disagreement between V_0 and B (so $z = \ell(0, s_0) + 1$ is in V_0 but $m_0(z) \notin B$, or $z \notin V_0$ but $m_0(z) \in B$) in which case $V_0 \nleqslant_Q B$. Or, $m_0(z)$ is undefined, so m_0 is partial, and thus again $V_0 \nleqslant_Q B$. In both cases R_0 is satisfied.

Now suppose $c \subset TP$. In this case there are infinitely many stages s for which $\ell(0, s) > m\ell(0, s)$, so $V_0 \leqslant_Q B$, however the infinite outcome is only visited finitely many times so after some stage s_0, m_0 does not point at any new elements that are outside of B. That is, for some n, if $m_0(x) \notin B$ then $m_0(x) < n$. In this case, V_0 is computable: go to a stage s_0 after which the infinite outcome is not visited and $B_{s_0} \upharpoonright n = B \upharpoonright n$. To compute if x is in V_0, go to a stage $s > s_0$ where $x < \ell(0, s)$ and compute $m_0(x, s)$. If $m_0(x, s) < n$ and $m_0(x, s) \notin B_s$ then $x \notin V_0$, otherwise $x \in V_0$ (it is possible $m_0(x, s) > n$ and $m_0(x, s) \notin B_s$ but then $m_0(x, s)$ later goes into B). This covers both the computable cases, when m_0 has a finite range and when m_0 has an infinite range but only finitely many things in the range are not in B. Since computable sets are not simple, R_0 is again satisfied.

Finally suppose $\infty \subset TP$, so there are infinitely many 0-expansionary stages. At each expansionary stage, m_0 must be pointing at 2 more elements outside of B than what it did at the previous expansionary stage. That is, at the d^{th} expansionary stage s, there are at least $(2d)$-many distinct elements $x \leq \ell(0,s)$ for which $m_0(x,s) \notin B_s$. Of these, $d-1$ have been restrained, one at each of the previous expansionary stages, and at most $d-1$ followers have been appointed below ∞. Thus at stage s, R_λ has at least 2 non-follower elements that it can choose from to restrain (the unrestrained one may then potentially be assigned as a follower before the stage ends). Since the elements that R_λ restrains are never later assigned as followers, the restraints are respected. The elements x corresponding to restrained elements $m_0(x,s)$ are enumerated into A_0, and the length of agreement ensures the x never enter V_0. Thus $|A_\lambda| = \infty$ and $A_\lambda \cap V_0 = \emptyset$, witnessing that V_0 is not simple. Hence R_0 is satisfied.

Inductive step: Suppose $|\xi| > 0$ and (i) through (v) hold for all $\sigma \subset \xi$.

Case 1: $|\xi| = 2e+1$. Go to the least ξ-stage s_0 where for all $\tau <_L \xi$ and $s > s_0$, if $\tau \not\subset \xi$, then s is not a τ-stage, and if $\tau \subset \xi$ then P_τ does not require attention at stage s. That is, we do not visit any path to the left of ξ after stage s_0, and any P_τ requirement above P_ξ has either already acted and put a follower into B, or if it is to receive a follower (that never goes in) then it has already received said follower.

Suppose $\xi^\frown 1 \subset TP$. If P_ξ does not require attention for any $t > s_0$, then it must be that $W_e \cap B \neq \emptyset$ and thus P_e is already satisfied. Otherwise, go to the first stage $s \geqslant s_0$ for which P_ξ requires attention. At this stage P_ξ has priority and receives a follower x. Then P_ξ never again requires attention and so x is never enumerated into B (if P_ξ did require attention at some stage $t > s$ then it would have highest priority at stage t and enumerate x into B, contradicting that $\xi^\frown 1 \subset TP$). This means that the element x following P_ξ never enters W_e, and thus P_e is satisfied.

Now suppose $\xi^\frown 0 \subset TP$. If s_0 (as chosen above) is a $\xi^\frown 0$-stage, then some P_σ with $|\sigma| = |\xi|$ must have already acted and put a follower in, satisfying P_e. Otherwise, P_ξ requires attention and receives a follower x at stage s_0. At this stage, x is not restrained by any R_τ, $\tau <_L \xi$. By induction, each R_τ above P_ξ always has a non-follower element to restrain at each subsequent expansionary stage, and so P_ξ's follower is never restrained by a higher priority requirement. Now there are two possibilities. First, some P_σ with $|\sigma| = |\xi|$ further to the right receives a follower and enumerates this follower in between ξ-stages. In this case P_ξ never requires attention again because P_e is satisfied. Otherwise, at some stage $s > s_0$, x enters $W_{e,s}$. At this stage, P_ξ requires attention and has highest priority, and so will enumerate x into B, satisfying P_e.

Case 2: $|\xi| = 2e$**.** Go to the least ξ-stage s_0 where for all $\tau <_L \xi$ and $s > s_0$, if $\tau \not\subset \xi$, then s is not a τ-stage. So we do not visit any path to the left of ξ after stage s_0. Note that any P_σ requirements above ξ that will ever enumerate a follower into B must have already done so by stage s_0, and as such R_ξ is never initialised again. Now the argument is similar to the base case. For the infinity outcome, R_ξ is allowed to restrain followers of higher priority P_τ as these will never be enumerated after stage s_0. At each ξ-expansionary stage, R_ξ and all other R_τ where $\tau\hat{}\infty \subset \xi$ see at least 2 new elements each, while only a single new follower may be appointed below R_ξ (after R_ξ has chosen which element it restrains). Then no matter how the elements seen by the R_τ's and R_ξ overlap, R_ξ will have a non-follower element it can restrain. Once restrained, this element will never be appointed as a follower and so the restraint is respected. As such R_ξ succeeds in building an A_ξ which witnesses that V_e is not simple and so R_e is satisfied. $\square$

This shows that all the R_e and P_e requirements are met. $\square$

3.5 Minimal pairs and half minimal pairs in the Q-degrees

Downey and Stob showed in 1984 that there is a minimal pair of wtt-degrees within the same Turing degree [20]. A similar argument gives the same result but with the sQ- and Q-degrees. For this proof we use a pinball machine, which is more well-suited to this argument than a tree of strategies because of how we 'trace' elements to ensure the minimal pair has the same Q-degree.

Theorem 57. *There is a minimal pair of sQ-degrees within the same Q-degree.*

Proof. We construct A_1 and A_2 on a pinball machine so that $A_1 \equiv_Q A_2$ and they satisfy the requirements

$$R_e: \text{ if } V_e \leqslant_{sQ} A_1, A_2 \text{ then } V_e \text{ is computable}$$

To make A_1 and A_2 non-computable we further satisfy

$$P_e : \overline{A_1} \neq W_e$$

Note that it suffices to make only one of A_1, A_2 non-computable since they have the same degree. We will have Q-like functions γ and δ that witness $A_1 \leqslant_Q A_2$ and $A_2 \leqslant_Q A_1$, respectively. We use a standard Friedberg strategy to meet the P_e requirements.

As in figure 3.1, the pinball machine has holes H_e for P_e requirements, and gates G_e for R_e requirements. Next to each gate G_e there is a corral C_e. These are arranged from bottom to top in order of priority, $G_0, H_0, G_1, \ldots$. Below the first gate G_0 are the enumeration pockets for A_1 and A_2.

Basic strategy. Pick a_1 targeted for A_1 to be a follower for P_e. At every stage s, we define γ and δ on all existing elements. So if a_1 does not enter A_1 at the next stage, we give it a trace $\gamma(a_1)[s]$ which is a fresh big number. Now for a_1 to go into A_1, we need $\gamma(a_1)[s]$ to go into A_2. So

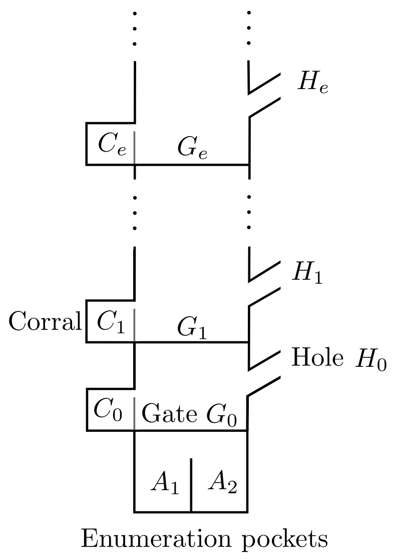


Figure 3.1: Sketch of the pinball machine

at the next stage, either $\gamma(a_1)$ goes in or we give it a trace $\delta(\gamma(a_1)[s])[s+1]$ which is a fresh number. We call these elements the traces of a_1, and denote $\gamma(a_1)$ by $t(a_1)$, $\delta(\gamma(a_1))$ by $t^2(a_1)$, $\gamma(\delta(\gamma(a_1)))$ by $t^3(a_1)$ and so on. At some later stage, suppose a_1 gets realised. That is, it enters W_e, so now P_e wants to put a_1 into A_1. At this stage, a_1 has a trace entourage with n elements, $a_1, t(a_1), \ldots, t^n(a_1)$. All of the traces (given by γ and δ) need to be enumerated into their respective sets (A_1 or A_2) in order for a_1 to be enumerated into A_1. Initialise all lower priority balls already on the machine and drop a_1 along with its trace entourage down the hole H_e and to the first unoccupied gate G_j. Put $a_1, \ldots, t^{n-1}(a_1)$ into the corral C_j and place the last element of the trace, $t^n(a_1)$, at the gate G_j. While $t^n(a_1)$ is at the gate G_j, we give it a trace at each stage as we did for a_1. We say $T^n(a_1)$ is the *primary ball* to which the new traces are associated with. The gate will open at the next j-expansionary stage, letting $t^n(a_1)$ along with its traces drop to the next unoccupied gate, initialising all lower priority balls. As before, we put $t^n(a_1)$ along with all but the last trace into the new gate's corral, and put the last trace at the gate (this last trace is now a primary ball too). If there is no unoccupied gate below, then $t^n(a_1)$ and its traces all get enumerated into their target sets, A_1 or A_2. Once $t^n(a_1)$ and all of its traces have gone in to their target sets, we take the next element $t^{n-1}(a_1)$ from the corral C_j to the gate G_j. While there is no element at the gate G_j we consider G_j to be unoccupied (even if there are elements in the corral). We do not take $t^{n-1}(a_1)$ out of the corral before $t^n(a_1)$ has gone in, since it could be that $t^n(a_1)$ gets stuck at a gate forever and never goes in. In that case we can't enumerate $t^{n-1}(a_1)$ in either, but had we moved it out to the gate it is possible it would have been allowed to go in, making either δ or γ wrong. When P_e releases its follower a_1, it is assigned another follower a_1' in case a_1 never gets enumerated. Should this follower be realised while a_1 is still in the machine, it and its trace entourage have a lower priority to a_1 and its traces. Once P_e gets a follower enumerated, it stops receiving new followers.

Definition 58. We say that a ball x *requires attention* at stage s if one of the following holds.

(i) x is a follower for a P_e requirement and has been realised (so $W_{e,s} \cap A_{1,s} = \emptyset$ and $x \in W_{e,s}$)

(ii) x is a primary ball at a gate G_j and s is a j-expansionary stage.

Construction. At stage s, find the highest priority ball x which requires attention. Cancel all lower priority balls. Drop x along with its trace entourage to the first unoccupied gate G_j below it. Place x and all but the last trace $t^n(x)$ in the corral C_j, placing $t^n(x)$ on the gate G_j. Declare $t^n(x)$ to be a primary ball. If there are no unoccupied gates, enumerate x and all of its traces into their target sets. Appoint a fresh large number to the highest priority P_e requirement that does not have a follower and has not yet been satisfied. Appoint a fresh large number as the next trace to each follower waiting at a hole and to each primary ball waiting at a gate (that is, extend the definitions of γ and δ).

Verification.

Lemma 59. *Each gate has finitely many primary balls that are permanent residents.*

Proof. Each gate receives at most one primary ball at a time. When a primary ball is at a gate, that gate is occupied and no new balls stop at this gate. The primary ball receives traces, but these are themselves not primary balls, they are only associated to the primary ball. Thus the gate can have at most one primary ball that is a permanent resident. □

Lemma 60. *Each P_e is met.*

Proof. Suppose not. Let e be smallest such that P_e is not met. Let s_0 be the least stage where all higher priority P_j requirements have been met. At stage s_0 (if not sooner), P_e receives a follower x_0. Then at some stage $s_1 > s_0$, this follower must get realised. For if it doesn't, P_e is met since $x_0 \in \overline{A_1}$ but not in W_e. Since all higher priority P_j have already been met, x_0 must be the highest priority ball that requires attention at stage s_1. Thus x_0 along with its trace entourage are dropped to the first unoccupied gate, and all lower priority balls are cancelled. Now P_e gets assigned a new follower x_1 since it is the highest priority requirement that has not yet been satisfied and does not have a follower at its hole. Meanwhile, since we assume that P_e does not get met, x_0 cannot ever be enumerated into A_1. So x_0 or one of its traces must get stuck forever at some gate G_j, $j < e$, and so is a permanent resident of G_j. By the same argument as for x_0, at some stage $s_2 > s_1$ we must have that x_1 is realised, and so x_1 and its trace entourage are dropped to the next unoccupied gate. This process can repeat at most $e + 1$ many times before every gate below H_e has a permanent resident. Now P_e is assigned yet another follower x, and by the same argument this follower must get realised at some later stage t. Now all the gates below H_e are occupied, and so x and its trace entourage are dropped all the way down and enumerated into their target sets. But then P_e is satisfied. Thus every P_e requirement must be met. □

Lemma 61. *For each e, there are infinitely many clear stages. That is, stages for which the only residents at gates G_j, $j \leqslant e$, are permanent residents.*

Proof. Go to a stage s_0 at which all permanent residents below G_e are already at their gates. For any stage $t > s_0$, let x be the ball of highest priority that is not a permanent resident at its final gate. When x moves at stage $s > t$, it will initialise all lower priority balls, in particular any ball at gates G_j for $j \leqslant e$ which are not permanent residents. If x is above gate G_e, then s is a clear stage for gate G_e. Otherwise, x must get enumerated and the stage s' in which this happens is a clear stage. □

Lemma 62. *Each R_e is met.*

Proof. If $V_e \not\leqslant_{sQ} A_1, A_2$ then R_e is met. In this case the length of agreement gets stuck at some stage s, after which the gate G_j never opens.

Suppose $V_e \leqslant_{sQ} A_1, A_2$, then G_e has infinitely many expansionary stages. Go to a stage s_0 at which all P_j requirements below G_e have finished acting. Since all P requirements are met, such a stage exists. To compute $V_e \restriction z$, go to a clear stage $s > s_0$ for which $\ell(e, s) > z$. Then $V_{e,s} \restriction z = V_e \restriction z$. To see this, consider how $V_{e,s} \restriction z$ could change after stage s. The only way this can happen is if some balls p, q get enumerated into each of A_1 and A_2 between e-expansionary stages. To affect the computation of $V_{e,s} \restriction z$, p and q must be below the use of this computation. Since the P_j requirements with $j < e$ have finished, p and q must have originated from a lower priority requirement above G_e. Any number assigned as a follower or a trace after stage s will be a fresh big number that cannot affect the computation, and so p and q must have already existed on the machine before stage s. Suppose p has higher priority than q, so q must enter before p can move, otherwise p will initialise q. Now, q must stop at gate G_e. For if a ball were to occupy gate G_e after the clear stage s, this ball must have lower priority to p and q else it would have initialised them when it moved to gate G_e. But then this ball is initialised when q moves, so q must stop at G_e. Then q moves at the next e-expansionary stage, and q (and its traces) must enter before p can move. This means that when p moves, it must also stop at gate G_e. But then p cannot enter before another e-expansionary stage occurs, and thus the computation of $V_{e,s} \upharpoonright z$ is preserved. Hence V_e is computable and R_e is met. □

□

Definition 63. A set A is *R-cappable* if there exists a non-computable set B such that $\deg_R(A) \cap \deg_R(B) = \mathbf{0}$. That is, A is half of a minimal pair in the R degrees, where R is a reducibility.

Definition 64. A set A is *promptly simple* if there is a computable function g such that $\forall e(|W_e| = \infty \to \exists^\infty x, s(x \in W_{e,s}$ and $x \in A_{g(s)}))$.

Notice that in this definition it suffices to have a single x where $x \in W_{e,s}$ and $x \in A_{g(s)}$, and this immediately gives us infinitely many such x in the intersection. To see this, consider $W_{e'} = W_e \setminus x$. Then there are z, s such that $z \in W_{e',s} \cap A_{g(s)}$ and z is also in $W_{e,s}$, and this can be iterated to obtain infinitely many such z.

Theorem 65. *If A is a c.e. set that is half of a minimal pair in the Q-degrees then A is half of a minimal pair in the Turing degrees.*

Proof. Fix A that is not half of a Turing minimal pair. By a Theorem of Ambos-Spies et. al, we know that A has promptly simple degree. We describe the idea in case A is promptly simple. We need to show that A is not half of a minimal pair in the Q-degrees.

Let A be promptly simple via g and let B be a given non-computable c.e. set. We build a set $Z \leqslant_{sQ} A, B$ with sQ-reductions m and n respectively, to meet requirements

$$R_e : Z \neq \overline{W_e}$$

Basic Strategy: Pick followers $z_0, \ldots, z_n \notin Z_s$ and set $m(z_i, s) = n(z_i, s) = i$. Once z_n has been realised (has entered W_e), we define a new follower z_{n+1}, so long as R_e is not yet met. To meet R_e we need to put a single z_i into Z. In order to keep Z sQ-below A and B, we can only put z_i in if both $m(z_i)$ and $n(z_i)$ enter A and B respectively. Here we use that A is promptly simple. Since B is an infinite c.e. set, for infinitely many x, s we will have $x \in B_s$ and $x \in A_{g(s)}$. Since B is non-computable, at some stage s we must have some small $i < n$ enter B_s, where at stage s we have followers $z_0, \ldots, z_n$. Compute $g(s)$ and check if $i \in A_{g(s)}$. If it is, which infinitely often it will be, then we can enumerate z_i and meet R_e.

Construction: Fix some $a \notin A$ and $b \notin B$. At stage s, run through substages e for $e \leqslant s$ as follows. If R_e is already met, $z_{e,i}$ is defined, then if $i \in B_{s+1} \setminus B_s$, set $n(z_{e,i}, s+1) = b$, and if $i \in A_{s+1} \setminus A_s$, set $m(z_{e,i}, s+1) = a$. Otherwise if R_e is not yet met:

- If R_e does not have a follower, assign a new follower $z_{e,0}$ to R_e.
- If R_e has followers $z_{e,0}, \ldots, z_{e,n}$ that have all been realised, assign a new follower $z_{e,n+1}$ to R_e.
- For new followers $z_{e,i}$, define $m(z_{e,i}, s) = n(z_{e,i}, s) = i$. Unless otherwise stated, keep $m(z_{e,i}, s+1) = m(z_{e,i}, s)$ and $n(z_{e,i}, s+1) = n(z_{e,i}, s)$.
- Suppose $i \in B_{s+1} \setminus B_s$. Compute $g(s+1)$, and check if $i \in A_{g(s+1)}$ or $i \in A_{s+1}$. If not, then set $n(z_{e,i}, s+1) = b$. Otherwise, enumerate $z_{e,i}$ into Z, meeting R_e. Keep $m(z_{e,i}, s+1) = n(z_{e,i}, s+1) = i$, and for any other $j \leqslant n$ for which $j \in A_{s+1} \setminus B_{s+1}$, define $m(z_{e,j}, s+1) = a$.
- Suppose $i \in A_{s+1} \setminus A_s$ but $n(z_{e,i}, s) = b$ (for instance if i entered B at an earlier stage t, but $s > g(t)$ so that $z_{e,i}$ did not get permission at stage t). Then define $m(z_{e,i}, s+1) = a$.

Verification:

Lemma 66. *Every R_e is met.*

Proof. Suppose R_e is not met. Then every follower $z_{e,i}$ that is assigned to R_e is eventually realised (enters W_e), because if $z_{e,i}$ is not realised then it is not in W_e and not in Z, and thus R_e is satisfied. So $z_{e,i}$ gets realised and R_e gets assigned yet another follower $z_{e,i+1}$, however none of these followers are ever enumerated, so R_e is assigned a follower $z_{e,i} \notin Z$ for every $i \in \mathbb{N}$. For $z_{e,i}$ to remain outside of Z, either i does not enter one or both of A and B, or i enters B at some stage s but only enters A after stage $\max\{s, g(s)\}$. However, each $z_{e,i}$ is targeted at i by m and n, which means that for every i, either i does not enter both A and B, or if it does enter both then it enters A after stage $\max\{s, g(s)\}$, where s is the stage when i entered B. This contradicts that A is promptly simple, and thus R_e must be met. □

Further, m and n are valid Q-reductions: If $z_{e,i}$ has been enumerated into Z then i is in both A and B, and in this case $m(z_{e,i}) = n(z_{e,i}) = i$. If $z_{e,i}$ is never enumerated, then either $m(z_{e,i}) = i \notin A$, or m retargets to $m(z_{e,i}) = a \notin A$ depending on when and if i enters A, and similarly for n. Finally, $m(z_{e,i}) \leqslant \max\{i, a\}$ and $n(z_{e,i}) \leqslant \max\{i, b\}$ and so these are sQ-reductions. □

Bibliography

[1] Arslanov, M. M., Batyrshin, I. I., and Omanadze, R. S. Structural properties of Q-degrees of n-c. e. sets. *Ann. Pure Appl. Logic 156*, 1 (2008), 13–20.

[2] Arslanov, M. M., and Omanadze, R. S. Q-degrees of n-c.e. sets. *Illinois J. Math. 51*, 4 (2007), 1189–1206.

[3] Ash, C., and Knight, J. *Computable structures and the hyperarithmetical hierarchy*, vol. 144 of *Studies in Logic and the Foundations of Mathematics.* North-Holland Publishing Co., Amsterdam, 2000.

[4] Batyrshin, I. I. Non-isolated quasi-degrees. *MLQ Math. Log. Q. 55*, 6 (2009), 587–597.

[5] Batyrshin, I. I. Q-reducibility and m-reducibility on computably enumerable sets. *Sibirsk. Mat. Zh. 55*, 6 (2014), 1221–1239.

[6] Bazhenov, N., Harrison-Trainor, M., and Melnikov, A. Computable Stone spaces. *Ann. Pure Appl. Logic 174*, 9 (2023), Paper No. 103304, 25.

[7] Bazhenov, N., Melnikov, A. G., and Ng, K. M. Every Δ^0_2 polish space is computable topological. *Proceedings of the American Mathematical Society* (2024).

[8] Belegradek, O. V. Algebraically closed groups. *Algebra i Logika 13* (1974), 239–255, 363.

[9] Boone, W. The word problem. *Annals of Math 70* (1959), 207–265.

[10] Ceĭtin, G. S. Algorithmic operators in constructive complete separable metric spaces. *Dokl. Akad. Nauk SSSR 128* (1959), 49–52.

[11] Chitaia, I. Hyperhypersimple sets and Q_1-reducibility. *MLQ Math. Log. Q. 62*, 6 (2016), 590–595.

[12] Chitaia, I., Ng, K. M., Sorbi, A., and Yang, Y. Minimal degrees and downwards density in some strong positive reducibilities and quasi-reducibilities. *J. Logic Comput. 33*, 5 (2023), 1060–1088.

[13] Chitaia, I., Omanadze, R., and Sorbi, A. Notes on conjunctive and quasi degrees. *J. Logic Comput. 31*, 5 (2021), 1317–1329.

[14] Cholak, P., Downey, R., and Stob, M. Automorphisms of the lattice of recursively enumerable sets: promptly simple sets. *Trans. Amer. Math. Soc. 332*, 2 (1992), 555–570.

[15] Church, A. An Unsolvable Problem of Elementary Number Theory. *Amer. J. Math. 58*, 2 (1936), 345–363.

[16] Downey, R., and Jockusch, C. G. Every low Boolean algebra is isomorphic to a recursive one. *Proc. Amer. Math. Soc. 122*, 3 (1994), 871–880.

[17] Downey, R., LaForte, G., and Nies, A. Computably enumerable sets and quasi-reducibility. *Ann. Pure Appl. Logic 95*, 1-3 (1998), 1–35.

[18] Downey, R., and Melnikov, A. Computable structure theory: A unified approach. Unpublished book.

[19] Downey, R., and Melnikov, A. Computably compact metric spaces. *preprint* (2023).

[20] Downey, R. G., and Stob, M. Structural interactions of the recursively enumerable T- and w-degrees. *Ann. Pure Appl. Logic 31*, 2-3 (1986), 205–236. Special issue: second Southeast Asian logic conference (Bangkok, 1984).

[21] Ershov, Y., and Goncharov, S. *Constructive models.* Siberian School of Algebra and Logic. Consultants Bureau, New York, 2000.

[22] Eršov, J. L. Positive equivalences. *Algebra i Logika 10* (1971), 620–650.

[23] Feiner, L. *Orderings and Boolean algebras not isomorphic to recursive ones.* ProQuest LLC, Ann Arbor, MI, 1968. Thesis (Ph.D.)–Massachusetts Institute of Technology.

[24] Feiner, L. Hierarchies of Boolean algebras. *J. Symbolic Logic 35* (1970), 365–374.

[25] Fischer, P. Pairs without infimum in the recursively enumerable weak truth table degrees. *The Journal of symbolic logic 51*, 1 (1986), 117–129.

[26] Fischer, P., and Ambos-Spies, K. Q-degrees of re sets. In *JOURNAL OF SYMBOLIC LOGIC* (1987), vol. 52, ASSN SYMBOLIC LOGIC INC 1325 SOUTH OAK ST, CHAMPAIGN, IL 61820, pp. 317–317.

[27] Fowler, D., and Robson, E. Square root approximations in old babylonian mathematics: Ybc 7289 in context. *Historia Mathematica 25*, 4 (1998), 366–378.

[28] Friedberg, R. M. Two recursively enumerable sets of incomparable degrees of unsolvability (solution of Post's problem, 1944). *Proc. Nat. Acad. Sci. U.S.A. 43* (1957), 236–238.

[29] Friedberg, R. M. Three theorems on recursive enumeration. I. Decomposition. II. Maximal set. III. Enumeration without duplication. *J. Symbolic Logic 23* (1958), 309–316.

[30] Garey, M. R., and Johnson, D. S. *Computers and intractability.* A Series of Books in the Mathematical Sciences. W. H. Freeman and Co., San Francisco, CA, 1979. A guide to the theory of NP-completeness.

[31] Gödel, K. über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatsh. Math. Phys. 38*, 1 (1931), 173–198.

[32] Goncharov, S. S., and Knight, J. F. Computable structure and non-structure theorems. *Algebra and Logic 41*, 6 (2002), 351–373.

[33] Grubba, T., Schroder, M., and Weihrauch, K. Computable metrization. *Mathematical Logic Quarterly 53* (2007), 381–395.

[34] Harrington, L., and Soare, R. I. Post's program and incomplete recursively enumerable sets. *Proc. Nat. Acad. Sci. U.S.A. 88*, 22 (1991), 10242–10246.

[35] Harrison-Trainor, M., and Melnikov, A. An arithmetic analysis of closed surfaces. *Preprint* (2021).

[36] Harrison-Trainor, M., Melnikov, A. G., and Ng, K. M. Computability of Polish spaces up to homeomorphism. *Journal of Symbolic Logic 85* (2020), 1664–1686.

[37] Hilbert, D. Mathematical problems (1900). In *Ideas that created the future—classic papers of computer science.* MIT Press, Cambridge, MA, [2021], pp. 45–50. Reprinted from [1557926].

[38] Hisamiev, N. G. Criterion for constructivizability of a direct sum of cyclic p-groups. *Izv. Akad. Nauk Kazakh. SSR Ser. Fiz.-Mat.*, 1 (1981), 51–55, 86.

[39] Kach, A. M., and Turetsky, D. Limitwise monotonic functions, sets, and degrees on computable domains. *J. Symbolic Logic 75*, 1 (2010), 131–154.

[40] Kalimullin, I., Khoussainov, B., and Melnikov, A. Limitwise monotonic sequences and degree spectra of structures. *Proc. Amer. Math. Soc. 141*, 9 (2013), 3275–3289.

[41] Khisamiev, N. Constructive abelian groups. In *Handbook of recursive mathematics, Vol. 2*, vol. 139 of *Stud. Logic Found. Math.* North-Holland, Amsterdam, 1998, pp. 1177–1231.

[42] Khisamiev, N. G. Hierarchies of torsion-free abelian groups. *Algebra i Logika 25*, 2 (1986), 205–226, 244.

[43] Khoussainov, B., Nies, A., and Shore, R. Computable models of theories with few models. *Notre Dame J. Formal Logic 38*, 2 (1997), 165–178.

[44] Kleene, S. C. *Introduction to Metamathematics.* P. Noordhoff N.V., Groningen, 1952.

[45] Koh, H. T., Melnikov, A., and Ng, K. M. Computable topological groups, 2022.

[46] Koh, H. T., Melnikov, A. G., and Ng, K. M. Counterexamples in effective topology. Submitted.

[47] Kreisel, G., Lacombe, D., and Shoenfield, J. R. Partial recursive functionals and effective operations. In *Constructivity in mathematics: Proceedings of the colloquium held at Amsterdam, 1957 (edited by A. Heyting)* (1959), Stud. Logic Found. Math., North-Holland, Amsterdam, pp. 290–297.

[48] Lachlan, A. H. On the lattice of recursively enumerable sets. *Trans. Amer. Math. Soc. 130* (1968), 1–37.

[49] Lupini, M., Melnikov, A., and Nies, A. Computable topological abelian groups. *J. Algebra 615* (2023), 278–327.

[50] Macintyre, A. Omitting quantifier-free types in generic structures. *J. Symbolic Logic 37* (1972), 512–520.

[51] Mal′cev, A. Constructive algebras. I. *Uspehi Mat. Nauk 16*, 3 (99) (1961), 3–60.

[52] Mal′cev, A. I. On recursive Abelian groups. *Dokl. Akad. Nauk SSSR 146* (1962), 1009–1012.

[53] Marčenkov, S. S. A certain class of incomplete sets. *Mat. Zametki 20*, 4 (1976), 473–478.

[54] Melnikov, A., and Ng, K. M. Separating notions in computable topology. Submitted, 2023.

[55] Melnikov, A. G. Computable abelian groups. *Bull. Symb. Log. 20*, 3 (2014), 315–356.

[56] Metakides, G., and Nerode, A. Effective content of field theory. *Annals of Mathematical Logic 17*, 3 (1979), 289–320.

[57] Moschovakis, Y. N. Recursive metric spaces. *Fund. Math. 55* (1964), 215–238.

[58] Mu ˇ cnik, A. A. On the unsolvability of the problem of reducibility in the theory of algorithms. *Dokl. Akad. Nauk SSSR (N.S.) 108* (1956), 194–197.

[59] Novikov, P. On the algorithmic unsolvability of the word problem in group theory. *Trudy Mat. Inst. Steklov 44* (1955), 1–143.

[60] Odifreddi, P. Strong reducibilities. *Bull. Amer. Math. Soc. (N.S.) 4*, 1 (1981), 37–86.

[61] Omanadze, R. S. On the upper semilattice of recursively enumerable sQ-degrees. *Algebra i Logika 30*, 4 (1991), 405–413, 507.

[62] Omanadze, R. S. Quasi-degrees of recursively enumerable sets. In *Computability and models*, Univ. Ser. Math. Kluwer/Plenum, New York, 2003, pp. 289–319.

[63] Omanadze, R. S. Some properties of r-maximal sets and $Q_{1,N}$-reducibility. *Arch. Math. Logic 54*, 7-8 (2015), 941–959.

[64] Omanadze, R. S. On the connections between *wtt*- and Q-reducibilities. *J. Logic Comput. 29*, 1 (2019), 37–51.

[65] Omanadze, R. S., and Chitaia, I. O. Q_1-degrees of c.e. sets. *Arch. Math. Logic 51*, 5-6 (2012), 503–515.

[66] Omanadze, R. v. S. A form of reducibility. *Sakharth. SSR Mecn. Akad. Moambe 83*, 2 (1976), 281–284.

[67] Omanadze, R. v. S. On bounded Q-reducibility. *Soobshch. Akad. Nauk Gruzin. SSR 100*, 1 (1980), 57–60.

[68] Post, E. L. Recursively enumerable sets of positive integers and their decision problems. *Bull. Amer. Math. Soc. 50* (1944), 284–316.

[69] Rabin, M. Computable algebra, general theory and theory of computable fields. *Trans. Amer. Math. Soc. 95* (1960), 341–360.

[70] Rice, H. G. Recursive real numbers. *Proc. Amer. Math. Soc. 5* (1954), 784–791.

[71] Rogers, Jr., H. *Theory of recursive functions and effective computability.* McGraw-Hill Book Co., New York-Toronto-London, 1967.

[72] Shore, R. A. Nowhere simple sets and the lattice of recursively enumerable sets. *J. Symbolic Logic 43*, 2 (1978), 322–330.

[73] Soloviev, V. D. Q-reducibility, and hyperhypersimple sets. In *Probabilistic methods and cybernetics, No. X-XI (Russian)*. Izdat. Kazan. Univ., Kazan′, 1974, pp. 121–128.

[74] Specker, E. Nicht konstrucktiv beweisbare Sätze der Analysis. *Journal of Symbolic Logic 14* (1949), 145–158.

[75] Turing, A. M. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proc. London Math. Soc. (2) 42*, 3 (1936), 230–265.

[76] Turing, A. M. On Computable Numbers, with an Application to the Entscheidungsproblem. A Correction. *Proc. London Math. Soc. (2) 43*, 7 (1937), 544–546.

[77] Yates, C. E. M. Three theorems on the degrees of recursively enumerable sets. *Duke Math. J. 32* (1965), 461–468.