Real Computability
and Hypercomputation

Martin Ziegler

December 20, 2007
Summary: Turing machines are generally agreed to be the appropriate mathematical idealization describing the capabilities of actual digital computers. With regard to computations on real numbers (being continuous rather than digital objects), two major extensions have become well established:

- **Recursive Analysis** considers computation on reals in terms of infinite approximations by fractions of integers.
- **BCSS machines** (‘algebraic model’) perform finitely many arithmetic operations on real numbers exactly.

However, doubts about the validity of the so-called Church-Turing Hypothesis have led to the study of

- notions of (discrete) computability that go beyond the Turing machine, so-called *hypercomputers*.

The present work surveys the three extensions, investigating and comparing the computational power of their various combinations.

I report on external as well as on own work. Concerning the latter, Section 2.3.4 is roughly based on [26, §3.1], Section 2.4.5 on [27, Section 5], Section 3.3.3 on [24], Section 5.3 on [22], Section 5.5 on [30], Section 4.3 on [28], Section 4.4 on [29], and Section 4.4.4 on [33, Section 3.2].

Acknowledgements: Most of this research has been performed as Project Zi 1009/1-1 supported by the German Research Foundation (DFG); I am grateful to the department of Electrical Engineering, Computer Science, and Mathematics of the University of Paderborn and particularly to Professors BÜRGISSE and MEYER AUF DER HEIDE for they kind hospitality, advice, and latitude. First contributions had been obtained while working at the department of Mathematics and Computer Science (IMADA) under host KLAUS MEER with project 21-04-0303 of the Statens Naturvidenskabelige Forskningsråd (SNF). Another major part, generously funded by JSPS scholarship PE 05501, was developed during an inspiring stay at the Japan Advanced Institute of Science and Technology (JAIST) under host ISHIHARA HAJIME.
# Contents

## 1 Introduction

1.1 Computability ................................................. 9
   1.1.1 Basics ............................................... 9
   1.1.2 Applications ....................................... 10

1.2 Models of Computation ...................................... 11

1.3 Computational Complexity .................................. 11

1.4 Church–Turing Hypothesis .................................. 12

1.5 Physics of Computation ..................................... 12
   1.5.1 Theoretical Physics and Computer Science .......... 13
   1.5.2 Meta-Theory of Physics ............................. 13
   1.5.3 Constructivism in Physics ......................... 13

1.6 General Outline ............................................. 14

## 2 Real Computation

2.1 Meta-Mathematical Remarks ................................ 15

2.2 Real Numbers ................................................ 15
   2.2.1 Algebra ............................................. 16
   2.2.2 Topology ............................................. 18
   2.2.3 Continuity and Calculus ........................... 20
   2.2.4 Semi-Algebraic Geometry ......................... 20

2.3 BCSS Machines ............................................... 21
   2.3.1 BCSS Recursion Theory ............................. 22
   2.3.2 Uncomputability .................................... 23
   2.3.3 Computability ...................................... 26
   2.3.4 Use of Real Constants .............................. 27
   2.3.5 Quantifier Elimination .............................. 28

2.4 Recursive Analysis .......................................... 30
   2.4.1 Binary Computability ................................ 30
   2.4.2 Recursive Real Functions ......................... 31
   2.4.3 Semi–Computability and Semi–Decidability ....... 33
   2.4.4 Enumerable Real Open Sets ....................... 34
   2.4.5 Gain and Vain of Non-Uniformity .................. 35
   2.4.6 Type-2 Theory of Effectivity ...................... 36

2.5 BCSS versus/with Recursive Analysis ...................... 39
   2.5.1 Variations of BCSS Machines ..................... 39
   2.5.2 Aspects of Classification ......................... 40
   2.5.3 Analytic Machines ................................ 40

2.6 Further Models .............................................. 41
   2.6.1 Domain Theory ..................................... 42
   2.6.2 Constructive Mathematics ....................... 42
   2.6.3 Markov Computability ............................. 43
   2.6.4 Analog Computation ............................... 44
### CONTENTS

2.6.5 Exact Computation Paradigm ................................................. 44  
2.7 Practice and Efficiency ......................................................... 45  
2.8 My Manifesto ................................................................. 45

### Hypercomputation ................................................................. 47

3.1 Oracle Turing Machines .......................................................... 47  
3.1.1 Power of the Halting Oracle .................................................. 47  
3.1.2 Arithmetical Hierarchy ......................................................... 49  
3.1.3 Hyperarithmetic and Analytical Hierarchy ............................... 51  
3.1.4 Post’s Problem ............................................................... 52

3.2 Physical Hypercomputers ........................................................ 52  
3.2.1 Relativistic Hypercomputation .............................................. 52  
3.2.2 Quantum Mechanical Hypercomputation ................................. 53  
3.2.3 Hypercomputers in Classical Physics ..................................... 54  
3.2.4 N-Body Problem ............................................................ 55

3.3 Further Discrete Hypercomputers .............................................. 55  
3.3.1 Infinite Time ............................................................... 56  
3.3.2 Fair Nondeterminism ......................................................... 56  
3.3.3 Infinite Parallelism ........................................................ 57

3.4 Critique of Hypercomputing .................................................... 59  
3.4.1 Harnessability ............................................................... 60  
3.4.2 Practical Feasibility ........................................................ 60  
3.4.3 Relevance ................................................................. 60

3.5 Real Hypercomputation .......................................................... 60

**II Real Hypercomputation** ....................................................... 63

4 Arithmetical Hierarchies in Recursive Analysis ................................ 65

4.1 Effective Borel Hierarchy ....................................................... 65  
4.1.1 Transfinite Levels .......................................................... 67  
4.1.2 Representing the Borel Hierarchy ......................................... 67

4.2 Arithmetical Hierarchy of Real Numbers ..................................... 69  
4.2.1 Some Relations to the Effective Borel Hierarchy ....................... 71  
4.2.2 Transfinite Levels ........................................................ 72

4.3 Hypercomputing Real Functions .............................................. 72  
4.3.1 Weak Function Evaluation and Continuity ............................. 72  
4.3.2 Hierarchy of Weakly Evaluable Functions ............................... 79  
4.3.3 Relative Weierstraß Computability versus Weak Evaluation ..... 82  
4.3.4 Weak Function Evaluation and Borel’s Hierarchy ..................... 85  
4.3.5 Are Discontinuous Real Functions Hypercomputable? .............. 88  
4.3.6 Markov with Oracles ....................................................... 88

4.4 Revising Type-2 Computation ................................................. 89  
4.4.1 Motivation ............................................................... 89  
4.4.2 The Jump of a Representation ........................................... 90  
4.4.3 Examples of Known Jumps ............................................... 94  
4.4.4 Jump of Set Representations $\theta_\sigma$ and $\psi_\sigma$ .......... 96  
4.4.5 Iterated Jumps and Applications ....................................... 97

4.5 Type-2 Nondeterminism ........................................................ 99  
4.5.1 Computational Power on Finite Inputs ............................... 101  
4.5.2 Computational Power on Infinite Inputs ............................. 102

* A very brief summary of this part (extended abstract) has appeared as 82.
## 5 BCSS Hypercomputation

5.1 Machines with Additional Operations .................................................. 105
  5.1.1 ... and Nondeterminism ......................................................... 106
5.2 Machines with Oracles ........................................................................ 106
  5.2.1 Relativized BCSS Recursion Theory ............................................. 107
  5.2.2 Relative Function Computability ................................................. 108
5.3 Post’s Problem over the Reals ............................................................... 109
  5.3.1 Embedding any Countable Lattice as BCSS Degrees ...................... 111
  5.3.2 The Case of Linear BCSS Machines ........................................... 112
5.4 Arithmetical Hierarchy ....................................................................... 114
  5.4.1 Completeness and Other Applications of the Normal Form Theorem ... 116
  5.4.2 Nondeterministic and Analytic Hierarchy .................................... 118
5.5 Real Computational Universality .......................................................... 118
  5.5.1 Word Problem for Groups .......................................................... 119
  5.5.2 Classical Setting ........................................................................ 119
  5.5.3 Presenting Real Groups ............................................................... 120
  5.5.4 Examples of Presented Real Groups ........................................... 121
  5.5.5 Reducibility to the Real Halting Problem .................................... 123
  5.5.6 Basics from Group Theory and Their Presentations ..................... 124
  5.5.7 First Effectivity Considerations .................................................. 125
  5.5.8 Benign Embeddings ................................................................. 127
  5.5.9 Reduction from the Real Halting Problem .................................. 129

## 6 Concluding Comparison .............................................................. 131

6.1 Discrete versus Real Hypercomputation ......................................... 131
6.2 Superrecursive Analysis versus BCSS Hypercomputers ................. 132
  6.2.1 Comparing their Arithmetical Hierarchies .................................. 132
  6.2.2 Effect of Non-/Uniformity .......................................................... 133
6.3 BCSS to Hyperrecursive Analysis: Equality is a Jump .................... 133
  6.3.1 Open BCSS Semi-Decidable Subsets ........................................ 133
  6.3.2 Continuous BCSS-Computable Functions .................................. 134
6.4 Robust Quasi-Strongly $\delta$-Q-Analytic Functions ......................... 135
  6.4.1 ... and BCSS-Computation or Recursive Analysis ...................... 136
  6.4.2 ... and Type-2 Nondeterminism ................................................. 137
  6.4.3 Finitely Revising Type-2 Computation ....................................... 137
  6.4.4 Finitely Revising Real Representation ...................................... 139
6.5 Omissions ......................................................................................... 139

## Bibliography .......... 140

## Own Publications ................................................. 151

## Index .......... 155
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Simple BCSS Algorithm unrolled into a directed acyclic graph and the set it decides</td>
<td>24</td>
</tr>
<tr>
<td>2.2</td>
<td>Heaviside’s function $h$ and its flipped variant $\overline{h} = 1 - h$</td>
<td>32</td>
</tr>
<tr>
<td>2.3</td>
<td>Relations between some Analytic Machines</td>
<td>41</td>
</tr>
<tr>
<td>3.1</td>
<td>Infinite comb with a wedge to probe</td>
<td>55</td>
</tr>
<tr>
<td>3.2</td>
<td>Sample Initial and Two Successor Configurations of Life</td>
<td>57</td>
</tr>
<tr>
<td>4.1</td>
<td>Classical (bold-face) and effective (light-face) Borel Hierarchy</td>
<td>66</td>
</tr>
<tr>
<td>4.2</td>
<td>The sets $U_z$ defined in the proof of Proposition 4.7</td>
<td>68</td>
</tr>
<tr>
<td>4.3</td>
<td>Illustration of the iterative construction employed in the proof of Claim 4.27</td>
<td>75</td>
</tr>
<tr>
<td>4.4</td>
<td>Illustration of the iterative construction employed in the proof of Claim 4.28</td>
<td>76</td>
</tr>
<tr>
<td>4.5</td>
<td>The first infinitely long iterative construction employed in the proof of Claim 4.29</td>
<td>77</td>
</tr>
<tr>
<td>4.6</td>
<td>Second infinitely long iterative construction employed in the proof of Claim 4.29</td>
<td>77</td>
</tr>
<tr>
<td>4.7</td>
<td>Further infinitely long iterative constructions employed in the proof of Claim 4.29</td>
<td>78</td>
</tr>
<tr>
<td>4.8</td>
<td>Nesting of some rational intervals of dyadic length contained in $f^{-1}[(y - \epsilon, \infty)]$</td>
<td>81</td>
</tr>
<tr>
<td>4.9</td>
<td>Piecewise linear and smooth unit pulse, and a non-overlapping superposition</td>
<td>84</td>
</tr>
<tr>
<td>4.10</td>
<td>Converting a sequence of stabilizing sequences into a sequence with revocations.</td>
<td>94</td>
</tr>
<tr>
<td>4.11</td>
<td>Models of Computation in Chomsky’s Hierarchies over finite/infinite strings</td>
<td>100</td>
</tr>
<tr>
<td>5.1</td>
<td>Correspondence between Classical Discrete and New Real Group Presentations</td>
<td>121</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Complexity is fundamental to theoretical computer science. For a problem $P$, it deals with the question:

How expensive (in terms of resources like memory or, primarily, time) is a computational solution to $P$?

However, logically, the question of computability comes before that of complexity:

Does $P$ admit of a computational solution at all?

Many problems are clearly computable (e.g. by means of exhaustive search); but for others, computability is not obvious or even fails.

1.1 Computability

Historically, computability was not considered an issue at all for a long time. More precisely, in 1900 David Hilbert, as number 10 of his famous list of mathematical problems, asked for a ‘procedure’ (which would nowadays be termed an algorithm) deciding the validity of number theoretical propositions, implying that such an algorithm should ‘obviously’ exist. This common belief, and in fact the entire implicit belief in the foundations of mathematics at that time expressed in Hilbert’s Program, was blown to pieces in 1931 with Kurt Gödel’s Incompleteness Theorem, revealing that even the theory of integers allows propositions which can be neither proven nor refuted, not to mention decided in an automated way. For a problem which is either true or false mathematically but undecidable to a computer, in 1936 Alan Turing introduced what has become known as the Turing Machine (TM) and proved it incapable of deciding the termination of another given TM—the famous Halting Problem $H$. Although this result is considered the dawn of computability theory, one should not forget that at that time ‘computer’ referred to a professional person, the first computing machines in the present sense having been built only in the 1940s. Still, the Turing Machine is nowadays employed as a model less of a human mathematician than of automated digital computing devices, see Section 1.2.

1.1.1 Basics

A language (also called a problem) is a subset $L$ of either the integers $\mathbb{N}$ or the set $\{0, 1\}^*$ of finite binary strings. It is decidable (or recursive) if a Turing machine can, upon input of $x$, within finite time (terminate and) determine whether $x$ belongs to $L$ or not. $L$ is semi-decidable if a Turing machine can terminate on inputs $x \in L$ and diverge for $x \notin L$. $L$ is recursively enumerable (commonly abbreviated as “r.e.”) if a Turing machine is able to output in arbitrary order all elements of $L$. If the complement of $L$ is r.e., $L$ is called co-r.e.

1 Regarding a formal definition, refer to any introductory text.
With the important technique of \textit{dove-tailing} (basically the interleaved simulation of countably many computations), semi-decidability can be shown to be equivalent to recursive enumerability: and recursiveness to both r.e. and co-r.e.

Another crucial ingredient is an effective enumeration of all TMs: there exists a \textit{universal} Turing machine $U$ which, given an encoding of a TM $M$ and input $x$, can simulate $M$ on $x$. The \textit{code} $\langle M \rangle$ of $M$ is called its \textit{Gödel index}, the mapping $M \mapsto \langle M \rangle$ a \textit{Gödelization}. Several codes and inputs may be combined into, and extracted back from, a single integer by virtue of a \textit{pairing function} like for instance

$$\mathbb{N} \times \mathbb{N} \ni (x, y) \mapsto \langle x, y \rangle := 2^x \cdot (2y + 1) \ .$$  \hspace{2cm} (1.1)

A given machine $M$ and input $x$ can be recoded into some $M'$ which already contains $x$ and, more precisely, ignores its own input $x'$ but always behaves like $M$ on $x$. Slightly more generally, the \textit{SMN Theorem} asserts that any (code of a) TM realizing a partial function $(x, y) \mapsto f(x, y)$ and a given $x_0$ may be converted into (the code of) a machine realizing $y \mapsto f(x_0, y)$. The reverse conversion is known as the \textit{UTM Theorem}, namely from a mapping $x \mapsto \langle M_x \rangle$ to the mapping $(x, y) \mapsto f_x(y)$, where $f_x$ denotes the partial function realized by TM $M_x$. SMN and UTM together permit effective adaptation of the arity of functions under consideration and are sometimes referred to as \textit{type conversion}. They also imply that a Gödelization must enumerate every computable function infinitely often: $f_{x_1} = f_{x_2} = \ldots$ for some increasing sequence $x_1, x_2, \ldots$; cf. e.g. \cite{Soar87} \textbf{Padding Lemma I.3.3}.

Now suppose that the Halting problem

$$H = \{ \langle M, x \rangle : M \text{ terminates on input } x \}$$  \hspace{2cm} (1.2)

were decidable. Its complement could then be semi-decided. This in turn would imply that the diagonal problem

$$D = \{ \langle M \rangle : M \text{ does not terminate on } \langle M \rangle \}$$  \hspace{2cm} (1.3)

could be accepted by some machine $M_0$. Now either $\langle M_0 \rangle \in D$ or $\langle M_0 \rangle \notin D$ holds. In the first case, by the definition of $D$, $M_0$ does not terminate on input $\langle M_0 \rangle$—contradicting that $M_0$ accepts $D$, i.e. terminates exactly on inputs from $D$ like $\langle M_0 \rangle$. Similarly, the second case $\langle M_0 \rangle \notin D$ means divergence of $M_0$ on input $\langle M_0 \rangle$ and implies $\langle M_0 \rangle \in D$: again, this is a contradiction.

The restricted (and thus seemingly easier) problem $\{ \langle M \rangle : M \text{ terminates on empty input} \}$ is sometimes also referred to as $H$. In fact each can be reduced to the other by virtue of the UTM theorem and are thus equally undecidable.

\subsection{1.1.2 Applications}

The easy part of the Halting Problem is of course verifying that a TM \textit{will} halt; $H$ is semi-decidable. The difficulty lies in identifying that a machine will continue running forever—which one can never be sure of within finite time. Were that possible, many important open mathematical claims (Goldbach Conjecture, Riemann Hypothesis etc.) could be settled just by setting up a TM $M$ to look for a counter example and checking whether this search eventually succeeds, that is, whether $M$ terminates.

One generally expects that these open questions may ultimately be solved \textit{without} having to resort to the Halting Problem but by mathematical reasonings (and the insights they provide). This has been achieved, e.g. for Fermat’s Last Theorem \cite{Wile95}. However, according to Rice’s Theorem, many other problems can provably be solved only in connection with $H$. Automated software verification, for instance, as desirable as it may be, is principally infeasible in that even the simplest requirement of correct software—termination upon empty input—cannot be decided algorithmically.

Since Turing’s start, a rich variety of further problems has been proven undecidable; to mention two rather famous ones:
• the Word Problem for finitely presented groups \cite{Novi59, Boon58};
• the integral solvability of Diophantine Equations (also known as Hilbert’s Tenth Problem) \cite{Mati70}.

These exhibit serious principal limitations of computational group theory and number theory. Their real counterparts will be among the topics of the present work in Sections \ref{sec5} and \ref{sec2}, respectively. A positive result in computability theory has recently been awarded the Gödel Prize 2002:

• The equivalence of linear pushdown automata can be decided by a Turing machine \cite{Seni01}.

1.2 Models of Computation

A formal abstraction of a computer $M$ is called a model of computation. It describes in sound mathematical terms which operations $M$ is capable of (and what cost they involve in terms of resources). Elevator controls or digital watches, for instance, are conveniently modeled by finite automata.

Algorithms often refer to an underlying model only implicitly. With regard to operations on bits, for example, the TM is generally accepted as the appropriate model. Euclid’s algorithm for calculating greatest common divisors, on the other hand, operates natively by arithmetic on integers, which is more naturally reflected by the Random Access Machine (RAM) model of computation or a \textsc{von Neumann} Machine. Regarding computations involving real numbers, Gaussian Elimination is an example of an algorithm pertaining to the BCSS model, while the Ellipsoid Method exemplifies a model commonly employed in Computable Analysis; see Chapter 2.

A problem’s complexity, as well as its computability, in general largely depend on the model of computation under consideration. The latter should thus be chosen carefully so as to acutely reflect actual computers’ capabilities. Too accurate a model, on the other hand, may be mathematically awkward and intractable to analysis. In any case, every model of computation is an idealization reflecting only some aspects of reality while neglecting others: The Halting Problem for a bounded memory computer (e.g. a nostalgic Commodore C64 or even a state-of-the-art 4GB iMac) is that of a finite automaton and thus decidable by a Turing machine, albeit at a ridiculously high cost in terms of running time.

1.3 Computational Complexity

The complexity of a calculation or decision problem $P$, subject to a model of computation, is the least cost incurred by any algorithmic solution. It thus describes the inherent difficulty of $P$ and allows the performance of a specific algorithm solving $P$ to be assessed. Since there may be an infinite number of solution procedures, the word ‘least’ mathematically corresponds to an infimum, meaning that optimal cost need not be attainable. This holds even when constant factors are neglected as in asymptotic big-$O$ notation. Constants can also be quite sensitive to minor modifications of the model of computation under consideration (For example, on a Turing machine, does a combined bit read/write and head move take one, two, or three steps?) and are therefore best ignored anyway.

Determining the complexity of $P$ amounts to

a) devising algorithms for it and analyzing their respective costs: these constitute upper complexity bounds;

b) complementing the latter by (ultimately matching) lower bounds,

that is, formal proofs establishing that, within the given model, every algorithmic solution of $P$ will incur a cost (asymptotically) at least as high as in a).
The uncomputability of $P$ can thus be regarded as an ultimate form of lower bound: the problem is intractable to algorithmic solution regardless of the allowed cost. Computability on the other hand, that is the construction (or merely the proof of existence) of an algorithm to its solution, precedes and opens the door for further complexity considerations.

1.4 Church–Turing Hypothesis

The Turing machine is nowadays generally employed as the model describing the principal capabilities of digital computers. More precisely, it is widely believed that

“every function which would naturally be regarded as computable can be computed by a Turing Machine.”

This has become known as the Church–Turing Hypothesis and is supported by

a) the continuous failure to come up with a device capable of solving, say, the Halting Problem; (In fact, all computers realized so far, from ZUSE’s Z3 to a MacBook pro, are basically (extremely fast but still) merely Turing Machines.)

b) the successful simulation of a vast variety of physical (and in particular of computing) systems on a Turing machine;

c) formal proofs that several other reasonable yet seemingly unrelated models of computation (e.g. $\mu$–recursion, WHILE–programs, $\lambda$–calculus) are in fact as powerful as the Turing machine.

Nevertheless, it should be kept in mind that the Church–Turing Hypothesis itself has not been and cannot be proven, simply because it is far too informal, with ambiguous notions like ‘naturally’. As a matter of fact, neither ALONZO CHURCH nor ALAN TURING actually put forward a claim as bold as the one above, and current science more carefully distinguishes various variants of this hypothesis; see e.g. [Ord02, Section 2.2] or [Cope02]. For example, the Physical Church–Turing Hypothesis claims that b) above generalizes:

“Every physical process can be simulated by a Turing machine.”

The Strong or Extended Church–Turing Hypothesis also includes complexity considerations in addition to computability, claiming that

“any ‘reasonable’ model of computation can be efficiently simulated on a probabilistic Turing machine.”

1.5 Physics of Computation

Both ambiguities, the notion of ‘the’ Church–Turing Hypothesis (while there are several of them) and the use of terms like ‘natural’ and ‘reasonable’, contribute to intense (and heated) disputes about their validity. Moreover, the involvement of both computer science and physics in the hypothesis increases the sources of possible misunderstandings.

Still, there seems to be a (slowly) growing consensus that the Physical Church–Turing Hypothesis is too bold and to be considered as wrong. For instance it has been pointed out [Gero86, p.546] that the theory of Quantum Gravitation (compare also Section 3.2.2) contains a sum ranging over non-isomorphic simplicial complexes, yet the isomorphism of 4-dimensional simplicial complexes is known to be undecidable by reduction to the Word Problem for groups (Markov 1958); see also Section 5.5
1.5. PHYSICS OF COMPUTATION

1.5.1 Theoretical Physics and Computer Science

A common restricted version of the physical Church–Turing Hypothesis considers not arbitrary physical processes but only those harnessable for universal computation:

**Hypothesis 1.1.** There exists no physical computing device capable of solving the Halting Problem.

It has only recently been noticed and stated explicitly [Sm06a, BeTu06] that the existence and non-existence of a physical object must be seen as subject to an underlying physical theory, that is as relative to a sound mathematical abstraction of (parts of) nature. This is quite similar to un-/computability claims also being subject to a model of computation as a mathematical abstraction of a computing system. Newtonian Mechanics, for instance, does not allow Black Holes but does allow faster-than-light travel; whereas in General Relativity, it is the other way round. A further example, the Second Law of Thermodynamics (non-decreasing entropy), yields a rather interesting ultimate (though very small) lower bound on the speed of irreversible computation (i.e., regarding complexity)—yet seemingly not on computability, i.e., regarding the principal computing power of physical systems [BeLa85]. And it turns out (see Chapter 3) that some physical theories do permit the Halting Problem to be solved!

1.5.2 Meta-Theory of Physics

As we have just seen, there is not just one Theory of Physics but a great variety of such theories; and the status of Hypothesis 1.1 may depend on the term ‘physical’ that it contains being understood according to one theory or to the other. Thus, a thorough investigation of the Church–Turing Hypothesis should consider several theories of physics and their relations to each other (e.g. sub-theories), which brings us to the field of meta-theory.

As promising as such an approach to a sound physical theory of computation may be, it has to cope with a major obstacle: As opposed to a model of computation (which usually somebody suggests and describes formally and which the scientific community then either rejects or accepts and uses), a physical theory is far more provisional in character: once proposed, it usually remains ‘under construction’ with new terms being added or others discarded (see below); and once the theory eventually ‘converges’ towards a stable formulation, it is already on the verge of being superseded by a superior one [Kuhn62].

Newtonian Mechanics for example, though successful for centuries in describing both small falling apples and large planets circling the sun, was eventually replaced by Quantum Mechanics describing the very, very small (e.g. elementary particles), and by Relativity Theory describing the very, very large (e.g. galaxies). General Relativity in turn used to contain a parameter, the cosmological constant \( \Lambda \), whose introduction Einstein himself later called the “biggest blunder” of his life and reverted, declaring \( \Lambda = 0 \); whereas nowadays observations of the relationship between distance and redshift indicate that \( \Lambda \) might in fact have a very small yet non-zero value. Moreover, reconciling Relativity with Quantum Mechanics has led to Relativistic Quantum Field Theories, though these still lack mathematical well-foundation.

Consequently, so very few physical theories have actually been formalized far enough (in the sense of successfully inhibiting misinterpretations by other physicists) that even the question “What is a physical theory anyway?” splits the community [Wei77, Ludw78, Schr96].

1.5.3 Constructivism in Physics

Apart from such issues, the term “exists” in Hypothesis 1.1 involves further ambiguity of interpretation. As has been pointed out in [24, Remark 1.4], this is quite similar to the case of ‘idealistic’ versus constructive mathematics: In the former, the following proposition holds true due to the principle of excluded middle (tertium non datur):

**Principle 1.2.** A sequence \((x_n)_{n \in \mathbb{N}}\) is either identically zero; or there exists some index \(n \in \mathbb{N}\) with \(x_n \neq 0\).
In constructive mathematics, however, this Limited Principle of Omniscience (LPO), is deprecated because the existence proof in the second case does not actually construct an \( n \) with \( a_n \neq 0 \). (Section 2.6.2 below expands on the close relations between constructive analysis and real number computability.) This is to be distinguished from LPO being non-contradictory, that is, from its double negation [Brid04]. Even among mathematicians (actually the majority) who do accept the LPO, attitudes differ concerning what can be regarded as its uncountable version:

**Principle 1.3 (Axiom of Choice).** Let \( I \) and \( X_i, i \in I \) denote arbitrary sets. If \( \prod_{i \in I} X_i = \emptyset \), then there exists some \( i \in I \) where \( X_i = \emptyset \).

In classical mathematics, i.e. with the permission to use proof by contradiction, this principle indeed also seems ‘obvious’; and is furthermore equivalent to several other famous and natural mathematical propositions: e.g. that every vector space admits a basis [Blas84]. Nevertheless, Principle 1.3 neither follows from nor contradicts conservative set theory [Goed40]; this justifies calling it an axiom and leaves it a matter of taste whether to include it in or to exclude it from a formal system.

These examples illustrate that some mathematical object to exist can be interpreted as

A) to insist that one actually constructs this object (and thus to reject LPO);

B) its non-existence leading to a contradiction (and thus to accept LPO but not necessarily the Axiom of Choice);

C) the object’s existence to not lead to a contradiction (and to accept e.g. the Axiom of Choice).

Similarly, the existence of a physical object as in Hypothesis 1.1 may be understood in the strong sense (A) of actually constructing it. Yet in most cases, showing it (C) to be consistent with physical laws is accepted as well; observe that this is how both positrons and Black Holes came into ‘existence’: as putative solutions of (and thus consistent with) Dirac’s Equation and Einstein’s General Relativity, respectively. It was only later that new experimental observations upgraded our conception of their existence to type (B).

### 1.6 General Outline

This concludes our brief introduction and background information to computability in the classical setting, dealing with well-established models of computation (primarily the Turing machine) for bits, words, integers, and rational numbers. We will now proceed to review more recent research, dropping the restriction to discrete problems (Chapter 2) and considering super-Turing models of computation (Chapter 3). Their synthesis then leads to the most current area of real hypercomputation (Part 1), which constitutes the core, and justifies the title, of the present work.
Chapter 2

Real Computation

‘Standard’ models of computation—Turing machines, Random Access Machines (RAMs), and even parallel variants like the PRAM—all operate on discrete objects: bits, Booleans, integers; at most, rational numbers (fractions) are considered in the form of numerator/denominator pairs of integers. These are the entities to be read, stored, processed, and output.

A large amount of scientific computation, however, evolves around continuum rather than discrete problems. Fields like fluid dynamics, computational material science, and space mission design are just a few of them. In fact, most of applied mathematics amounts to solving various classes of ordinary or partial differential equations over the reals. This raises the need for a formal model (see Section 1.2) to describe and analyze the prospects and limits of computations over \( \mathbb{R} \). Introducing the two major ones will be the purpose of Sections 2.3 and 2.4.

2.1 Meta-Mathematical Remarks

Some people question the need for a model of real (not to mention complex) computation. They argue, convincingly indeed, that scientific computation deals with rationals only, if not with floating point numbers. This is of course mostly true (except for the algebraic numbers handled in LEDA \cite{BKM95}, for example, and even transcendentals as in computer algebra systems): as true as the observation that all quantities obtainable from physical experiments—and thus, by extensionality, entire nature itself—are but rationals; objects like \( \sqrt{2} \) and \( \pi \) being made up ‘artificially’.

In the last consequence, such considerations lead to an ultrafinitistic point of view (quite consistent with floating point numbers). While I secretly agree with these opinions, the majority of mathematicians nevertheless consider real numbers and computations on them for a simple reason: assigning to certain sequences of rational numbers (like \( x_{n+1} = \frac{x_n^2 + 2}{2x_n} \)) a short name (like \( \sqrt{2} \)) to talk about, and granting this abbreviation a mathematical identity to calculate with (like \( \sqrt{2} \cdot \sqrt{2} = 2 \)), yields such enormous succinctness, elegance, and ease of comprehension (for instance, what is the uniform probability distribution on \([0,1]\) considered as the unit interval of rationals?) as to make this idealization well worth the ontological overhead. Moreover, insights obtained in an idealized realm have often led to interesting implications back in a more applied world; e.g. complex to real analysis.

We are thus facing two sides of Occam’s Razor: extending (finitely constructable, countable cardinality) rationals \( \mathbb{Q} \) to (not finitely constructable, continuum cardinality) \( \mathbb{R} \) yields a tremendous increase in conceptional complexity, which it is fair to call unnecessary; on the other hand the transition from the minimal-ontological, ‘practical’ world \( \mathbb{Q} \) to the Platonic realm of idealizations \( \mathbb{R} \) does benefit from an elegance and ease of description within \( \mathbb{R} \) which brings with it a decrease in conceptional complexity: compare, for example, the simplicity of Newton’s Law in its differential form (i.e. referring to \( \mathbb{R} \)) to one based on \( \mathbb{Q} \) only. Equipping \( \mathbb{R} \) with a notion of computability (i.e. a certain sense of infinite constructability by finitely describable machines) can be regarded as a compromise between the two extremes.
To sum up, pure mathematics and theoretical computer science have both contributed and still contribute essentially to the undeniable success of scientific computing—and vice versa. Having said this, the present section proceeds to review common models of real number computation: the two most common being the algebraic BCSS machine and the Type-2 machine (approximation paradigm) in Sections 2.3 and 2.4, respectively. (Also prepare to encounter another personal comment in Section 2.8.)

2.2 Real Numbers

Before addressing computation on reals, let us look at $\mathbb{R}$ itself. For many purposes, common high school intuition is fairly sufficient: $\frac{1}{2}$, $\sqrt{2}$, and $\pi$ are real numbers; there are arithmetic operations $+,-,\times,\div$ and a linear order $<$; and one can take limits of (converging) sequences. Many computer or numerical scientists have often, in the course of their daily business, forgotten what reals really ‘are’, that is, their construction or definition. In fact, $\mathbb{R}$ is far more complicated than, say, the set $\mathbb{Q}$ of fractions (which, including number theory, can justly be regarded as a fairly complex mathematical structure itself), even just in terms of size: $\text{Card}(\mathbb{R}) = 2^\text{Card}(\mathbb{Q}) > \text{Card}(\mathbb{Q})$, there exists no surjective mapping from $\mathbb{Q}$ onto $\mathbb{R}$. The question of whether there exist sets of cardinality strictly between $\mathbb{Q}$ and $\mathbb{R}$ is known as the Continuum Hypothesis and has been proven logically independent of ZFC (Zermelo-Fraenkel set theory with axiom of choice) by GÖDEL and COHEN. We briefly recall two classes of properties of $\mathbb{R}$ which are trivial on $\mathbb{Q}$: algebraic and topological ones. Each turns out to correspond to one major model of real number computation.

2.2.1 Algebra

Like $\mathbb{Q}$, $\mathbb{R}$ is an (ordered) field to which the general concepts of abstract algebra apply:

**Definition 2.1.** Fix a subfield $E$ of the field $F$.

- a) For $A \subseteq E$, let $F(A)$ denote the smallest subfield of $E$ containing $F$ and $A$.
- b) The degree $[E : F]$ is the dimension of $E$ considered as an $F$–vector space.
- c) For $e \in E$, call $\deg_F(e) := [F(e) : F]$ the degree of $e$.
- d) If $\deg_F(e) < \infty$, $e$ is algebraic over $F$ otherwise transcendental over $F$.
- e) Field extension $E$ is algebraic over $F$ if every element $e \in E$ is algebraic over $F$.
- f) For an integral domain $R$, a non-zero polynomial $p \in R[X]$ is called irreducible (over $R[X]$) if it is not constant and if every factorization $p = q_1 \cdot q_2$ with $q_1,q_2 \in R[X]$ requires $q_1$ or $q_2$ to be constant.

When dealing with $E = \mathbb{R}$, we occasionally omit $F$ in the case that $F = \mathbb{Q}$ and denote by $\mathbb{A}$ the set of algebraic reals.

**Fact 2.2.**

- a) The degree of iterated field extensions is multiplicative: For fields $F \subseteq E \subseteq D$, $[D : F] = [D : E] \cdot [E : F]$.
- b) For fields $F \subseteq E \ni e$, $e$ is algebraic over $F$ iff there exists a non-zero polynomial $p$ with coefficients in $F$ such that $p(e) = 0$.
- c) More precisely, let $e$ be algebraic over $F$. Then $F(a) = F[a]$ and $\deg_F(e) < \infty$ coincides with the degree of every irreducible $p \in F[X] \setminus \{0\}$ satisfying $p(e) = 0$.
- d) If $R$ is a unique factorization domain and $F$ its field of fractions, then a polynomial $p \in R[X]$ irreducible over $R[X]$ is also irreducible over $F[X]$. 
2.2. REAL NUMBERS

e) To an algebraic field extension $E$ over $F \supseteq \mathbb{Q}$ of finite degree, there exists some $e \in E$ (a so-called primitive element) such that $E = F[e]$.

Proof. Cf. e.g. [Lang93] Proposition VII.§1.4. Claim d) is known as Gauss’ Lemma [Lang93 Section V.§6]. Regarding e), $E$ has characteristic 0 and is therefore a separable extension of $F$. [Cohn91 Corollary 3.4.3]; now apply [Cohn91 Theorem 3.9.2]. \[ \square \]

Roughly 2,500 years ago, HIPPAUS OF METAPONTUM proved (and was drowned for doing so) that $\sqrt{2} \notin \mathbb{Q}$; equivalently: that $[\mathbb{Q}(\sqrt{2}) : \mathbb{Q}] = 2$. This proof immediately extends to squarefree numbers $t$ and single higher roots: $[\mathbb{Q}(t^{1/n}) : \mathbb{Q}] = n$ for $n \in \mathbb{N}$. But how about field extensions by several roots? Here are three answers:

Lemma 2.3. a) Let $\mathbb{Q} \subseteq F \subseteq \mathbb{R}$ denote a tower of fields, $t \in \mathbb{N}$ be squarefree, and $n_1, \ldots, n_k \in \mathbb{N}$. Then $F(\sqrt[n_1]{\ldots}, \sqrt[n_k]{\ldots})$ coincides with $F(\sqrt[t]{\ldots})$, where $N := \text{lcm}(n_1, \ldots, n_k)$ denotes the least common multiple. In particular $[F(\sqrt[n_1]{\ldots}, \sqrt[n_k]{\ldots}) : F] = N$.

b) Let $t_1, \ldots, t_k \in \mathbb{N}$ be distinct and squarefree. Then $[\mathbb{Q}(\sqrt{t_1}, \ldots, \sqrt{t_k}) : \mathbb{Q}] = 2^k$.

c) For $d \in \mathbb{N}$, let $p_1, \ldots, p_k$ denote distinct prime numbers and $n_1, \ldots, n_k \in \mathbb{N}$. Then

$$[\mathbb{Q}(\sqrt[n_1]{p_1}, \ldots, \sqrt[n_k]{p_k}) : \mathbb{Q}] = n_1 \cdot n_2 \cdots n_k .$$

The first two claims admit elementary proofs; that of b) is quoted from [Rick00].

Proof. a) Recall that the following properties of lcm:

$$n_1 \mid \text{lcm}(n_1, \ldots, n_k) = \text{lcm}(\text{lcm}(n_1, \ldots, n_{k-1}), n_k)$$

and

$$\text{lcm}(a, b) = ab/\gcd(a, b) = ab/(ra + st) \quad \text{with } r, s \in \mathbb{Z} .$$

Therefore, each $t^{1/n}$ is a power of $t^{1/N}$ and thus in $F(\sqrt[t]{\ldots})$; while, conversely, $t^{1/\text{lcm}(a, b)} = (t^{1/a})^s \cdot (t^{1/b})^r \in F(\sqrt[t]{\ldots})$ yields $t^{1/N}$ to belong to $F(\sqrt[n_1]{\ldots}, \ldots, \sqrt[n_k]{\ldots})$ by induction on $k$.

Hence we have indeed established $t^{1/N}$ as a primitive element.

b) Define $S_n (n \geq 0)$ to be the set of square roots of products of subsets of some $n$ distinct primes. (So $S_0$ has $2^n$ elements, $S_0 = \{1\}$, $S_1 = \{1, \sqrt{2}\}$, $S_2 = \{1, \sqrt{2}, \sqrt{3}, \sqrt{6}\}, \ldots$) Define $F_n$ to be the subset of $\mathbb{R}$ generated as a vector space over $\mathbb{Q}$ by the elements of $S_n$. Then:

i) The elements of $S_n$ are linearly independent over $\mathbb{Q}$. (This yields the claim!)

ii) Every element of $F_n$ whose square is rational is a rational multiple of an element of $S_n$.

iii) $F_n$ is a field.

I prove this proposition by induction over $n$. It is clearly true for $n = 0$ ($S_0 = \{1\}$, $F_0 = \mathbb{Q}$). Suppose it is true for $n = k$. Let $p$ be the $(k + 1)$-st prime, $r = \sqrt{p}$. Then every element of $F_{k+1}$ is of the form $a + br$, where $a$ and $b$ are in $F_k$.

If $S_{k+1}$ is not linearly independent, then 0 can be expressed in this form with $a$ and $b$ linear combinations of elements of $S_k$, not both $a$ and $b$ being the trivial linear combination: since $S_k$ is linearly independent, this implies that $a$ and $b$ are not both zero. Since $a + br = 0$, it follows that neither $a$ nor $b$ is zero; we have $r = -a/b$ (using the fact that $F_k$ is a field), and $p = (a/b)^2$; this contradicts (ii) for $n = k$, using unique factorization in the integers and counting powers of $p$. Hence we have (i) for $n = k + 1$.

Now $(a + br)^2 = a^2 + pb^2 + 2abr$; if this is rational, then since $S_{k+1}$ is linearly independent, $2ab = 0$, so either $a = 0$ or $b = 0$. If $a = 0$, then $(a + br)^2 = pb^2$, so if this is rational then $b^2$ is rational, so $b$ is a rational multiple of an element of $S_k$, so $a + br = br$ is a rational multiple of an element of $S_{k+1}$; similarly if $b = 0$ then $a + br$ is a rational multiple of an element of $S_{k+1}$ (indeed, of an element of $S_k$). This shows (ii) for $n = k + 1$. 
$F_{k+1}$ is clearly a ring; to show that it is a field, it suffices to show that every non-zero element is invertible. This is true because \((a + br)^{-1} = (a - br)/(a^2 - pb^2); a^2 - pb^2\) is non-zero because it equals \((a + br) \cdot (a - br)\), so it is invertible because it is in $F_k$, which is a field. This shows (iii) for $n = k + 1$.

c) Cf. [Besi40 Theorem 2]; see also [Albu03 bottom of p.2].

**Transcendence**

**Definition 2.4.** For fields $F \subseteq E$, a set $A \subseteq E$ is algebraically dependent over $F$ if there exists $n \in \mathbb{N}$, distinct $a_1, \ldots, a_n \in A$, and a non-zero polynomial $p \in F[x_1, \ldots, x_n]$ such that $p(a_1, \ldots, a_n) = 0$. The transcendence degree $\text{tr.deg}_F(E)$ of $E$ over $F$ is the cardinality of a maximal, algebraically independent subset of $E$.

**Fact 2.5.**

\begin{itemize}
\item[a)] $e$ and $\pi$ are transcendental.
\item[b)] The transcendence degree of $\mathbb{R}$ over $\mathbb{Q}$ is of continuum cardinality [Cohn91 Exercise 5.1(4)].
\item[c)] Let $a_1, \ldots, a_n$ be algebraic yet linearly independent over $\mathbb{Q}$. Then $e^{a_1}, \ldots, e^{a_n}$ are algebraically independent over $\mathbb{Q}$ [Bake75 Theorem 1.4].
\item[d)] For fields $F \subseteq E \subseteq L$, $\text{tr.deg}_F(L) = \text{tr.deg}_F(E) + \text{tr.deg}_E(L)$.
\end{itemize}

Claim a) follows from c) which in turn is known as the Lindemann-Weierstraß Theorem. Claim d) can be regarded as a transcendent and additive variant of the multiplicativity of the algebraic degree (Fact 2.2a); cf. e.g. [Lang93 Exercise X.5]. Regarding the following claim, refer for instance to [Lang93 Exercise X.6]:

**Fact 2.6.** Let $E$ denote a field extension of $F$ which is finitely generated in the sense that $E = F(x_1, \ldots, x_d)$ for finitely many $x_1, \ldots, x_d \in E$. Then any intermediate field between $E$ and $F$ is also finitely generated.

**2.2.2 Topology**

$\mathbb{R}$ is by construction the metric completion of $\mathbb{Q}$: the set of limits of rational Cauchy sequences:

\[(x_n)_n \subseteq \mathbb{Q}, \quad |x_n - x_m| \leq 2^{-n\cdot m} \quad \Rightarrow \quad \exists x \in \mathbb{R} : |x - x_n| \leq 2^{-n+1} .\]

The same applies to the finite Cartesian product $\mathbb{R}^k, k \in \mathbb{N}$. Equipped with the Euclidean norm, $\mathbb{R}^k$ is a locally compact, separable, connected space with a countable base of open rational balls $B(\bar{q}, r) = \{ \bar{x} : |\bar{x} - \bar{q}| < r \}, \bar{q} \in \mathbb{Q}^k, 0 < r \in \mathbb{Q}$. The class of open subsets of $\mathbb{R}^k$ is denoted by $\mathcal{O}^k$; we write $A^k$ for the class of closed subsets; $\overline{S}$ and $S^\circ$ for the closure and interior, respectively, of a set $S$.

**Fact 2.7.** Let $X$ denote an arbitrary topological space, $I$ an arbitrary index set, and $G, H, M_i \subseteq X, i \in I$. Then

\[
\begin{align*}
1. \bigcup_{i \in I} \overset{\circ}{M_i} & \subseteq \left( \bigcup_{i \in I} M_i \right)^\circ \\
2. \cap_{i \in I} M_i & \supseteq \bigcap_{i \in I} M_i \\
3. G^\circ \cap H^\circ & = (G \cap H)^\circ \\
4. (X \setminus G)^\circ & = X \setminus \overset{\circ}{G} \\
5. \cap_{i \in I} \overset{\circ}{M_i} & \supseteq \left( \bigcap_{i \in I} M_i \right)^\circ \\
6. \bigcup_{i \in I} M_i & \subseteq \bigcup_{i \in I} \overset{\circ}{M_i} \\
7. \overset{\circ}{G} \cup H & = \overset{\circ}{G \cup H} \\
8. \overset{\circ}{X \setminus H} & = X \setminus \overset{\circ}{H} \\
9. U \subseteq \overset{\circ}{U} & \quad 11. U = A^\circ \quad \Rightarrow \quad \overset{\circ}{U} = U \\
10. A \supseteq \overset{\circ}{A} & \quad 12. A = \overset{\circ}{U} \quad \Rightarrow \quad \overset{\circ}{A} = A .
\end{align*}
\]

Let $U$ denote an open and $A$ a closed subset of $X$. 

2.2. REAL NUMBERS

In Euclidean space, every non-empty open set $U$ is of second category (BAIRE’s Category Theorem), that is $U = \bigcup_n A_n$ implies $\exists n : \overline{A}^c \neq \emptyset$.

Cantor Space

arises by taking the binary expansions of real numbers as the set $\{0,1\}^\omega$ of infinite 0/1–sequences equipped with metric

$$d(\bar{\sigma}, \bar{\tau}) = \max\{2^{-n} : n \in \mathbb{N}, \sigma_n \neq \tau_n\}.$$

It is compact, separable, and totally disconnected.

Borel Hierarchy

Fix a topological space $X$. We write (bold-face) $\Sigma_1$ for the class of open subsets of $X$, $\Pi_1$ for its complements, i.e. the closed subsets of $X$, and $\Delta_1$ for those which are both open and closed (clopen): $\Delta_1 \supseteq \{\emptyset, X\}$ with equality iff $X$ is connected.

Countable unions of closed sets, known as $F_\sigma$, form the class $\Sigma_2$; $\Pi_2$ consists of $G_\delta$ sets, that is countable intersections of open ones; and $\Delta_2 = \Sigma_2 \cap \Pi_2$.

Example 2.8. For $X = \mathbb{R}$, the set $\mathbb{Q}$ of rational numbers, being a countable union of (closed) singletons, belongs to $\Sigma_2$; but not to $\Pi_2$ [Boto79 A13.5].

The Cantor set, consisting of all real numbers in $[0,1]$ having a ternary expansion involving only digits 0 and 2 (but not 1), is closed (i.e. in $\Pi_1$) and uncountable yet nowhere dense [GeOl03 §8.1].

$\Sigma_3$ contains all countable unions of members from $\Pi_2$; $\Pi_3$ its complements and so on for $\Sigma_d$ and $\Pi_d$, $d \in \mathbb{N}$ — the (bold-face) Borel Hierarchy of subsets of $X$. For $X = \mathbb{R}^k$ and $X = \{0,1\}^\omega$ one can show it to be strict [Kech95]:

$$\Delta_1 \subsetneq \Sigma_1 \subsetneq \Delta_2 \subsetneq \Sigma_2 \subsetneq \ldots \subsetneq \Delta_d \subsetneq \Sigma_d \subsetneq \ldots \quad (2.1)$$

Transfinite Levels

To the finite Borel Hierarchy $\Sigma_{<\omega}$ ($\omega$ the first infinite ordinal) belong exactly the sets $\bigcup_{d \in \mathbb{N}} S_d$ where $S_d \in \Sigma_d$; their complements constitute $\Pi_{<\omega}$. Countable unions over the latter form $\Sigma_{<\omega+1} = \Sigma_\omega$ and so on until $\Sigma_{<\omega+\omega}$, then $\Sigma_{<2\omega}$, $\ldots$, $\Sigma_{<3\omega}$, $\ldots$, $\Sigma_{<4\omega}$, $\ldots$, $\Sigma_{<\omega^2}$, etc: transfinite induction yields a class $\Sigma_{\alpha}$ for any ordinal $\alpha$. Again, strictness holds for $X = \mathbb{R}^k$ and $X = \{0,1\}^\omega$: $\Sigma_\alpha \cup \Pi_\alpha \subsetneq \Delta_\beta$ whenever $\alpha < \beta$, provided that $\alpha$ is a countable ordinal! As a matter of fact the transfinite Borel hierarchy of levels $\alpha$ running through all countable ordinals exhausts the $\sigma$–algebra $\Delta_1^\sharp$ of Borel sets, that is the smallest collection of subsets of $X$ containing the open ones such that both complements and countable unions remain within this collection.

Analytic and Projective Hierarchy

Borel sets are not closed under projection: this error of Lebesgue was spotted by Suslin, see e.g. [Kech95 Theorem 14.2]. The projections of Borel sets constitute the class $\Sigma_1^1$ of analytic\footnote{Not to be confused with sets in the (light-face, i.e. normal print) analytical hierarchy in Section 3.1.3} sets:

$$\Sigma_1^1 := \{ \text{proj}_X(A) : A \subseteq X \times X \ G_\delta \}, \quad \text{proj}_X(A) = \{ x : \exists y : (x, y) \in A \} \quad (2.2)$$

where $X = \{0,1\}^\omega$ [Kech95 Exercise II.14.3] or $X = \mathbb{R}^k$ [Mosc80 top of p.59]. Analytic sets are closed under projection, their complements form the class $\Pi_1^1$. The famous Suslin Theorem asserts that $\Sigma_1^1 \cap \Pi_1^1$ coincides with the above $\sigma$–algebra $\Delta_1^\sharp$ of Borel sets and was the starting point for descriptive set theory.

Projections of $\Pi_1^1$–sets (in $X \times X$) yield $\Sigma_2^1$, their complements $\Pi_2^1$ and so on: the Projective Hierarchy.
2.2.3 Continuity and Calculus

For topological spaces $X$ and $Y$, a (total) function $f : X \to Y$ is continuous if the pre-image $f^{-1}[V] = \{x \in X : f(x) \in V\}$ of every open set $V \subseteq Y$ is again open in $X$; equivalently: the pre-image of every closed set is closed. Furthermore, if $X$ is a first countable space (such as a subset of some $\mathbb{R}^k$), $f : X \to Y$ is continuous iff it is sequentially continuous: Whenever sequence $(x_n) \subseteq X$ converges to $x \in X$, the image sequence $(f(x_n)) \subseteq Y$ converges to $f(y)$.

For $Y = \mathbb{R}$, $f$ is lower semi-continuous if its pre-image under interval $(y, \infty)$ is open for every $y \in \mathbb{R}$; equivalently [Rand68, Chapter 6.7]: $f(\lim_n x_n) \leq \lim \inf_n f(x_n)$ for all convergent sequences $(x_n)$.

An analytic function, that is a power series $x \mapsto \sum_{n=0}^{\infty} a_n x_n$ with positive radius of convergence, is identically zero $a_n \equiv 0$ iff it vanishes on some infinite subset of its domain.

The uniform limit of (continuous) polynomials is again continuous. Conversely, Weierstraß has proven

Fact 2.9 (Weierstraß Approximation). Every continuous real function $f : K \to \mathbb{R}$ on a compact Euclidean subset $K \subseteq \mathbb{R}^k$ is the uniform limit of rational polynomials $P \in \mathbb{Q}[X_1, \ldots, X_k]$:

$$\sup_{x \in K} \| f - P_n \| =: \| f - P_n \|_K \to 0.$$  \hfill (2.3)

2.2.4 Semi-Algebraic Geometry

as it were combines Sections 2.2.1 and 2.2.2 by studying (the topology of) sets of solutions to finite systems of real polynomial equalities and inequalities.

Definition 2.10. Fix a field $F \subseteq \mathbb{R}$ and $d \in \mathbb{N}$. A set

$$\bar{B} = \{ \bar{x} \in \mathbb{R}^d : p_1(\bar{x}) = \ldots = p_k(\bar{x}) = 0 \land q_1(\bar{x}) > 0 \land \ldots \land q_\ell(\bar{x}) > 0 \}$$  \hfill (2.4)

of solutions to a finite system of polynomial (in)equalities with $p_1, \ldots, p_k, q_1, \ldots, q_\ell \in F[X_1, \ldots, X_d]$ is called basic semi-algebraic over $F$. A subset of $\mathbb{R}^d$ semi-algebraic over $F$ is a finite union of ones that are basic semi-algebraic over $F$. It is countably semi-algebraic over $F$ if the union involves countably many members, all being basic semi-algebraic over $F$.

Observe that every semi-algebraic set is semi-algebraic over some finitely generated real field extension of $\mathbb{Q}$. It is well-known (BÉZOUT, BEN-OR, ...) that any semi-algebraic set has only finitely many connected components; and each such connected component is again semi-algebraic (over the same field $F$) [BPR05, Section 5.2]. Moreover, every non-empty set semi-algebraic over $F$ contains a point which is (component-wise) algebraic over $F$.

Definition 2.11. A field $F \subseteq \mathbb{R}$ is real closed if $0 \neq p \in F[X]$ and $p(x) = 0$ for $x \in \mathbb{R}$ implies $x \in F$.

$\mathbb{R}$ itself is of course real closed. An important non-trivial example is given in the following

Fact 2.12. The set $\mathbb{R}_c$ of real numbers with Turing-decidable binary expansion is a real closed field!

Proof. Cf. e.g. [Wei00, Corollary 6.3.10] in combination with [Wei00, Lemma 4.2.1].

That projections of semi-algebraic sets are again semi-algebraic is far from obvious, but a consequence of TARSKI’s Quantifier Elimination. Slightly more generally in the framework of real closed fields and computation over algebraic structures, it holds

Fact 2.13. Fix $d, k \in \mathbb{N}$, a field $F \subseteq \mathbb{R}$ and real closed field $E$ with $F \subseteq E \subseteq \mathbb{R}$. Let $\bar{B} \subseteq E^{d \times k}$ be the set of solutions (restricted to $E$) of a finite system of polynomial inequalities with coefficients from $F$. Then the set

$$\bar{B} := \{ \bar{x} \in F^d : \exists y_1 \in F \forall y_2 \in F \exists y_3 \in F \ldots \theta_k y_k \in F : (\bar{x}, y_1, \ldots, y_k) \in \bar{B} \}$$
is the set of solutions (restricted to $E$) of a finite system of polynomial inequalities with coefficients from $F$, too.

Moreover, the latter system can be obtained from the former by algebraic computation over $(F, +, -, \times, \div, <)$; even uniformly in $d, k, F$.

Proof. See e.g. [BPR03, Section 2.5.1].

## 2.3 BCSS Machines

A Turing machine over alphabet $\{0, 1\}$ can, with its tape, store, process, and modify in each step one bit; including the capability to remember in its finite control (corresponding to a ‘program’) a finite number of bits. RAMs similarly handle integers $\mathbb{Z}$ as basic entities, finitely many of them appearing as constants in a program. From a computability point of view, the Turing machine and RAM are equivalent since they can obviously simulate each other—although possibly at the expense of an exponential slowdown [Scho79].

Regarding that $\{0, 1\} = \mathbb{F}_2$ and $\mathbb{Z}$ are both rings, this leads to the $R$–machine as a joint generalization of both Turing machine and RAM, where $R$ denotes some ring [BCSS98]; or, more generally, to an $A$–machine where $A$ denotes some algebra [Tuck80, TuZu00]. In the spirit of the original paper [BSSS80], we shall mostly confine ourselves to the algebra $A = (\mathbb{R}, +, -, \times, \div, <)$ of (ordered) reals, occasionally also taking into consideration the $(\mathbb{R}, +, -, <)$ and $(\mathbb{C}, +, -, \times, \div, =)$. An $\mathbb{R}$–machine $M$ can in each step add, subtract, multiply, divide, and branch on the result of comparing two reals. Its memory consists of an infinite sequence of cells, each capable of holding a real number and accessed via two special index registers (similar to a two-head Turing machine). A program for $M$ may store a finite number of real constants.

Prior to its formal introduction in 1989, this model of computation had of course already been used implicitly in most numerical algorithms [Cuck02] and is visible behind the standardized semantics of common programming languages such as FORTRAN [ANSI-F]. Indeed, a floating point operation, whether implemented in software or hardware [IEEE75], generally takes time (bounded) independently of the value being processed. An $R$–machine is also the programming interface which computer algebra systems typically provide the user with [GaGe03]. Finally, the BCSS model clearly benefits from its mathematical elegance and, as we shall see below, from the powerful tools of abstract algebra, algebraic geometry, and model theory applicable to it. The major disadvantage: it neglects issues of accuracy and precision in numerical calculations. This may occasionally limit its practical relevance and have counter-intuitive consequences (Example 2.22) but is no reason to generally abandon it; recall (Section 1.2 and Chapter 2) that indeed every model of computation, and the real numbers themselves, result from idealization and therefore differ from reality in some way or another.

**Definition 2.14.**

a) Let $X \subseteq \mathbb{R}^* := \bigcup_{d \in \mathbb{N}} \mathbb{R}^d$, i.e. a set of finite sequences of real numbers. Its dimension, $\dim(X)$, is the smallest $D \in \mathbb{N}$ such that $X \subseteq \bigoplus_{d \leq D} \mathbb{R}^d$; $\dim(X) = \infty$ if no such $D$ exists.

b) A BCSS-machine $M$ over $\mathbb{R}$ is a finite set of instructions labeled by $1, \ldots, N$. A configuration of $M$ is a quadruple $(n, i, j, \bar{y}) \in I \times N \times N \times \mathbb{R}^*$. Here, $n$ denotes the currently executed instruction, $i$ and $j$ are used as addresses (copy registers) and $\bar{y}$ is the actual content of the registers of $M$. The initial configuration of $M$’s computation on input $\bar{x} \in \mathbb{R}^*$ is $(1, 1, 1, \bar{x})$. If $n = N$ and the actual configuration is $(N, i, j, \bar{y})$, the computation stops with output $\bar{y}$. The instructions $M$ is allowed to perform are of the following types:

**computation:** $n : y_s \leftarrow y_k \circ_n y_l$, where $\circ_n \in \{+,-,\times,\div\}$; or $n : y_s \leftarrow \alpha$ for some $\alpha \in \mathbb{R}$.

The register #s will get the value $y_k \circ_n y_l$ or $\alpha$, respectively. All other register entries remain unchanged. The next instruction will be $n + 1$; moreover, the copy register $i$ is either incremented by one, or replaced by 0, or remains unchanged. The same holds for copy register $j$. 
branch: \( n: \text{if } y_0 \geq 0 \text{ goto } \beta(n) \text{ else goto } n + 1. \) The next instruction (where \( \beta(n) \in I \))
is determined according to the answer to the test. All other registers are not changed.

copy: \( n: y_i \leftarrow y_j, \) i.e. the content of the “read” register is copied to the “write” register.
The next instruction is \( n + 1; \) all other registers remain unchanged.

c) We call a partial function \( f : \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^* \) **BCSS-computable**
if a BCSS-machine \( M \) realizes it; specifically, computation of \( M \) has to succeed on inputs \( \bar{x} \in \text{dom}(f) \) and diverge for \( \bar{x} \notin \text{dom}(f). \) Some set \( L \subseteq \mathbb{R}^* \) is **semi-decidable** if
it is the domain of a computable function; \( L \) is **BCSS-decidable** iff its characteristic function \( \chi_L : \mathbb{R}^* \rightarrow \{0, 1\} \) is BCSS-computable.
We call \( L \) a **decision problem** (or a **language**) over \( \mathbb{R}^*. \)

An instruction “\( y \leftarrow \alpha \)” corresponds to access to a predefined constant \( \alpha \) (like \( \sqrt{2} \), \( \pi \) or \( \exp(1) \) etc.)
of which a program, having finite length \( N, \) can contain at most finitely many. The computability
of single reals thus being trivial in the BCSS-model, let us turn to function computation. The
input to and the output from \( M \) are vectors \( \bar{x} \in \mathbb{R}^d \) accompanied with the (in general varying)
dimension \( d, \) encoded e.g. as \((d, x_1, \ldots, x_d) \in \mathbb{R}^{d+1}. \)

**Example 2.15.**
a) Gaussian Elimination immediately yields a BCSS algorithm for computing
matrix ranks and solving systems of linear equations.

b) The Simplex Algorithm realizes a BCSS solution to systems of linear inequalities. (See,
however, Example 2.18c)

c) The set \( \mathbb{N} \) of integers is BCSS-decidable:

Given \( x \in \mathbb{R}, \) accept if \( x = 0; \) reject if \( 0 < x < 1; \) otherwise iterate with \( x \mapsto x - 1. \)

Note that the number of steps performed by the BCSS machine is in a) polynomially bounded (by \( \mathcal{O}(n^3) \) for \( n \times n \) matrices), in b) exponentially bounded (this bound being attained by Klee–Minty Cubes; see also [31]), and in c) finite yet unbounded\(^4\) in the dimension \( d \) of the input.

**Remark 2.16.** The present work focuses on computation over \( \mathbb{R}. \) Another variant, the
complex BCSS-machine defined and operating over the algebra \((\mathbb{C}, +, -, \times, \div, =), \) is common as well
[BCL04, BCL05]. Many but not all results carry over to this case: Decomposition of \( z = x + iy \)
into real or imaginary parts \( x, y \in \mathbb{R}, \) for instance, is not feasible, nor is comparison “\( z > 0 \)” even
when the compared value \( z \) is known (e.g. from a restriction imposed to the inputs) to be real
and thus ordered. In particular, and as opposed to Example 2.15c, the set of integers can be shown to
be undecidable to a complex BCSS-machine.

### 2.3.1 BCSS Recursion Theory

The BCSS machine is nowadays mostly employed for complexity theory investigations [MeMi97, BCS97, BuCu05, MaMe07], yet since its origins computability has also been considered [BSS89, Sections 7–10] and gave rise to seminal results like [BCSS98, Section 2.4] and [Meer93]; see also [MeMi97, Section 5]. This theory shares many—though not all, cf. Corollary 2.20 structural similarities with its classical counterpart; compare Section 1.1.4

**Fact 2.17.**
a) A real language \( L \subseteq \mathbb{R}^* \) is decidable iff both \( L \) and its complement are semi-decidable.
The graph \( \{(\bar{x}, \bar{f}(x)) : \bar{x} \in \text{dom}(f)\} \) of a computable total function \( f \) is decidable;
that of a partial function is semi-decidable.

b) Non-empty \( L \) is semi-decidable iff it is enumerable in the sense of coinciding with the
range \( \{f(R^n) \} \) of a computable total function \( f : \mathbb{R}^* \rightarrow \mathbb{R}^*. \)

---

\(^4\)Some publications seem ambiguous about this divergence requirement.

\(^5\)Refer to [CMP92] for a closer investigation of this effect. Over complex numbers, however, a computation which terminates on all inputs \( \bar{x} \in \mathbb{C}^n \) does so in bounded time [CuRo93].

\(^5\)Even if it is only countable, \( L \) in general need not coincide with the image \( f[N] \) of a computable function: see Example 2.28 below.
2.3. BCSS MACHINES

c) One can naturally encode a BCSS machine $M$ as $⟨M⟩ ∈ \mathbb{R}^*$ and thus obtain a real Gödelization, together with a universal BCSS machine, and obtain SMN and UTM properties.

d) The real Halting problem

$$H = \{⟨M, x⟩ : M \text{ terminates on input } x\} ⊆ \mathbb{R}^*$$

(2.5)

is undecidable to a BCSS machine.

Regarding the Gödelization in Fact 2.17c), the idea is to put the (finitely many) machine constants $\alpha_1, \ldots, \alpha_D$ into the components of the vector $⟨M⟩$, plus an additional integer describing the program’s (discrete) control flow as in the classical Turing case; compare [BCSS98, Section 3.5].

The one additional dimension $D + 1$ can be saved by virtue of a mixed real/integer pairing function like

$$\mathbb{R} × \mathbb{N} \ni (x, n) ↦ → (\lfloor x \rfloor, n) ∈ \mathbb{R}.$$  

(2.6)

There is, however, no BCSS–computable fully real pairing function $\mathbb{R} × \mathbb{R} → \mathbb{R}$, cf. Corollary 2.20.

2.3.2 Uncomputability

The proof of Fact 2.17d) proceeds identically to that of the classical case in Section 1.1.1. In the discrete realm, such diagonalization arguments (and their variations, see Section 3.1.4) seem to be virtually the only approach to negative results. A particularly nice feature of BCSS computability theory is (sometimes so-called pure) algebra as a second ingredient and tool to prove claims like the following:

Example 2.18. The following real languages are BCSS undecidable:

a) the sets $\mathbb{Q}$ of rational numbers and $A$ of algebraic reals as well as the Cantor set;

b) the Mandelbrot Set

$$\{(x, y) ∈ \mathbb{R} : ∃k ∈ \mathbb{N} \forall n ∈ \mathbb{N} : \left| p_{x+iy} \circ \ldots \circ p_{x+iy}(0) \right| ≤ k\}, \quad p_c(z) := z^2 + c$$

and the set of starting points $x_0 ∈ \mathbb{R}$ leading to convergence of Newton’s Iteration $x_n ↦ → x_{n+1} := x - \frac{f(x)}{f'(x)}$ where $f(x) := x^3 - 2x + 2$.

c) Integer Linear Programming, that is feasibility over $\mathbb{Z}^d$ of a system of linear inequalities with real coefficients, is also undecidable to a BCSS machine for $d ≥ 2$.

d) the functions $[0, ∞) ↦ → \sqrt{x}$ and $\mathbb{R} ⊃ x ↦ → \exp(x)$ are both BCSS-uncomputable.

In fact, $\text{graph}(\exp)$ is not BCSS semi-decidable.

Item c) follows from a) by reduction: Given $a ∈ \mathbb{R}$,

$$a \cdot x ≤ y \land a \cdot x ≥ y \land x ≥ 1$$

is a system of linear inequalities which admits of a solution $(x, y) ∈ \mathbb{Z}^2$ iff $a ∈ \mathbb{Q}$.

In general, proofs of negative claims in the BCSS setting proceed based on a so-called path decomposition [BCSS98, §2.3 Theorem 1], that is based on unrolling a putative algorithm (or rather machine) $M$ into a (possibly infinite) algebraic computation tree $T$ [BCS97, Section 4.4]: Every branch of $M$ is executed according to the sign of a (previously computed) quolynomial $f$, evaluated on the input vector $⃗{x} ∈ X$; compare Figure 2.1. A branching node $u$ in this tree

---

6We follow A. SCHÖNHAGE in calling the quotient $p/q$ of two (possibly multivariate) polynomials a quolynomial, as opposed to the term “rational function”, which may be confused with $f ∈ \mathbb{Q}[X]$. 
Figure 2.1: Simple BCSS Algorithm unrolled into a directed acyclic graph, and the set it decides

thus leads to a partition\(^7\) of the set \(X_u \subseteq X\) of inputs \(\bar{x} \in X\) arriving at \(u\) into the three subsets 
\{\(\bar{x} \in X_u : f(\bar{x}) < 0\)\}, \{\(\bar{x} \in X_u : f(\bar{x}) = 0\)\}, and \{\(\bar{x} \in X_u : f(\bar{x}) > 0\)\} which form in turn the respective sets of inputs of the three successor nodes of \(u\) in \(T\). Starting with \(M\)'s domain \(\mathbb{R}^d\) 
(i.e. \(X_r\) of \(T\)'s root \(r\)), straightforward induction on the depth of \(u\) in \(T\) reveals \(X_u\) to be basic semi-algebraic (recall Section 2.2.4); more precisely: basic semi-algebraic over the field extension \(F := \mathbb{Q}(c_1, \ldots, c_D) \subseteq \mathbb{R}\) generated by the (finitely many!) real constants \(c_1, \ldots, c_D\) stored in \(M\). 
Moreover, the set \(X\) of inputs leading to termination of \(M\) coincides with the disjoint union of all \(X_v\), \(v\) ranging over the (at most countably many) leaves of \(T\); and on each \(X_v\), \(M\) calculates a quodnomial with coefficients from \(F\).

Now every semi-algebraic set has only finitely many connected components (Section 2.2.4); so 
a semi-decidable set, being a countable union of semi-algebraic \(X_u\), can have at most countably many 
connected components: but \(\mathbb{R} \setminus \mathbb{Q}\), \(\mathbb{R} \setminus \mathbb{A}\), and the Cantor set each consist of uncountably 
many isolated points, and therefore cannot be semi-decidable, thus establishing Example 2.18a)

\(^7\)Notice that, if \(f \equiv 0\) on \(X_u\), the branch is trivial. We will w.l.o.g. suppose that such dispensable tests/nodes 
\(u\) have been removed from the tree \(T\) under consideration.
by virtue of Fact 2.17(a).

Item b) is from BCSS98 Section 2.4. In order to prove c), exploit that

i) a BCSS machine computes, on every leaf $v$, a quolynomial $f_v|_{X_u}$;

ii) a quolynomial is defined uniquely by its values on an infinite set;

iii) neither $\exp(x)$ nor $\sqrt{x}$ are quoleynomials.

Lemma 2.19 below shows $X_u$ to indeed contain an open set for some $u$ and thus concludes the proof of Example 2.18(d). Ingredient iii) is of course well-known; nevertheless, the following two elegant arguments might be of interest to the reader: First suppose that $\exp(x) \cdot p(x) = q(x)$ for polynomials $p, q$ of degree at most $d$. Then $(d + 1)$-fold differentiation annihilates the right hand side $q$, whereas the derivative of $\exp(x) \cdot p(x)$ according to the product rule maintains the form $\exp(x) \cdot \tilde{p}(x)$, where $\tilde{p}(x) = p(x) + p'(x)$ has the same degree as $p$: contradiction. Secondly, let $\deg(p/q) := \deg(p) - \deg(q)$ denote the degree of a quolynomial; it extends that of polynomials, is independent of factors common to $p$ and $q$, integral valued, and satisfies $\deg(f \cdot g) = \deg(f) + \deg(g)$ (over characteristic 0). In particular, if $x \mapsto \sqrt{x}$ were quolynomial, then $1 = \deg(\text{id}) = \deg(\sqrt{-}) = 2 \deg(\sqrt{\cdot})$ would contradict $\deg \in \mathbb{Z}$.

\begin{proof}
Let $T$ be an algebraic computation tree (not necessarily obtained from unrolling a BCSS computation with finitely many constants) whose domain $X \subseteq \mathbb{R}^d$ includes an open set. Then there exists a leaf $v$ for which $X_v$ is open.

\begin{proof}
Apply Baire’s Category Theorem (recall Section 2.2.2) to the set equation $X = \bigcup_{v \text{ leaf of } T} X_v$:

Some $X_v$ must have non-empty interior. On the other hand, this $X_v$ is a finite intersection of closed $p^{-1}_i[0]$ and open $q^{-1}_i(0, \infty)$ sets (Definition 2.10). Each $p^{-1}_i[0]$ must have non-empty interior according to Fact 2.7(b), i.e. $p_i \equiv 0$ by uniqueness of analytic functions (Section 2.2.2). So $X_v \neq \emptyset$ is an intersection of open sets only.

The above argument applies to BCSS machines over arbitrary analytic functions. For the usual case of computation over $(+, -, \times, \div)$, here is an alternative more algebraic

\begin{proof}
Let $c_1, c_2, \ldots, c_n, \ldots$ denote the (countably many) real coefficients of the quoleynomials $f_u$ appearing in $T$’s nodes $u$. By Fact 2.5(b) there exist $t_1, t_2, \ldots$ such that $A := \{c_1, c_2, \ldots, t_1, t_2, \ldots\}$ is algebraically independent. That is, if a multi-variate polynomial with rational coefficients vanishes on any finite subset of $A$, then it is identically zero. In particular, a $d$–variate polynomial over the field extension $\mathbb{Q}(c_1, c_2, \ldots)$, whose value on $\vec{t} = (t_1, \ldots, t_d)$ equals 0, vanishes identically—in contradiction to Footnote 7. Since the algebraic independence of $A$ is invariant under rational translations $t_i \mapsto t_i + q_i$ for $q_i \in \mathbb{Q}$, we may, by density, assume $\vec{t} \in X$ and conclude: For any $f_u$ present in $T$, it holds $f_u(\vec{t}) \neq 0$. (The existence of such a $\vec{t}$ would otherwise have been concluded by appealing to Baire’s Theorem.) The leaf $v$ of $T$ which input $\vec{t}$ ends up in has therefore as $X_v$, a finite intersection of its predecessor conditions, each either of the form “$f_u > 0$” or “$f_u < 0$”, but not “$f_u = 0$”. By continuity of $f_u$, these conditions correspond to open sets $X_u$; hence $X_v$ is open, too.

A related argument—with algebraic instead of transcendence degree—will play a major role in Section 5.3.

\end{proof}

\end{proof}

Corollary 2.20. There is not BCSS–computable real pairing function:

Let $\emptyset \neq U \subseteq \mathbb{R}^2$ be open and $f : U \rightarrow \mathbb{R}$ injective. Then $f$ is BCSS–uncomputable.

We give two arguments, one based on a (much stronger) result in Algebraic Topology and the other on an (also far more general) one in Semi-Algebraic Geometry.
CHAPTER 2. REAL COMPUTATION

Proof. Suppose \( f \) is BCSS–computable. According to Lemma 2.19, \( f \) is a quolynomial on some non-empty open \( U' \subseteq U \).

Put one way, \( f \) is therefore continuous, so apply Brouwer’s Invariance of Domain, cf. e.g. [Deim85, Theorem 4.3]. Put another way, \( f \) is semi-algebraic and thus accessible by [BPR03, Lemma 5.26]. Both independently ensure \( f[U'] \subseteq \mathbb{R} \) to be open in \( \mathbb{R}^2 \); contradiction.

2.3.3 Computability

The negative results from Section 2.3.2 are complemented by (some surprising) positive ones, that is, problems solvable by a BCSS machine. As opposed to Example 2.18d) for instance, the graph \( \{(x,y) : y \geq 0, x = y^2\} \subseteq \mathbb{R}^2 \) of \( \sqrt{\cdot} \) is easily decidable. Here comes a counterpart to Example 2.18a):

Example 2.21. Both the sets \( \mathbb{Q} \) of rational numbers and \( \mathbb{A} \) of algebraic reals are semi–decidable: Given \( x \in \mathbb{R} \), try all pairs \((r,s) \in \mathbb{Z}^2\) for whether \( x = r/s \) and abort when found: this search terminates iff \( x \in \mathbb{Q} \). Similarly for \( \mathbb{A} \) try, according to Fact 2.2b), all non-zero quolynomials \( p \in \mathbb{Q}[X] \{0\} \) whether \( p(x) = 0 \).

One generalization of this observation is the computation of a totally discontinuous partial function, taken from [Weih00, Example 9.7.2]:

Example 2.22. The following function is BCSS–computable:

\[
f : \subseteq \mathbb{R} \rightarrow \mathbb{R}, \quad x \mapsto \begin{cases} 1 : & x \in \mathbb{Q} \\ 0 : & x \notin \mathbb{Q} \land x^2 \in \mathbb{Q} \\ \bot : & x^2 \notin \mathbb{Q} \end{cases}
\]

Upon input of \( x \), semi–decide (cmp Example 2.21) whether \( x^2 \in \mathbb{Q} \); if so, test whether \( x \in \mathbb{Q} \) (and output 0 or 1 accordingly): take the fractional representation \( x^2 = r/s \) found along the way, make \( r, s \in \mathbb{Z} \) co-prime, and determine whether both are integer squares.

Another generalization asserts the computability of the algebraic degree \( \deg(a) \) of \( a \in \mathbb{A} \), recall Definition 2.1

Lemma 2.23. The function \( \deg : \mathbb{A} \rightarrow \mathbb{N} \), \( a \mapsto \deg(a) \) is BCSS–computable.

In view of Example 2.18a), restricting the degree function to algebraic numbers is of course essential here; in other words: While for reasons of mathematical convenience one may define \( \deg(x) := \infty \) for transcendental \( x \), a BCSS machine cannot detect that case.

Proof. Enumerate all non-constant \( p \in \mathbb{Z}[X] \) irreducible over \( \mathbb{Z}[X] \) — a property in classical \( \mathbb{N}P \) by virtue of [Can81] and thus BCSS–decidable. Evaluate each such \( p \) on \( a \): if \( p(a) = 0 \), return \( \deg(p) \) and terminate; otherwise continue with the next \( p \).

By virtue of Fact 2.2, this search will succeed and yield the correct value.

Remark 2.24. An elementary decision procedure for irreducibility in \( \mathbb{Z}[X] \) proceeds as follows (although not within nondeterministic polynomial time): Given \( p \in \mathbb{Z}[X] \) of degree \( n - 1 > 0 \) and content 1, choose some \( n \) arbitrary distinct arguments \( x_1, \ldots, x_n \in \mathbb{Z} \) and multi-evaluate \( y_i := p(x_i) \). Observe that, if \( q \in \mathbb{Z}[X] \) is a non-trivial divisor of \( p \), then \( z_i := q(x_i) \) divides \( y_i \) for each \( i = 1, \ldots, n \). This suggests going through all (finitely many) choices for \( (z_1, \ldots, z_n) \in \mathbb{Z}^n \) with \( z_i \mid y_i \), calculating the interpolation polynomial \( q \in \mathbb{Q}[X] \) to data \( (x_i, z_i) \), and checking whether its coefficients are integral and whether \( q \) divides \( p \).
2.3.4 Use of Real Constants

The ability of a BCSS machine to store real constants (like $\sqrt{2}, \pi, c$) can also be ‘abused’:

**Example 2.25.** Encode the discrete Halting problem $H \subseteq \mathbb{N}$ into the binary expansion of a real number by letting $c := \sum_{n \in H} 2^{-n} \in \mathbb{R}$. A BCSS machine may extract the binary digits back from $c$ by iteratively computing (as in the proof of Example 2.17(a)) $c_0 := c$ and $c_{n+1} := c_n - \lfloor c_n \rfloor$. Especially if $c$ is stored, it can decide $H$ (but of course not $H$ in view of Fact 2.17(b)).

Similarly, every discrete problem is BCSS–decidable; every function $f : \subseteq \mathbb{N} \to \mathbb{N}$ is BCSS–computable by encoding $f$ into a real constant $c = \langle f \rangle$; cf. [BSS93, Example 6].

To avoid such unrealistic power, the literature is occasionally restricted to BCSS machines with no (or, equivalently, only rational) constants; see e.g. [Koi97a, Koi97b] or [BuCu03, Definition 2.4]. Alternatively, one may permit constants—provided that their binary expansion is classically describable—that is, $c \in \mathbb{R}$ computable in the sense of Recursive Analysis below (cf. Lemma 2.40 and Example 2.37).

A different question concerns the **number** of real constants: Every discrete problem can be decided by a BCSS machine with just one $c \in \mathbb{R}$; whereas for real problems, Example 2.29 below reveals that each additional cell permitted to store an arbitrary real constant yields a strict gain in computational power. Its proof is based on our following characterization of BCSS–computable functions inspired by [Mich90], a strengthening of [BCSS98, §2.3 Theorem 1] and precise observation of which infinite algebraic decision trees arise from unrolling a BCSS computation and which do not.

**Lemma 2.26.** For $f : \subseteq \mathbb{R}^* \to \mathbb{R}^*$, and $c_1, \ldots, c_j \in \mathbb{R}$, consider the following claims:

a) $f$ is computable by a BCSS Machine with constants $c_1, \ldots, c_j \in \mathbb{R}$.

b) There is an integer sequence $(d_n)_n$ such that $\text{dom}(f) = \bigcup_n B_n$ is the countable disjoint union of sets $B_n \subseteq \mathbb{R}^{d_n}$ semi-algebraic over field extension $F := \mathbb{Q}(c_1, \ldots, c_j)$, and each restriction $f|_{B_n}$, $n \in \mathbb{N}$, is a polynomials with coefficients from $F$.

b) There exists $c_{j+1} \in \mathbb{R}$ such that $f$ is computable by a BCSS Machine with constants $c_1, \ldots, c_j, c_{j+1}$.

Then a) implies b), from which in turn c) follows.

Roughly speaking, the difference between uniformity (a,c) and non-uniformity (b) amounts to at most one additional real constant.

**Proof.** The implication “a)$\Rightarrow$b)” follows from unrolling the computations according to Section 2.3.2. To see “b)$\Rightarrow$c)” observe that, for each $n \in \mathbb{N}$, $B_n$ and $f|_{B_n}$ can be described by and computed from $(d_n, n \in \mathbb{N})$ coefficients of the polynomials constituting $f|_{B_n}$ and the system of (in)equalities (2.4). These coefficients in turn lie by prerequisite in $F = \mathbb{Q}(c_1, \ldots, c_j)$, that is, are polynomials in $c_1, \ldots, c_j$ with coefficients from $\mathbb{Q}$ and thus describable with finitely many bits. Thus, encoding these data for all $n \in \mathbb{N}$ into $c_{j+1} \in \mathbb{R}$ in a way similar to Example 2.25 we arrive at the desired BCSS machine.

So for an arbitrary finite number of real constants, we obtain [Cuck92, Theorem 2.4]:

**Corollary 2.27.** a) A set $L \subseteq \mathbb{R}^*$ is BCSS semi–decidable iff it is countably semi-algebraic over a finitely generated field extension.

b) A function $f : \subseteq \mathbb{R}^* \to \mathbb{R}^*$ is BCSS–computable iff $\text{dom}(f)$ is countably semi-algebraic over some finitely generated field extension $F$ and, on each basic semi–algebraic piece, $f$ is a quolynomial over this $F$.

In particular if $B \subseteq F^*$ for a finitely generated field extension $F \subseteq \mathbb{R}$ and $f$ is computable, then $f|_B$ lies in (Cartesian products of) a finitely generated field extension, too.
Maybe the gist about BCSS computability amounts to the observations that any discrete information can be encoded into a constant, and real data processing proceeds within the field generated by the input and the machine’s constants.

**Example 2.28.** The countable set \( \mathbb{L} = \{2^{1/n} : n \in \mathbb{N}\} \) is decidable but not the range \( f[\mathbb{N}] \) of a computable real function \( f \) on \( \mathbb{N} \).

**Proof.** Given \( 1 < x \in \mathbb{R} \), testing “\( x^n = 2 \)” for \( n \in \mathbb{N} \) yields a semi-decision procedure; rejecting \( x \) if \( x^n > 2 \) shows \( \mathbb{L} \) to be decidable. On the other hand, the field generated by \( \mathbb{L} \subseteq \mathbb{R} \) is algebraic of infinite degree (recall Lemma 2.3a) and hence not finitely generated (Fact 2.2e).

\( \square \)

**Example 2.29.** Consider \( d \) distinct prime numbers \( p_i \) and take the exponentials of their respective square roots:

\[ \mathbb{B} \subseteq \{ \exp(\sqrt{2}), \exp(\sqrt{3}), \exp(\sqrt{5}), \exp(\sqrt{7}), \exp(\sqrt{11}), \exp(\sqrt{13}), \exp(\sqrt{17}), \ldots \}, \ |\mathbb{B}| = d. \]

\( \mathbb{B} \) is decidable by a BCSS machine with \( d \) real constants but not even semi-decidable by one with \( d - 1 \) of them.

**Proof.** Decidability by a machine with \( |\mathbb{B}| = d \) constants \( c_i := \exp(\sqrt{p_i}) \) is obvious. Suppose semi-decidability of \( \mathbb{B} \) by a machine with \( d - 1 \) real constants. By virtue of Lemma 2.20(a\( \Rightarrow \)b), it is countably semi-algebraic over some rational field extension \( F = \mathbb{Q}(c_1, \ldots, c_{d-1}) \); even (finely) semi-algebraic because \( |\mathbb{B}| < \infty \). According to \( \text{[BPR03} \text{ SECTION 5.2]} \), each connected component (i.e. each singleton \( \{c_j\} \)) contains (and thus equals) a point algebraic over \( F \). On the other hand, \( \mathbb{B} \)'s transcendence degree exceeds that of \( F \): The algebraic numbers \( \sqrt{p_i} \) are well-known to be linearly independent over \( \mathbb{Q} \) (Lemma 2.3b); hence their exponentials are algebraically independent by virtue of Fact 2.5(b).

\( \square \)

### 2.3.5 Quantifier Elimination

Over Booleans and integers, a language is semi-decidable (i.e. the domain of a computable function) iff it coincides with the range of a total computable function (is r.e.). In order to assert the same over the reals (Fact 2.17b), it is easy to convert a semi-decision procedure \( M \) for \( \mathbb{B} \neq \emptyset \) to the computation of a total function \( f : \mathbb{R}^* \rightarrow \mathbb{R}^* \) with \( f[\mathbb{R}^*] = \mathbb{B} \): Store some \( \bar{b} \in \mathbb{B} \) as a constant.

Upon input of \( (z_1, \ldots, z_n, t) \), simulate \( M \) on \( \bar{z} = (z_1, \ldots, z_n) \) for at most \( t \) steps: if it terminates before, then output \( \bar{z} \in \mathbb{B} \); otherwise output \( \bar{b} \), qed. For the converse, we may in the discrete case semi-decide “\( z \in \text{range}(f) \)” by evaluating \( f \) on all possible arguments \( x \in \mathbb{N} \) and terminate when one with \( f(x) = z \) is found. However, that approach does not work over the reals having uncountably many candidates \( x \) for \( f(x) = z \). A nondeterministic BCSS machine may guess \( x \) and verify \( f(x) = z \). But here the issue arises again: How can one decide deterministically the language of a nondeterministic machine which may make continuous real guesses—as opposed to the ‘only’ exponentially many discrete ones available to a classical nondeterministic Turing machine? This is where Tarski’s Quantifier Elimination (Fact 2.13) comes into play: it implies

**Lemma 2.30.** Hilbert’s Tenth Problem over the reals, that is the question of whether a given system of polynomial (in)equalities \( (2.4) \) has a solution \( \bar{x} \in \mathbb{R}^d \) (or, to put it differently, whether the emptiness of a given semi-algebraic set) is BCSS-decidable!

**Proof.** Let \( \bar{y} \in \mathbb{R}^k \) denote the given vector encoding the polynomial (in)equalities. These conditions \( p_i(\bar{x}) = 0 \) and \( q_j(\bar{x}) > 0 \) amount to constant-free polynomial (in)equalities in \( (\bar{x}, \bar{y}) \). In particular, the set \( \mathbb{B} \) of tuples \( (\bar{x}, \bar{y}) \) where \( \bar{x} \) solves (the system encoded by) \( \bar{y} \) is semi-algebraic over \( \mathbb{Q} \). We are interested in the membership problem of \( \bar{y} \) to \( \mathbb{B} := \{ \bar{y} \mid \exists \bar{x} : (\bar{x}, \bar{y}) \in \mathbb{B} \} \). This is decidable because, according to Fact 2.13 one can compute a system of polynomial (in)equalities over \( \mathbb{Q} \) whose solutions \( \bar{y} \) exhaust exactly \( \mathbb{B} \).
Corollary 2.31. a) Non-emptiness of the language accepted by a BCSS machine, i.e., the

\[ \{ (M) \mid \exists \vec{x} \in \mathbb{R}^* : M \text{ terminates on } \vec{x} \} , \]

is semi-decidable.

b) A problem (semi-)decidable by a nondeterministic BCSS machine is also (semi-)decidable by a
deterministic one.

Now Fact 2.17b) follows easily: Given \( \vec{z} \in \mathbb{R}^* \), nondeterministically ‘guess’ \( \vec{x} \in \mathbb{R}^* \) and verify that

both \( \vec{x} \in \text{dom}(f) \) and \( f(\vec{x}) = \vec{z} \) hold: if they do not, diverge. Corollary 2.31b) in turn is an easy
consequence of a) and the UTM/SMN property.

The proof of Corollary 2.31b) applies Lemma 2.30 to each leaf-set \( \mathbb{X}_v \) of the algebraic
computation tree as obtained by the following variant of Lemma 2.26b) uniform in \( M \):

Fact 2.32 (Effective Path Decomposition). Given a code \( (M) \) of a BCSS machine \( M \) with
constants \( c_1, \ldots, c_D \), one can enumerate (coefficients over \( \mathbb{Q}(c_1, \ldots, c_D) \) of) the sequence \( (f_v)_{v \in \overline{T}} \)
of quolynomials and (polynomial systems whose solutions constitute) semi-algebraic sets \( (\mathbb{X}_v)_{v \in \overline{T}} \)
associated with the nodes \( u \in \overline{T} \) of the algebraic computation tree \( \overline{T} \) induced by \( M \) in the sense of
Section 2.3.2.

Here, an element \( a \) of \( \mathbb{Q}(c_1, \ldots, c_D) \) is encoded by a \((D+1)-\text{variate polynomial } p \over \mathbb{Q}\) and
rational numbers \( \alpha < \beta \) such that \( a \) is, within the interval \((\alpha, \beta)\), the unique zero of \( p(\cdot; c_1, \ldots, c_D) \).

When deciding emptiness “\( \mathbb{X}_v = \emptyset \)?”, this implicitness of the description of the polynomial system
for leaf-set \( \mathbb{X}_v \) can be dealt with by introducing \( a \) as a further variable and applying Quantifier
Elimination as in the proof of Lemma 2.30.

In View of Real Constants...

Re-examination of the above proof of Fact 2.17b) reveals the following strengthening (generalized later in Theorem
5.28):

Scholium 2.33. For \( c_1, \ldots, c_D \in \mathbb{R} \) and \( L \subseteq \mathbb{R}^* \), the following are equivalent:

a) \( L \) is semi-decidable by a BCSS-machine with constants \( c_1, \ldots, c_D \).

b) \( L = \text{range}(f) \) for some partial \( f : \subseteq \mathbb{R}^* \rightarrow \mathbb{R}^* \) computable by a BCSS-machine with constants
\( c_1, \ldots, c_D \).

If \( L \neq \emptyset \), they in turn imply

c) There exists \( c_{D+1} \in \mathbb{R} \) such that \( L = \text{range}(g) \) for some total \( g : \mathbb{R}^* \rightarrow \mathbb{R}^* \) computable by a
BCSS-machine with constants \( c_1, \ldots, c_D, c_{D+1} \).

Indeed, neither the universal BCSS-machine used to enumerate the paths of \( M \) in Fact 2.32 nor
Quantifier Elimination (recall Fact 2.13), need any constants in addition to those already present in
(or given by the description of) \( M \); similarly for the easy converse implication “\( a \Rightarrow b \)” where \( f(\vec{x}, t) \)
is defined as the output of \( M \) simulated on \( \vec{x} \) for \( t \) steps (recall the beginning of this Section 2.3.5).

However in order to establish, in c), \( L \) as the range of a total function, the above argument needs
closer inspection: storing some arbitrary \( \vec{b} \in L \) may require additional constants. Observe that
\( L \) contains a point \( \vec{b} \) algebraic over \( F \): combine Lemma 2.26 with Section 2.2.4. So \( E := F(\vec{b}) \)

is an algebraic extension of \( F \) of degree at most \( d := \text{dim}(\vec{b}) < \infty \), hence \( E = F[\beta] \) for some \( \beta \in \mathbb{R} \)
by Fact 2.2c). In particular, \( \vec{b} \in E^d = \mathbb{Q}(c_1, \ldots, c_D, \beta)^d \) can be stored using the one additional
\( c_{D+1} := \beta \).

The general necessity of such a \((D+1)-\text{st constant can easily be seen from

Example 2.34. \( L := \{ \sqrt{2} \} \) is semi-decidable by a constant-free machine but not the range of a
total \( f : \mathbb{R}^* \rightarrow \mathbb{R}^* \) computable without constants: Any such function must have \( f(0) \in \mathbb{Q} \).
2.4 Recursive Analysis

We now consider another way of extending the classical Turing machine from \{0, 1\} (or \(\mathbb{Z}\) or \(\mathbb{Q}\)) to \(\mathbb{R}\) that is quite different from the BCSS model. Recall how numerical programming uses rational numbers (like \(\frac{\pi}{7}\) and \(\frac{\pi}{11}\)) as approximations to reals (like \(\sqrt{2}\) and \(\pi\)), expecting to approach the exact result as precision increases, that is, in the limit:

**Definition 2.35.** Call \(x \in \mathbb{R}\) **naively computable** if there exists a Turing machine to generate a sequence \((q_n)_{n \in \mathbb{N}}\), \(q_n \in \mathbb{Q}\), with \(\lim_{n \to \infty} q_n = x\).

The drawback of this notion (and the reason for ‘naively’ in its name) arises from the observation that a sequence like 3, 3.1, 3.14, 3.141, 3.1415 may converge not to \(\pi\) but, continuing as \((0, 0, \ldots)\) say, to 0. More precisely, any finite initial segment of a sequence allows no conclusions whatsoever about its limit; one has to ‘wait’ for the entire infinite output \((q_n)_{n \in \mathbb{N}}\) to be computed—not a very practical idea (although it will prove useful when real hypercomputation is considered in Section 2.4.2). As a remedy, let us equip the rational approximations with error bounds:

**Definition 2.36.** Call \(x \in \mathbb{R}\) **Cauchy–computable** if there exists a Turing machine to generate two rational sequences \((q_n)_{n \in \mathbb{N}}\) and \((\epsilon_n)_{n \in \mathbb{N}}\) with \(|x - q_n| \leq \epsilon_n \to 0\). Let \(\mathbb{R}_c \subseteq \mathbb{R}\) denote the set of Cauchy–computable reals.

This is more sensible an idealization of scientific numerical computation; **validated** computing, to be precise \(\text{[loK]}\). Computability of some real number thus amounts to the ability to approximate it up to arbitrarily prescribable error. As a matter of fact, w.l.o.g. \(\epsilon_n = 2^{-n}\) in Definition 2.36 because otherwise we may proceed to the subsequence \(\tilde{q}_{n_k} := q_{k_n}\) with \(k_n := \min\{k : \epsilon_k \leq 2^{-n}\}\) computable from \((\epsilon_n)\).

It seems to have gone largely unnoticed that this notion of real number computability was already considered by ALAN TURING \([\text{Turi37}]\) and was the major motivation for introducing ‘his’ machine, nowadays usually employed only for the discrete realm. Its syntax thus remains unchanged, only the semantics are slightly modified by reducing computation on \(\mathbb{R}\) to an infinite sequence of (finite) calculations on \(\mathbb{Q}\). This idealizes (compare Section 1.2 and Chapter 2) the fact that we may abort the computation within any finite time after having reached the desired, arbitrarily prescribable accuracy \(\epsilon\) to, say, \(x = \sqrt{2}\) or \(x = \pi\). Infinity of computation corresponds to a real number carrying an infinite amount of information (cf. Example 2.25). As opposed to the BCSS model, it renders the computability even of single reals \(x \in \mathbb{R}\) a nontrivial property: there are uncountable many of them, yet the number of Turing machines is only countably infinite.

**Example 2.37.**

a) Every rational number is Cauchy–computable.

b) \(\pi\) is computable, e.g. based on the Bailey–Borwein–Plouffe formula \(\pi = \sum_{k=0}^{\infty} \frac{1}{16^k} \left( \frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right)\); and so is \(\sqrt{2}\) (Fact 2.12).

c) The number \(x_H := \sum_{n \in H} 2^{-n}\), where \(H \subseteq \mathbb{N}\) denotes the Halting Problem, is not Cauchy–computable but naively computable. \([\text{Spec49}]\).

Things become even more interesting when we consider real function computability in Section 2.4.2 below.

Apart from computability, Recursive Analysis also allows interesting complexity investigations \([\text{Mu86}]\ [\text{Ko91}]\). These are beyond the focus of the present treatment, however.

### 2.4.1 Binary Computability

The notion of real computability introduced, together with the Turing machine, in \([\text{Turi36}]\) amounts\(^9\) to

---

\(^9\) Turing actually employed decimal rather than binary expansion.
2.4. RECURSIVE ANALYSIS

Definition 2.38. A real number \( x \in (0,1) \) is **binarily computable** if \( x = \sum_{n \in A} 2^{-n} \) for some decidable \( A \subseteq \mathbb{N} \).

(Note that a binary expansion need not be unique.)

Now binary computability implies Cauchy computability even uniformly, by virtue of the sequence \( q_n := \sum_{i \in A} 2^{-i} \) with \( |x - q_n| \leq 2^{-n} \). The reverse conversion in general fails; for instance the approximation \( q_2 := \frac{1}{4} \) of \( x \) up to \( 2^{-2} \) does not permit any conclusions about any bit of \( x \)'s binary expansion: \((1.0000...)_2\) and \((0.1110...)_2\) are both possible. Moreover, as simple an operation as addition is Cauchy computable but not binarily computable:

**Example 2.39.** Given two rational sequences \( (q_n) \) and \( (p_n) \) with \( |x - q_n|, |y - p_n| \leq 2^{-n} \), \( (q_{n+1} + p_{n+1}) \) obviously satisfies \(|x + y) - (q_{n+1} + p_{n+1})| \leq 2^{-n}\).

Given two binary sequences \( (a_1, a_2, \ldots) \) and \( (b_1, b_2, \ldots) \) with \( x = \sum_n a_n 2^{-n} \) and \( y = \sum_n b_n 2^{-n} \), a Turing Machine is in general not able to calculate from these a binary sequence \( (c_1, c_2, \ldots) \) with \( x + y = \sum_n c_n 2^{-n} \).

(The proof is a standard discontinuity argument on Cantor space; cmp Fact 2.57.)

This deficiency of Definition 2.38 was pointed out in [Tur37] and led ALAN TURING to Definition 2.36.

Under the hypothesis that \( x \) be irrational (or just not a dyadic fraction), Cauchy approximations do allow conversion into \( x \)'s (now unique) binary expansion [Wei00, Theorem 4.1.13.1], while rational \( x \) is trivially binarily computable. We thus have

**Lemma 2.40.** A real number is Cauchy–computable iff it is binarily computable.

Binary computability becomes slightly more messy a notion when we look at semi-decidability:

**Example 2.41.** There are two real numbers \( a \) and \( b \) with semi-decidable binary expansions whose sum \( a + b \) has no semi-decidable binary expansion!

**Proof.** Let \( a := \sum_{n \in H} 4^{-n} \) and \( b := \sum_{n \in \mathbb{N}} 4^{-n} \); the first has semi-decidable binary expansion, the second even a decidable one. Their sum \( a + b = \sum_{n \in \mathbb{N}} c_n 4^{-n} \) has expansion coefficients \( c_n = (10)_2 \) if \( n \in H \) and \( c_n = (01)_2 \) if \( n \not\in H \), so their lower binary digits cannot be semi-decidable.

\( \square \)

2.4.2 Recursive Real Functions

Computing a single real number \( x \) involves output only, whereas a real function \( f \) is naturally considered as a transformation from input \( x \) to output \( y = f(x) \):

**Definition 2.42.** Call \( f: \mathbb{R} \to \mathbb{R} \)

- **a) effectively evaluable or \((\rho \to \rho)\)-computable** if some Turing machine can, upon input of \( x \) (by means of fast converging rational approximations \( q_n \) in the sense of Definition 2.36 with \( c_n = 2^{-n} \)), output (fast convergent rational approximations \( p_m \) to) \( y = f(x) \);
- **b) **Weierstraß–computable if some Turing machine can (just output) uniform approximations to \( f \), namely a sequence of (degrees and coefficients of) rational polynomials \( Q_n \in \mathbb{Q}[X] \) with the following effective version of Equation 2.3:

\[
\| f - Q_n \|_{[-n,n]} := \sup_{-n \leq x \leq n} |f(x) - Q_n(x)| \leq 2^{-n}
\]

- **c) see page 33**

**Example 2.43.** The following are effectively evaluable:

- **a) Addition** \( +: \mathbb{R}^2 \ni (x, y) \mapsto x + y \in \mathbb{R} \):

\[
|x - q_n| \leq 2^{-n} \quad \text{and} \quad |y - p_n| \leq 2^{-n} \quad \text{imply} \quad |(x + y) - (q_{n+1} + p_{n+1})| \leq 2^{-n}.
\]
b) Similarly negation \( x \mapsto -x \), multiplication \( \times : (x,y) \mapsto xy \), and inversion \( 0 \neq x \mapsto 1/x \);

c) (Multivariate) polynomials with computable coefficients;

d) \((x,y) \mapsto \max(x,y), \ x \mapsto |x|, \ (x_1, \ldots, x_n) \mapsto \sqrt{x_1^2 + \cdots + x_n^2} \);

e) Standard transcendental functions like \( x \mapsto e^x \) or \( 0 < x \mapsto \log(x) \);

f) Analytic \( x \mapsto \sum_n a_n x^n \) (on any closed ball contained in its disc of convergence), provided that \((a_n)\) is a computable real sequence;

g) Also non-analytic (but continuous) \( x \mapsto e^{-1/|x|} \).

Proof. See [Weih00 Section 4.3]. \( \square \)

Observe that \((Q_n)\) in Definition 2.42 converges to \( f \) uniformly (in terms of the sup norm) on any compact interval \([-n, +n] \subseteq \mathbb{R} \); hence the continuity of the polynomials carries over through the limit and requires that any Weierstraß–computable \( f \) be continuous. This turns out to be also a requirement implicit in Item a) and is sometimes called the Main Theorem of Recursive Analysis. Its proof, given below, amounts to an information theory discontinuity argument on the Baire space of infinite rational sequences; a significant generalization appears in Fact 2.62.

**Proposition 2.44 (Main Theorem for real functions).** Let \( f : [0,1] \to \mathbb{R} \) be effectively evaluable. Then \( f \) is continuous.

![Heaviside's function](image)

Figure 2.2: Heaviside’s function \( h \) and its flipped variant \( \overline{h} = 1 - h \)

**Proof.** First consider Heaviside’s function (cf. Figure 2.2)

\[
h : \mathbb{R} \to \mathbb{R}, \quad x \mapsto 0 \text{ for } x \leq 0, \quad x \mapsto 1 \text{ for } x > 0
\]

(2.7)

as a prototype of a discontinuous total real function. Feed into a hypothetical Turing machine \( \mathcal{M} \) evaluating \( f \) the rational sequence \( q_n := 2^{-n} \) as valid fast converging approximations for \( x = 0 \). By hypothesis it will then spit out a sequence \((p_m)_m \subseteq \mathbb{Q} \) with \( |p_m - y| \leq 2^{-m} \) for \( y = h(x) = 0 \); in particular, \( |p_2 - \mathring{y}| > 2^{-2} \) for \( \mathring{y} := 1 \). Up to output of \( p_2 \), \( \mathcal{M} \) has executed a finite number \( N \in \mathbb{N} \) of operations and in particular read at most the initial part \( p_0, p_1, \ldots, p_N \) of the input.

Now re-use \( \mathcal{M} \) in order to evaluate \( h \) at \( \mathring{x} := p_N > 0 \) given by the rational sequence \((\mathring{q}_n) := (q_0, q_1, \ldots, q_N, q_N, \ldots) \) coinciding with \((p_n)\) for \( n \leq N \). Being a deterministic machine, \( \mathcal{M} \) will then proceed exactly as before for its first \( N \) steps; in particular the output \((\mathring{p}_m)\) agrees with \((p_m)\) up to \( m = 2 \). Hence \( |\mathring{p}_2 - \mathring{y}| > 2^{-2} \) contradicting that \( \mathcal{M} \) is supposed to output fast converging approximations to \( \mathring{y} = h(\mathring{x}) \).

For the case of a general function \( f : \mathbb{R} \to \mathbb{R} \) with discontinuity at some \( x \in \mathbb{R} \), let \( y = f(x) \neq \lim_k f(x_k) = \mathring{y} \) with a real sequence \( x_k \) converging to \( x \). There exists \( M \in \mathbb{N} \) with \( |y - \mathring{y}| > 2^{-M+2} \); by proceeding to an appropriate subsequence of \((x_k)\), we may suppose w.l.o.g. that \( |x - x_k| \leq 2^{-k-2} \) and \( |f(x_k) - \mathring{y}| \leq 2^{-M} \). Then there is a rational double sequence \((q_{k,n})\) such that \( |x_k - q_{k,n}| \leq 2^{-n-1} \); thus \( |x - q_{n,n}| \leq 2^{-n} \). One can therefore feed rational approximations
(q_{n,n}) in order to evaluate f at x and obtain in turn approximations (p_m) \subseteq \mathbb{Q} to y. As before, p_M is output after only some finite initial part (q_{n,n})_{n \leq N} of the input has been read. Then
\[ |q_{n,n} - x_N| \leq |q_{n,n} - x_n| + |x_n - x| + |x - x_N| \leq 2^{-n-1} + 2^{-n-2} + 2^{-N-2} \leq 2^{-n} \]
for n \leq N reveals this very initial part to also be the start of a sequence of fast converging rational approximations to \( \tilde{x} := x_N \), whereas
\[ 2^{-M+2} < |y - \tilde{y}| \leq |y - p_M| + |p_M - f(\tilde{x})| + |f(\tilde{x}) - \tilde{y}| \leq 2^{-M} + |p_M - f(\tilde{x})| + 2^{-M} \]
shows that \((p_m)_{m \leq M}\) do not constitute approximations to \( f(\tilde{x}) \): contradiction. \( \square \)

Recall that a function \( f : X \to Y \) is continuous iff the pre-image \( f^{-1}[V] \subseteq X \) is open for every open \( V \subseteq Y \); and a set \( U \subseteq \mathbb{R} \) is open iff it is a countable union of open intervals with rational centers and radii
\[
U = \bigcup_{n \in \mathbb{N}} B(q_n, r_n), \quad q_n \in \mathbb{Q}, 0 \leq r_n \in \mathbb{Q}, \quad B(q, r) = \{x : |x - q| < r\}. \tag{2.8}
\]

This suggests a further notion of computability for continuous real functions based on the requirement that the mapping \( V \mapsto f^{-1}[V] \) from open sets to open sets be effective:

**Definition 2.42 (cont.)** Call \( f : [0, 1] \to \mathbb{R} \)

- c) **effectively continuous** if some Turing machine can, upon input of open \( V \) by means of a rational sequence \((\vec{q}_n, r_n)\) with \( V = \bigcup_n B(\vec{q}_n, r_n) \), output a rational sequence \((\vec{p}_m, s_m)\) with \( f^{-1}[V] = \bigcup_m B(\vec{p}_m, s_m) \).

Now the good news is that the above three notions of computable real functions have turned out to be equivalent:

**Fact 2.45.** Function \( f : [0, 1] \to \mathbb{R} \) is effectively evaluable iff it is Weierstraß–computable and iff it is effectively continuous. (The equivalence holds even uniformly in the sense that a Turing machine can convert back and forth between the three above ways of representing \( f \); cf. Section 2.4.6.)

This follows from the works of GRZEGORCZYK [Grze57], POUR-EL, CALDWELL, HAUCK [PECa75, Hauc76], and LACOMBE [Laco57, Laco58]. Equivalence to Definition 2.42) is known as Effective Weierstraß Theorem; see e.g. the textbook [PERi89, Section 0.7].

**Example 2.46.**

a) A real c is (Cauchy–) computable iff the constant function \( f(x) \equiv c \) is computable.

b) The characteristic function of \( \{0\} \) is not computable (i.e., the test \( \text{“} x = 0 \text{?”} \) is undecidable in Recursive Analysis).

Indeed, \( \chi_{\{0\}} : \mathbb{R} \to \{0, 1\} \) has a discontinuity and thus fails to be computable by virtue of the ‘Main Theorem’. Recursive Analysis is often criticized for this negative example; particularly by the BCSS community, but also by numerical analysts: to whom on the other hand it is gospel that numerical programs must not use tests for equality.

Section 1.3 of the present work addresses Question 1.22 whether hypercomputers may allow discontinuous real functions to be evaluated.

### 2.4.3 Semi–Computability and Semi–Decidability

Since \( H \) is undecidable, the real number \( x_H = \sum_{n \in \mathbb{N}} 2^{-n} \) lacks Cauchy–computability Example 2.37); on the other hand, the semi–decidability of \( H \) allows to computably approximate \( x_H \) from below; in fact to compute all rational lower approximations:
CHAPTER 2. REAL COMPUTATION

Definition 2.47. Call \( x \in \mathbb{R} \) lower computable if the Dedekind cut \( \mathbb{Q}_{<x} = \{ q \in \mathbb{Q} : q < x \} \) is semi-decidable. Semi-decidability of Dedekind cut \( \mathbb{Q}_{>x} = \{ q \in \mathbb{Q} : q > x \} \) means right computability of \( x \).

In the same way as recursivity amounts to both r.e. and co-r.e. (Section 1.1.1), a real number is Cauchy-computable iff it is both left and right computable. This equivalence in fact holds uniformly (Fact 2.61); here, the strictness of the Dedekind cuts \( \mathbb{Q}_{<x} \) and \( \mathbb{Q}_{>x} \) in Definition 2.47—i.e., are not complementary (in \( \mathbb{Q} \)) to each other for \( x \in \mathbb{Q} \)—enters crucially [Weih00, Example 4.1.14.2].

A different notion of real semi-computability, related to classical semi-decidability, is exemplified as follows: Although the equality of real numbers is \( \rho \)-undecidable due to the Main Theorem, inequality is at least semi-decidable.

Observation 2.48. There exists a Turing machine which, given a rational sequence \((q_n)_{n} \) with \(|x - q_n| \leq 2^{-n}\), terminates iff \( x \neq 0 \): simply search for some \( n \in \mathbb{N} \) with \( |q_n| > 2^{-n} \).

2.4.4 Enumerable Real Open Sets

As pointed out at the end of Section 2.4.2, the concept of decidability of real subsets makes no sense in Recursive Analysis. That of semi-decidability, however, does lead to a natural notion, nicely related to Definitions 2.47 and 2.42a+c):

Fact 2.49. Call \( U \subseteq \mathbb{R} \) semi-decidable if some Turing machine can, upon input of \( x \in \mathbb{R} \) by means of \((q_n)_{n} \subseteq \mathbb{Q} \) with \(|x - q_n| \leq 2^{-n}\), terminate in the case of \( x \in U \) and diverge in the case of \( x \notin U \). Then every semi-decidable set is necessarily open. More precisely, for \( U \subseteq \mathbb{R} \) the following are equivalent:

i) \( U \) is semi-decidable.

ii) Some Turing machine can enumerate rational centers \( q_n \) and radii \( r_n \geq 0 \) of open intervals exhausting \( U \) according to Equation (2.8).

iii) The characteristic function \( \chi_U : \mathbb{R} \to \{0, 1\} \) of \( U \) is effectively left-evaluable (also called \((\rho \rightarrow \rho_{<})\)-computable) in the sense that a Turing machine can, upon input of \((q_n)_{n} \) for \( x \in \mathbb{R} \) as above, output the Dedekind cut \( \mathbb{Q}_{<f(x)} \).

Proof. See e.g. [Weih00]. \( \square \)

In view of the classical equivalence between semi-decidability and recursive enumerability (Section [1.1.1], it is thus justified to denote, as in Recursive Analysis, a set \( U \subseteq \mathbb{R} \) as r.e. open if it satisfies one (and thus all) of the above Conditions i) to iii).

We quote from [Zhon98, Proposition 3.1] an illustrative Example 2.50.

Example 2.50. The complement of Cantor’s Set (recall Example 2.8) is r.e. open: it consists of the (not necessarily disjoint) union of all open rational intervals of the form \((0.t_1t_2 \ldots t_n1)_3, (0.t_1t_2 \ldots t_n2)_3\) with ternary digits \( t_1, \ldots, t_n \in \{0, 1, 2\} \) and \( n \in \mathbb{N} \); these intervals are obviously enumerable.

Another connection of Fact 2.49i) to classical semi-decidability, similar to Observation 2.48 is given in the following

Fact 2.51. There exists a Turing machine which, given a sequence of (rational centers and radii of) open rational balls \((B_n)_{n} \), terminates iff \( \bigcup_n B_n \supseteq [0, 1] \).

The proof exploits Heine-Borel compactness of \([0, 1]\) in the obvious way, recall Section 2.2.2.
2.4. RECURSIVE ANALYSIS

2.4.5 Gain and Vain of Non-Uniformity

Recall Section 1.1.1: A problem $R$ is decidable iff membership “$x \in R$” can be decided by some Turing machine; more precisely: iff there exists a machine $M$ which, for all $x$, answers yes whenever $x \in R$ and no whenever $x \notin R$.

Example 2.52. The famous “$\mathcal{P} = \mathcal{NP}$?” problem is decidable:
Let $M_1$ always output yes, $M_0$ always no; either one or the other is correct.

Of course this ‘solution’ to a Millennium Prize Problem is unlikely to earn $1,000,000 from Mr. Landon T. Clay. We have seen a similar situation within real number computability in Lemma 2.40: for every two Turing machines $M_x$ and $M_y$ deciding respective binary expansions of $x, y \in \mathbb{R}$, there exists (!) a machine $M_{x+y}$ deciding the expansion of $x + y$ yet (a code of) $M_{x+y}$ can be shown to be uncomputable from (codes of) $M_x$ and $M_y$. This kind of situation is referred to as non-uniform, whereas the Cauchy–computability of addition holds uniformly, see Example 2.39.

Another simile comes from classical mathematics, where a function $f : \mathbb{R} \to \mathbb{R}$ is continuous iff, for every $x, \epsilon$ there exists $\delta$ such that $|x - y| \leq |\delta|$ implies $|f(x) - f(y)| \leq \epsilon$; whereas uniform continuity means that for every $\epsilon$ there exists $\delta$ such that, for all $x$, $|x - y| \leq |\delta|$ implies $|f(x) - f(y)| \leq \epsilon$. The difference between uniformity and non-uniformity thus corresponds to swapping quantifiers $\exists \forall$ to $\forall \exists$.

The issue of uniform computability also arises when we consider a sequence $(x_n)_n$ of real numbers: When should it be considered effective? Requiring that each member $x_n$ is computable leaves the possibility that there is a corresponding, separate Turing machine $M_n$ for each $n$. The following uniform notion thus makes more sense:

Definition 2.53. A sequence $(x_n)_n$ is Cauchy–computable if there exists a Turing machine to generate a rational double sequence $(q_{n,m})_{n,m}$ with $|x_m - q_{n,m}| \leq 2^{-n}\forall m$.

We amplify this illustration of non-/uniformity with the more elaborate example showing that

Rational Zeros of Real Polynomials are Decidable

Equality “$x = 0$” of rational numbers $x = p/q$, given by (e.g. the respective binary expansions of) numerator $p$ and denominator $q$, is of course decidable. The equality of real numbers on the other hand is undecidable (recall Example 2.46b) yet nonuniformly trivially decidable. Between these two extremes lies

Problem 2.54. For $p \in \mathbb{R}[X_1, \ldots, X_n]$ and $\bar{q} \in \mathbb{Q}^n$, decide whether $p(\bar{q}) = 0$.

When $p$ is part of the input (its coefficients given as above by rational approximations), this includes the test for equality of reals and is thus undecidable. When $p$ is on the other hand arbitrary (its coefficients not even necessarily computable) but fixed, Problem 2.54 becomes decidable:

Theorem 2.55. a) For each $p \in \mathbb{R}[X_1, \ldots, X_n]$, the set $\{\bar{q} \in \mathbb{Q}^n : p(\bar{q}) = 0\}$ is decidable.

b) For each $p \in \mathbb{R}_c[X_1, \ldots, X_n]$, the function $\text{sgn}_c : \mathbb{Q}^n \to \mathbb{Q}$, $p(x) \mapsto \text{sgn}(p(x))$ is computable.

c) Let $S \subseteq \mathbb{R}^n$ be semi-algebraic over $\mathbb{R}$. Then $S \cap \mathbb{Q}^n$ is decidable in $\mathbb{Q}^n$.

Proof. a) Let

$$p(\bar{x}) = \sum_{k_1=0}^{d_1} \cdots \sum_{k_n=0}^{d_n} a_{(k_1, \ldots, k_n)} \cdot x_1^{k_1} \cdots x_n^{k_n}, \quad a_{(k_1, \ldots, k_n)} \in \mathbb{R} \quad \forall k_i = 0, \ldots, d_i, \; i = 1, \ldots, n.$$

Take a basis $\{b_1, \ldots, b_m\}$ for the finite-dimensional $\mathbb{Q}$–vector space

$$V := \{q_{(0, \ldots, 0)}a_{(0, \ldots, 0)} + \cdots + q_{k-\bar{k}}a_{(k_1, \ldots, k_n)} + \cdots + q_{(d_1, \ldots, d_n)}a_{(d_1, \ldots, d_n)} : q_{\bar{k}} \in \mathbb{Q}\}.$$
Each coefficient of \( p \) is thus of the form \( a_k = \sum_{i=1}^{m} A_{i,k} b_i \) with unique \( A_{i,k} \in \mathbb{Q} \). Now for given \( x \in \mathbb{Q}^n \),

\[
0 = p(x) = \sum_{i=1}^{m} b_i \cdot \sum_{k_1=0}^{d_i} \ldots \sum_{k_n=0}^{d_n} A_{i,k} \cdot x_1^{k_1} \ldots x_n^{k_n} =: R_i(x) \in \mathbb{Q}
\]

holds if and only if \( R_i(x) = 0 \) for all \( i = 1, \ldots, m \) because the \( b_i \) are linearly independent over \( \mathbb{Q} \). The equalities \( R_i(x) = 0 \) in turn are of course decidable by means of exact rational arithmetic.

b) Since the coefficients of \( p \) are computable, one can effectively evaluate it on \( x \) and then apply Observation \ref{obs:2.48} to the result \( p(x) \) to simultaneously semi-decide both cases “\( p(x) < 0 \)” and “\( p(x) > 0 \)”.

The third case “\( p(x) = 0 \)” is decidable by a).

c) \( S \) is the set of solutions \( x \) to some finite system of polynomial (in)equalities \( p_i(x) = 0 \) and \( q_j(x) > 0 \) with \( p_i, q_j \in \mathbb{R}[x_1, \ldots, x_n] \). For \( x \in \mathbb{Q}^n \), these are decidable according to b). \( \square \)

The non-uniformity implicit in this result consists in the arbitrary but fixed polynomial (system) and, more specifically, in \textit{finding}, for its coefficients, a \( \mathbb{Q} \)-basis and decomposition therein. Not only Turing machines fail at this task: a \( \mathbb{Q} \)-basis for the seemingly simple (since at most 3-dimensional) vector space spanned by \( (1, e \cdot \pi, e + \pi) \) is unknown to date \cite[pp.153]{EHH91}.

### 2.4.6 Type-2 Theory of Effectivity

Recursive Analysis used to suffer from a vast variety of notions, some subtly distinct, some different but equivalent. For instance consider the Definitions of a computable real number \ref{def:2.36} (with its two variants: \( \epsilon_n \) arbitrary or \( \epsilon_n = 2^{-n} \)) \ref{def:2.35} with alternative Dedekind cuts \( \mathbb{Q}_{\leq x}, \mathbb{Q}_{< x}, \mathbb{Q}_{\geq x}, \mathbb{Q}_{> x} \): Which of them are equivalent? uniformly or non-uniformly?

It was the achievement of Weihrauch’s \textit{Type-2 Theory of Effectivity} (TTE) to have formalized these notions by means of so-called representations; see \cite{Weih95, Weih00}. Within this framework, computability notions like the above ones have been systematically compared and the results formulated concisely both for the uniform and the non-uniform case \cite[Section 4.1]{Weih00}. In order to prevent misconceptions, I should emphasize that TTE does not constitute a new model of computation but a framework for and (if you like, \textit{meta}-) theory of (both previous and new) notions of computability. It extends, from countable universes to those of continuum cardinality, the (classically usually implicit) requirement that objects (such as graphs or integers) need to be encoded into strings over a finite alphabet (e.g. \( \{0,1\} \)).

**Definition 2.56.** Let \( A \) be an arbitrary set.

i) A \textit{representation} \( \alpha \) of \( A \) is a surjective partial mapping \( \alpha : \subseteq \{0,1\}^\omega \to A \) from the Cantor space of infinite strings onto \( A \).

ii) An infinite string \( \bar{\sigma} \) with \( \alpha(\bar{\sigma}) = a \) is called an \textit{\alpha-name} of \( a \). We say that \( a \) is \( \alpha \)-computable if it has a \( \alpha \)-name.

iii) For an additional set \( B \) with representation \( \beta \), a partial function \( f : \subseteq A \to B \) is called \( (\alpha \to \beta) \)-\textit{computable} if a Turing machine can convert every \( \alpha \)-name of every element \( a \in A \) to some \( \beta \)-name of \( b = f(a) \). \( f \) is \( (\alpha \to \beta) \)-\textit{continuous} if there exists a (Cantor-) continuous function \( F : \subseteq \{0,1\}^\omega \to \{0,1\} \) such that \( \beta(F(\bar{\sigma})) = f(\alpha(\bar{\sigma})) \) for all \( \bar{\sigma} \in \text{dom}(f \circ \alpha) \). Such \( F \) constitutes an \( (\alpha \to \beta) \)-\textit{realization of} \( f \).

iv) For two representations \( \alpha \) and \( \beta \) of the same set \( X \), \( \alpha \) is \textit{reducible} to \( \beta \), written as “\( \alpha \preceq \beta \)”, if the identity function on \( X \) is \( (\alpha \to \beta) \)-computable. \textit{Equivalence} “\( \alpha \equiv \beta \)” of \( \alpha \) and \( \beta \) means that in addition \( \beta \) is conversely reducible to \( \alpha \). We write “\( \alpha \preceq_1 \beta \)” if the identity on \( X \) is \( (\alpha \to \beta) \)-continuous.
2.4. RECURSIVE ANALYSIS

Whereas classical (i.e. Type-1) computability deals with objects from a countable universe (e.g. \(\{0,1\}^*, \mathbb{N}, \text{or} \mathbb{Q}\) consisting of (or encoded in) finite strings, the Type-2 computability of objects \(a\) in some uncountable universe \(A\) is thus defined in terms of infinite strings as names of \(a\); and computability of a function amounts to effective conversion of names of arguments to names of values, that is to the commutativity of the following diagram with some realization \(F : \subseteq \{0,1\}^\omega \to \{0,1\}^\omega\) as a computable function on Cantor space:

\[
\begin{array}{c}
\sigma \in \{0,1\}^\omega \\
\downarrow \alpha \\
\bar{\sigma} \in \{0,1\}^\omega \\
\beta \\
\downarrow \gamma \\
\nu \in A \\
\downarrow f \\
b \in B
\end{array}
\] (2.9)

Note that the Turing machine in Definition 2.56iii) may behave arbitrarily both on inputs \(\bar{\sigma} \notin \text{dom}(\alpha)\) and for \(\alpha(\bar{\sigma}) \notin \text{dom}(f)\). The **Main Theorem now takes the form of**

**Fact 2.57 (Main Theorem for functions on Cantor Space).** Let \(F : \subseteq \{0,1\}^\omega \to \{0,1\}^\omega\) be computable. Then \(F\) is continuous (w.r.t. Cantor topology).

A simple proof can be found in [Weih00 Theorem 2.2.3]. In particular, \((\alpha \to \beta)\)-computability requires \((\alpha \to \beta)\)-continuity!

Items b), c), and d) below are TTE's rephrasing of Definitions 2.36, 2.47, 2.35, respectively.

**Example 2.58.**

a) The **standard representation** \(\nu : \subseteq \{0,1\}^\omega \to \mathbb{Q}\) has domain \(\text{dom}(\nu) = \{0,1\}^\omega \cap \{p \cdot 0^0 : p, q \in \mathbb{N}\} \circ \{0,1\}^\omega\); it assigns to \(\bar{\sigma} = 01^p 01^q 0 \cdots\) the number \(\nu(\bar{\sigma}) = +p/q\), to \(\bar{\sigma} = 11^p 01^q 0 \cdots\) the number \(\nu(\bar{\sigma}) = −p/q\).

b) The **Cauchy representation** \(\rho : \subseteq \{0,1\}^\omega \to \mathbb{R}\) is the mapping, from sequences \((q_n)\) of rationals (encoded and delimited as in a) with \(|q_n – x| \leq 2^{-n}\) for some \(x \in \mathbb{R}\), to this very \(x\).


c) The **left representation** \(\rho_\leq : \subseteq \{0,1\}^\omega \to \mathbb{R}\) is the mapping from (appropriately encoded) enumerations of the Dedekind cuts \(\mathbb{Q}_{<x}\) to \(x \in \mathbb{R}\); the same applies appropriately for the right representation \(\rho_\geq\).

d) The **naive representation** \(\rho_{\mathcal{CN}}\) assigns to (an encoding of) a convergent rational sequence \((q_n)\) its limit \(x = \lim_n q_n\).

e) The **Weierstraß representation** \([\rho = \rho]_{\mathcal{C}(\mathbb{R})}\) assigns, to (an encoding of the degrees and coefficients of) a sequence \((Q_n)\) of polynomials over \(\mathbb{Q}\) with \(\|Q_n – f\|_{|n, +n]} \leq 2^{-n}\) for some \(f \in \mathcal{C}(\mathbb{R})\), this very \(f\).

f) Fix \(d \in \mathbb{N}\). Let \(\theta_{<}\) denote the following (inner) representation of the class \(\mathcal{O}^d\) of open subsets of \(\mathbb{R}^d\): a name of \(U \in \mathcal{O}^d\) is (an encoding of) a list of rational centers \(\bar{q}_n\) and radii \(r_n\) of open Euclidean balls with \(U = \bigcup_n B(\bar{q}_n, r_n)\).

The representation in Item e) corresponds of course to Definition 2.42); its surjectivity relies on the Weierstraß Approximation Theorem, recall Fact 2.9. Item f) corresponds to Fact 2.49i) and permits Definition 2.42c) to be rephrased as \((\theta_{<} \to \theta_{<})\)-computability of the pre-image mapping \(V \mapsto f^{-1}[V]\) on open subsets of \(\mathbb{R}\).

**Remark 2.59.** It is central to the concept of TTE (Section 2.4.6) that a representation \(\alpha : \subseteq \{0,1\}^\omega \to A\) induces a notion of effectivity on all elements of the represented space \(A\): both computable and uncomputable ones are assigned names and can be input to a Type-2 Machine computing some function \(f : A \to B\).

It may at first seem unnatural to define effective function evaluation \(f : x \mapsto f(x)\) even on uncomputable inputs \(x\). However, this requirement avoids many pathologies (unlike, for instance, Markov-computability; see Section 2.6.3 below) and is crucial to the equivalence of effective evaluation to several other natural notions of function computability (recall Fact 2.45).
Constructing New Representations from Given Ones

can often proceed canonically and categorically [Weih00, Section 3.3]. We will mention three important examples:

**Fact 2.60.** Let index set \( I \) be either \( \mathbb{N} \) or finite.

a) If \( \alpha_i \) is a representation for \( A_i, i \in I \), then there is a natural representation \( \prod_{i \in I} \alpha_i \) for the Cartesian product \( \prod_{i \in I} A_i \).

b) For each \( i \in I \), let \( \alpha_i \) be a representation for (the same) \( A \). There is a representation \( \bigwedge_{i \in I} \alpha_i \) of \( A \) (which roughly incorporates the information provided by all \( \alpha_i \)).

c) Let \( \alpha \) be a representation for \( A \) and \( \beta \) one for \( B \). There is a natural representation \( [\alpha] \to \beta \) for the class of \( (\alpha \to \beta) \)–continuous functions \( f : A \to B \) (basically encoding an \( (\alpha \to \beta) \)-realization \( F : \subseteq \{0,1\}^\omega \to \{0,1\}^\omega \); see [Weih00, Definition 3.3.13].

d) For the case \( A = B = \mathbb{R} \) and \( \alpha = \beta = \rho \), the representation from c) is equivalent to the one in Example 2.58e) [Weih00, Section 6.1].

e) The representation from c) satisfies SMN and UTM–like properties, even uniformly [Weih00, Theorem 3.3.15].

A real sequence \((x_m)_m\) for instance is computable in the sense of Definition 2.53 iff it is computable with respect to the representation \( \prod_m \rho \) according to Fact 2.60a); which in turn equivalent to \([\nu]^{\mathbb{N}} \to \rho \) according to Fact 2.60c) where \( \nu \) denotes the restriction (in image) to \( \mathbb{N} \) of the representation \( \nu : \subseteq \{0,1\}^\omega \to \mathbb{Q} \) from Example 2.58h) [Weih00, Lemma 3.3.16].

Fact 2.60d) is the uniform version of Fact 2.45. Analogously, the uniform equivalence of Cauchy-computability to simultaneous left and right computability mentioned in Section 2.4.3 can now be formulated (and easily proved) as

**Fact 2.61.** It holds \( \rho \equiv \rho_\prec \land \rho_\succ \).

Every representation \( \alpha : \subseteq \{0,1\}^\omega \to A \) induces on \( A \) a topology inherited from Cantor space. If this final topology complies with the original one in the sense of [Weih00, Definition 3.2.7], \( \alpha \) is called admissible. For instance, \( \rho \) is admissible with the usual Euclidean topology on \( \mathbb{R} \) [Weih00, Lemma 4.1.4]. The following consequence of Fact 2.57 thus generalizes Proposition 2.44 and yields a further version of the Main Theorem:

**Fact 2.62 (Main Theorem of TTE).** Let \( \alpha \) and \( \beta \) be admissible representations of \( A \) and \( B \), respectively. Then every \( (\alpha \to \beta) \)-computable function \( f : \subseteq A \to B \) is \( (\alpha \to \beta) \)-continuous and in particular continuous.

**Proof.** See [Weih00, Theorem 3.2.11].

**Multivalued Functions**

Many mathematical proofs involve ‘choosing’ an arbitrary member from a set asserted to be non-empty and continuing arguing about that element. The equivalence of \( \varepsilon, \delta \)-continuity with sequential continuity for real functions, for instance, exploits the Archimedean Property of \( \mathbb{R} \):

**Example 2.63.**

a) To any given real \( x \), there exists an integer \( N \) with \( N > |x| \).

b) Every mapping \( f : \mathbb{R} \to \mathbb{N} \) with \( f(x) > |x| \) is discontinuous and thus uncomputable by the Main Theorem.

c) Given \((q_n) \subseteq \mathbb{Q} \) with \( |x - q_n| \leq 2^{-n} \), \( N := [\|q_0\|] + 3 \) is easily computable and satisfies \( N > |x| \).
Item c) exploits the arbitrariness of a choice \( N \) with \( N > |x| \). This amounts to replacing (single-valued) \( f \) by a multi-valued function \( \bar{f} : \mathbb{R} \rightarrow \mathbb{N} \) with \( \bar{f}(x) \subseteq \{ N \in \mathbb{N} : |x| < N \} \), that is a relation \( R_{\bar{f}} = \{(x,N) : x \in \mathbb{R}, N \in \mathbb{N}, |x| < N \} \). More generally, relaxing \((\alpha \rightarrow \beta)\)-computability from \( g \subseteq A \rightarrow B \) to \( \bar{g} \subseteq A \Rightarrow B \) in the sense of [Weih00, Definition 3.1.3] means that not only the \( \beta \)-name output for \( b = g(a) \) but also the computed value \( b \in \bar{g}(a) \) itself may depend on which \( \alpha \)-name for \( a \) the Turing machine is given.

### 2.5 BCSS versus/with Recursive Analysis

Choosing an appropriate model of computation (recall Section 1.2) is generally a difficult and hotly disputed task. In the discrete realm (Type-1 setting: bits, integers, …) the Turing machine has become widely accepted, a particular argument in its favor being its equivalence to several other seemingly unrelated reasonable notions of computability (Section 1.4, Item c). Over the reals, however, it holds

**Observation 2.64.** BCSS machines and Recursive Analysis have noncomparable computational power:

a) Heaviside’s function is computable by the first but, in spite of its simplicity, not to the latter (see Proposition 2.44).

b) The equally simple exponential function on the other hand behaves in the opposite way, cf. Example 2.18.

(Regarding the semi-decidability of open sets, however, a comparison has succeeded in terms of degree theory, see Section 6.3.)

The question of which of the two models is ‘the’ appropriate one is a constant source of disputes and (meta-) arguments [Koep01, Brat00]. Other approaches try to combine both models and are described in the sequel. \( \mathbb{R} \)-analytic machines (Section 2.5.3), for instance, can be regarded as a synthesis of Recursive Analysis with BCSS-computations. But let us first discuss

#### 2.5.1 Variations of BCSS Machines

Every model of computation reflects some aspects of reality better and falls down on others due to the inherent idealizations. Some major criticisms of the BCSS machine concern

a) their use of arbitrary (although only finitely many) real constants;

b) exponentially long (in terms of bits) integers like \( 2^{2^n} \) obtainable within linearly many steps \( O(n) \) by repeated squaring;

c) the ability to perform all operations and in particular comparisons exactly;

d) the uncomputability of elementary functions like square root, exponentiation, logarithm, sine, and cosine; compare Example 2.18d).

We have investigated the impact of a) in Section 2.3.4. In order to counter Criticism d), we may enhance the BCSS machine’s set of operations to also include square root, exponentiation, or sine. This leads to hypercomputation and is discussed in Section 5.1. Conversely, Criticism b) can be circumvented by excluding multiplication and division as operational primitives. This leads to linear BCSS machines [Koir94, ChuKo95, FoKo90], compare Section 5.3.2. Criticism c), too, can be avoided: e.g. by including numerical condition numbers in the analysis [CuSm99]—or by reconciling BCSS Theory with Recursive Analysis, for instance by disallowing tests for equality and by changing the semantics of inequality tests [BrHe98]. This naturally gives rise to a mathematically elegant synthesis of algebraic complexity theory with real computability theory [25]. Another approach to obtaining a more realistic algebraic complexity (not computability) theory charges, to a BCSS machine’s operations, costs depending on the size of their operands [Koi97a, MadHW97, Brav05].
2.5.2 Aspects of Classification

We may roughly classify a model of real computation according to whether

- an admissible calculation takes finitely or infinitely many steps;
- input is presented exactly or by means of approximations;
- the entities under consideration (i.e., real or rational numbers) can be handled exactly or approximately
- and subjected to which kinds of operations and constants;
- the result is to be obtained exactly or approximately.

Notions of Approximation

In the case of approximate computations, one may furthermore distinguish

- certified (i.e. with error bounds) versus ultimate (i.e. in the sense of eventually tending towards the limit) precision;
- absolute ($\Delta_z = |z - z^*|$) versus relative ($\delta_z = \frac{|z - z^*|}{|z|}$) errors,
- being forward (i.e. referring to error of $y = f(x)$) or backward bounded/stable (i.e. referring to deviating $x$).

whereas computability in Recursive Analysis means infinitely long calculations on rational numbers approximating input and output with arbitrary classically-recursive operations and guaranteed absolute error bounds in the sense of forward stability. (This gives another way of looking at the ‘Main Theorem’: For a discontinuous function, the absolute forward error $\Delta_y$ is unbounded in terms of the input error $\Delta_x$.) Uncertified approximation occurs in naive computations, recall Definition 2.35.

Remark 2.65. Most models of approximate real computation refer to absolute error: for the simple reason that the difference $x - y$ of two positive numbers $x, y$ may have relative error unbounded in that of the input $x, y$. Determining the result up to arbitrarily prescribable relative error may thus be desirable in practice but is in general infeasible. Thus in order for this notion to make sense one has to resort to input given exactly, that is without any (relative or absolute) error.

2.5.3 Analytic Machines

Combining the items of the above classification results in a plethora of notions of real computation. Not all of them make sense (e.g. Remark 2.65) but most which do appear in the family of Analytic Machines proposed by Chadzerek and Hotz, introduced and compared in [HVS95, ChHo99].

In the sense of the above Classification 2.5.2 they name such a machine (or rather the function $f : x \mapsto y = f(x)$ it computes) according to whether

- its primitive operations and comparisons apply to rational numbers (“$\mathbb{Q}$”) or to real ones (“$\mathbb{R}$”);
- the result $y$ appears after finitely many steps (“computable”) or is the limit of an infinite computation (“analytic”);
- in the latter case: is input $x \in \mathbb{R}$ accessed directly and exactly or by means of a (specific) rounding function
  \[ \hat{\rho} : \mathbb{R} \times \mathbb{N} \to \mathbb{Q}, \quad |x - \hat{\rho}(x, n)| < 2^{-n} \]
as rational approximations of prescribable precision (“$\delta$–$\mathbb{Q}$” machine).
2.6. FURTHER MODELS

- If the result \( y \) of a computation \( x \mapsto f(x) = y \) is independent of the rounding function employed, the adjective “robust” is affixed to it.

- Are the (not necessarily rational) output approximations \( (p_n) \) to \( y \) accompanied by error bounds \( |y - p_n| \leq \varepsilon_n \to 0 \)? In this case call the machine “strongly analytic”.

- Suppose a computation provides error bounds \( \varepsilon_n \to 0 \) but violates \( |y - p_n| \leq \varepsilon_n \) an (arbitrary yet) finite number of times: this case is named “quasi-strongly analytic”.

Figure 2.3, taken from [ChHo99], illustrates weaker/stronger relations among certain members of this family and indicates which ones satisfy closure under composition. Thus, exactly the BCSS–

![Diagram](image)

...computable functions are “\( \mathbb{R} \)–computable” and “\( \mathbb{Q} \)–computability” is equivalent to classical (Type-1) computability. The class of \( (\rho \to \rho) \)–computable functions considered in Recursive Analysis coincides with the robust strongly \( \delta \)–\( \mathbb{Q} \)–analytic ones (which are in particular continuous); whereas \( (\rho \to \rho_{Cn}) \)–computability amounts to robust \( \delta \)–\( \mathbb{Q} \)–analyticity and includes some discontinuous functions but lacks closure under composition.

Even stronger a model, an \( \mathbb{R} \)–analytic machine, receives exact real number inputs yet has to only approximate the output. Apart from the practically unattainable power and structural lack of closure under composition, it does reflect how numerical analysis treat, for instance, integrals or nonlinear equations. It may therefore come as quite a surprise that even this machine provably cannot decide the stability of a dynamical system [ChHo99, SECTION 3.2]; more precisely it cannot decide, for a given initial value \( \vec{x}(0) \), whether the solution \( \vec{x}(t) \) of a fixed ordinary differential equation (ODE) in two dimensions converges as \( t \to \infty \). This is related to and generalizes previous results on discrete dynamical systems and Turing machines [Moor90, Delv06].

2.6 Further Models

The BCSS model (Section 2.3) and Recursive Analysis (Section 2.4) are the two major notions of computability for real number problems and the primary focus of the present work. Their
synthesis (exact real arithmetic/tests and approximate output computation, cf. the classification in Section 2.5.2) naturally leads to certain Analytic Machines dealt with in Section 2.5.3. However, for sake of completeness, we will now also briefly survey some other notions of real computation.

2.6.1 Domain Theory

dates back to Dana Scott [Scot70, Scot72]. It embeds real numbers $\mathbb{R}$ in the domain $\mathbb{I} \mathbb{R}$ of closed real intervals; and continuous real (partial) functions $f : \subseteq \mathbb{R} \to \mathbb{R}$ in order–continuous partial functions $F : \subseteq \mathbb{I} \mathbb{R} \to \mathbb{I} \mathbb{R}$. This leads to another formalization of interval computation very similar to Cauchy–computation, with two subtle differences:

First, evaluating a real function $f : \mathbb{R} \to \mathbb{R}$ in the sense of Definitions 2.36 and 2.42 corresponds to a computable transformation $((q_n), (\epsilon_n)) \mapsto ((p_m), (\delta_m))$ on rational sequences. An output interval $(p_m - \delta_m, p_m + \delta_m)$ is thus allowed to depend on several input intervals $q_n, n \leq N$, where $N$ may be computed adaptively, that is in turn depending on $m$ and the values $q_n$. However, $F$ (restricted to the effective basis $\mathcal{B}$ of all rational intervals) has to assign a value to every single rational input interval. An involved compactness argument [Weih00, Theorem 9.5.2] shows that in principle this additional constraint can indeed always be satisfied. In practice, however, it renders implementing a Domain algorithm technically more involved than doing so within the setting of Recursive Analysis.

Regarding partial real functions, the second difference makes Domain Computation truly more restrictive than Recursive Analysis: When a partial $f$ is evaluated on an argument $x \notin \text{dom}(f)$, the latter allows the Turing machine to behave arbitrarily; whereas the former requires it to diverge (strong computation). For total functions, however, the two are equivalent [Weih00, Theorem 9.5.2].

2.6.2 Constructive Mathematics

Constructive mathematicians interpret quantifiers and connectives differently from classical mathematics, e.g. in logical formulas like

$$\forall x \forall \epsilon > 0 \exists \delta > 0 \forall y : (|x - y| \leq \delta \Rightarrow |f(x) - f(y)| \leq \epsilon).$$

Such use of the same words (‘continuity’) but with different semantics of course tends to cause misunderstandings; compare the titles of [BiBr85, Weih00] etc. (The term Church’s Thesis also has a different meaning.) Moreover, there is not just one constructive mathematics but an entire bunch of flavors [Brid04] adding to an almost Babylonian confusion of language: e.g. Intuitionistic [Heyt71], Bishop–style [BiBr85], Markov–style (see Section 2.6.3) and, more recently, Reverse Mathematics [Ishi05, Ishi06].

Their common underlying idea, however, is simple and indeed quite convincing: A claim “$\exists y : P(y)$” that some object $y$ with property $P$ exists should be proven by constructing such an object, rather than indirectly by raising a contradiction from its non-existence. (The latter is denoted by “$\neg\neg\exists y : P(y)$” and, lacking excluded middle, carefully distinguished from the former.) And “$\forall x \exists y : P(x, y)$” means that $y$ is to be constructed from (and thus uniformly in) $x$; cf. Section 2.4.5.

Constructive versus Recursive Analysis

‘Constructing $y$’ roughly amounts to specifying an algorithm ‘computing $y$’—thus the title of the text book [BiBr85]—and the relations between Constructive and Recursive Analysis are indeed quite close, with many proofs and theorems in the first having carried over to the second [Liet04]. It is the converse transfer direction which may cause problems:

Weak König’s Lemma, for instance, is considered non–constructive and easily seen to fail computably, yet has been proven equivalent to WEIERSTRASS’ Approximation Theorem Fact 2.9 [Simp99, Theorem IV.2.5] which in turn does hold computationally [PERi89, Section 0.7]!

Among the causes of such differences, we mention
2.6. FURTHER MODELS

• diverging purposes:
  A major goal of (in particular Reverse) Constructive Mathematics is the restriction and
  identification of minimal choices of logical axioms in order to prove certain claims.

• the range of objects under consideration:
  In Recursive Analysis, a real function (and evaluation thereof; cf. Section 2.4.2) is also
  defined for non-computable arguments \( x \).

For instance, constructive mathematicians say that a continuous function \( f \) on \( [0,1] \) need not
attain its supremum; whereas in Recursive Analysis, it is attained (even with a computable value \( y = \sup f \)) although on a possibly non-computable argument \( x \). For a more thorough comparison
between Recursive and (various flavors of) Constructive Mathematics, we recommend [Bees85, Chapter III].

As a consequence of the two items above, an algorithm in Recursive Analysis, although gen-
erally yielding a construction of some object \( x \), is typically shown to be correct on the basis of
classical logic. Such a proof may in particular refer to \( x \) (which constructively amounts to a
circular argument) in order to assert its computability.

Example 2.66. A function \( f : \mathbb{R}^n \rightarrow \mathbb{R}^m \) is called open iff

\[
\forall \bar{x} \forall \epsilon > 0 \exists \delta > 0 : f \left[ B(\bar{x}, \epsilon) \right] \supseteq B(f(\bar{x}), \delta)
\]

Computable openness of course means that \( \delta \) be computable from \( x \) and \( \epsilon \). The assertion that
the calculated \( \delta \) indeed satisfies “\( f \left[ B(\bar{x}, \epsilon) \right] \supseteq B(f(\bar{x}), \delta) \)” however may be shown classically [27, Example 34b]); whereas constructively, the latter inclusion amounts to

\[
\forall \bar{v} \in B(f(\bar{x}), \delta) \exists \bar{u} \in B(\bar{x}, \epsilon) : \bar{v} = f(\bar{u})
\]

which requires that, in addition to \( \delta \), \( \bar{u} \) also be ‘computed’.

More generally, formula involving iterated alternating quantifiers “\( \forall x \exists y \exists u \exists v : P(x, y, u, v) \)” , inter-
preted constructively, require both \( y \) and \( v \) to be constructed; whereas in Recursive Analysis it
may often suffice for the first existential claim to be asserted computably.

2.6.3 Markov Computability

Since most real numbers cannot be computed, why would one consider function computability
(Definition 2.42a) in terms of effective evaluation \( x \mapsto f(x) \) on all \( x \in \mathbb{R} \)? This question has led
MARKOV to study a notion where only computable reals \( x \) arise, input in the form of any Gödel
index \( e \) (i.e., basically an encoding) of a Turing machine \( M_e \) computing (rational approximations \( q_n \) up to error \( 2^{-n} \) to \( x \) and output similarly as an index \( e' \) for (rational approximations \( p_m \) up to error \( 2^{-m} \) to \( f(x) \); compare [Weih00, Section 9.6]. On the basis of SMN and UTM Theorems, it is easy to see that every function computable in the sense of Definition 2.42 is also Markov computable.

It turns out that some kind of ‘Main Theorem’ also holds for Markov Computation.

Fact 2.67. Let \( f : (-1,1) \rightarrow \mathbb{R} \) be total and Markov computable. Then \( f \) is continuous; even
\( (\rho \rightarrow \rho) \)-computable.

The second claim is known as Tseitin’s Theorem [Weih00, Theorem 9.6.6]. We illustrate the first
claim in a special case:

Proof (Heaviside’s function is Markov-uncomputable). Assume the converse; then one can solve
the Halting problem as follows: Given a(n encoding of a) Turing machine \( \langle M \rangle \), define

\[
q_{n,M} := 2^{-n} \quad \text{if } M \text{ does not terminate within } n \text{ steps;} \quad q_{n,M} := 2^{-n_0} \quad \text{if } M \text{ terminates at step } n_0 \leq n.
\]

---

Construction Reverse Mathematics can thus be viewed as a kind of complexity theory with logical axioms as resources (recall Section 1.3) instead of time or space
Observe that, if \( \langle M \rangle \notin H \), then \(|q_{n,M} - x_M| \leq 2^{-n}\) for \( x_M := 0 \); and \(|q_{n,M} - x_M| \leq 2^{-n}\) for \( x_M := 2^{-n_0} > 0 \) if \( \langle M \rangle \in H \) with \( n_0 \) denoting the number of steps \( M \) performs before termination; in particular it holds either \( h(x_M) = 0 \) or \( h(x_M) = 1 \), depending on whether \( \langle M \rangle \in H \) or \( \langle M \rangle \notin H \).

Simulating \( M \) for \( n \) steps reveals \((q_{n,M})_m \subseteq \mathbb{Q}\) as recursive uniformly in \( M \); and UTM and SMN Theorems ensure a Gödel index \( e \) for a Turing machine \( M_e \) computing \((q_{n,M})_m \) (i.e. \( x_M \)) effectively obtainable from \( \langle M \rangle \). By hypothesis, one calculates from \( e \) in turn an index \( e' \) of a Turing machine \( M_{e'} \) producing \((p_m)_m \subseteq \mathbb{Q}\) with \(|h(x_M) - p_m| \leq 2^{-m}\). In particular \( p_2 \leq 1/2 \) iff \( \langle M \rangle \in H \), i.e. by simulating the machine \( M_{e'} \) effectively obtained from \( \langle M \rangle \) one can decide the termination of \( M \)—a contradiction. 

By taking only computable \( x \) into consideration, the notion of Markov computability may arguably be not fully a model of real number computation. Moreover, it leads to pathological counterexamples of discontinuous computable partial functions on a dense domain [Weih00, Example 9.6.5].

Also, Markov himself actually did not think in terms of Turing machines and Gödelizations but in terms of constructive mathematics—plus what has become known as Markov’s Principle:

\[ \forall x : \mathbb{N} \to \{0, 1\} : \left( \neg \forall n : x(n) = 0 \right) \Rightarrow \exists n : x(n) = 1. \]

This proposition is classically true; it is well permitted in computability theory, corresponding to an unbounded search (i.e. \( \mu \)–recursion as opposed to primitive recursion) with applications for instance in dove-tailing; but, in many flavors of constructive mathematics other than Markov–style, it is rather frowned upon.

### 2.6.4 Analog Computation

Analog machines like the Differential Analyzer [Bush31] preceded digital computers historically (and for decades flourished alongside them). An elegant theory of computability for such devices has been devised by C. Shannon [Shan41], resulting in a characterization in terms of algebraic differential functions, that is, (components of vector) functions \( \vec{y} = \vec{y}(t) \) of continuous time \( t \in \mathbb{R} \) satisfying a differential equation

\[ P(t, \vec{y}(t), \vec{y}'(t)) = 0 \quad (2.10) \]

where \( P \) denotes an arbitrary multi-variate real polynomial. Since solutions to (2.10) may in general exist only locally, the domain of definition requires special attention lest the notion may become surprisingly ambiguous [Pour74, Kawa06]. Variations lead for example to a class of real recursive functions [Moor90, CMC02, GrCo03].

However, analog computation, in spite of its many relations to BCSS machines and Recursive Analysis [Pour74, Kawa06], exceeds the scope of the present survey. For further information, refer to [Orpo97, BoCa07].

### 2.6.5 Exact Computation Paradigm

The Exact Computation Paradigm is promoted, for example by Chee Yap [YaDu95], as a formalization of the requirements in Computational Geometry. Specifically, it considers inputs \( \vec{x} \) given exactly in the form of a rational (or, slightly more general, algebraic) number/vector. Only the output real \( f(\vec{x}) \) is to be approximated by rationals with prescribable precision.

A structural disadvantage is that the composition of two computable functions in general is not computable in this sense. Moreover, as with Markov (Section 2.6.3), restriction to rational or algebraic arguments makes Exact Computation not fully a model of real number computation. From a practical point of view, however, it may be closer to reality than any of the above. In Computational Geometry, specifically, most problems can be formalized as evaluating a function with both real (continuous) and discrete (combinatorial) values; see for instance Problem 2.68 below. Getting the discrete components calculated right (exactly or, equivalently, approximately up to error \( \epsilon < \frac{1}{2} \)) is crucial—lest the output be inconsistent [KMPSY04]—and can be achieved only on exact input.
**Problem 2.68 (Combinatorial Convex Hull).** Given points $\bar{x}_1, \ldots, \bar{x}_n$ in $d$-dimensional space, determine both the number and coordinates of those lying on the boundary of $\text{chull}(\{\bar{x}_1, \ldots, \bar{x}_n\})$, where

$$ \text{chull}(A) := \bigcap_{A \subseteq C \subseteq \mathbb{R}^d \text{ convex}} C = \left\{ \sum_{i=0}^d \lambda_i \bar{x}_i : \bar{x}_0, \ldots, \bar{x}_d \in A, \lambda_0, \ldots, \lambda_d \geq 0, \sum \lambda_i = 1 \right\} . $$

In this and many other cases, mere computability is easy\textsuperscript{11} to establish by means of exact rational/algebraic arithmetic because the input is given exactly. The challenge then consists in doing so efficiently, see Section 2.7.

### 2.7 Practice and Efficiency

All models are of course (and must be, recall Chapter 2) idealizations—in one direction or another—of actual numerical programming. Conversely, they have spurred (or are related to) attempts to actually realize and implement them.

As already mentioned, computer algebra systems and libraries like LEDA [BKM+95, BFMS99] and Core [KLPY99] all come reasonably close to the capabilities (and semantics) of a BCSS machine; the latter is naturally biased towards the Exact Computation Paradigm (cf. Section 2.6.5). In view of its practical applications in Computational Geometry, LEDA is particularly suitable for low (algebraic) degree calculations and includes many sophisticated optimization heuristics such as filters [FoWy93] and constructive root bounds [BFMS00]; see also [LPY05].

Another implementation, based on the aforementioned publication [BrHe98] and thus realizing the computational model of Recursive Analysis, is the iRRAM [Muel01]. Its strengths consist in calculations involving high algebraic degree and transcendental numbers, functions, and operations. See for instance [Lamb05] for a competitor.

### 2.8 My Manifesto

As pointed out in the introduction, the purpose of classical Computability Theory is to identify, with regard to discrete problems, the limits of what computers can and what they cannot do even in principle. Correspondingly, Real Computability is concerned with problems involving real numbers. BLUM, CUCKER, SHUB, and SMALE write in [BCSS98, Section 1.4]:

*The developments described in the previous section (and the next) have given a firm foundation to computer science as a subject in its own right. Use of the Turing machines yields a unifying concept of the algorithm well formalized. […] The situation in numerical analysis is quite the opposite. Algorithms are primarily a means to solve practical problems. There is not even a formal definition of algorithm in the subject. […] Thus we view numerical analysis as an eclectic subject with weak foundations; this certainly in no way denies its great achievements through the centuries.*

Investigations on Real Computability thus have relevance to practical applications; the specific range of applicability, that is the question of which practical aspects such research reflects and which it does not, depend on the model of real computation under consideration; recall Section 1.2.

Specifically, BCSS machines were introduced as an abstraction of floating point computations (one time step per operation) and neglect stability issues. In contrast, stability constitutes the major focus of Recursive Analysis as a formalization of arbitrary precision computations over $\mathbb{R}$. Each model has its own counter-intuitive implications (e.g. Example 2.22 and Proposition 2.44) and rather sharply splits the community into unconditional supporters and fierce opponents.

\textsuperscript{11}For an exception where this requires highly non-trivial arguments, see [CCKG04]
I myself prefer to avoid such dogmatism and choose from case to case either the one or the other model, depending on which one seems more appropriate and yields the more interesting and mathematically rich results. Moreover, combining the strengths of both models has proved to be a seminal synthesis, recall Sections 2.5.2 and 2.5.3 and see Section 6.4 below.
Chapter 3

Hypercomputation

Following [CoPr99], we agree with

**Convention 3.1.** Any (model of a) computer whose principal power strictly exceeds that of the Turing machine is called a hypercomputer.

This notion is far from being a definition and may involve ambiguities. For instance, the BCSS machine’s inability to decide ‘its own’ termination suggests it does not count as a hypercomputer; whereas the ability to solve the discrete Halting Problem $H$ might also justify the contrary. For the purpose of the present work it is fortunately not necessary to choose sides on this issue; see Chapter 5. Also, the case of the model of Recursive Analysis is clear: being but an ordinary Turing machine, this definitely does not constitute a hypercomputer. Instead, we shall enhance it to produce a hypercomputer in various ways and investigate the resulting power in Chapter 4.

But first a review of discrete hypercomputation: Section 3.1 recalls the basics of oracle computation. A device capable of deciding $H$ (or even a higher undecidable one like $\emptyset^{(d)}$ for some $d \geq 1$), plugged into a PC’s expansion slot, most naturally corresponds to the oracle Turing machine model. Approaches to physical solutions to $H$ will be discussed in Section 3.2. Other theoretical models for them are deduced in Section 3.3.

### 3.1 Oracle Turing Machines

The study of hypercomputers (although they were not originally called that) actually also dates back to Turing’s work: his PhD thesis [Turi39] deals with what are now known as Oracle Turing Machines. This formalization allows us to study, in a mathematically consistent way, a question like

*What could we compute—and what still not—if some (undecidable) problem $O$ were decidable?*

It also yields a way to gauge (undecidable) problems according to their ‘difficulty’:

**Definition 3.2.** Let $M^O$ denote an oracle Turing machine $M$ equipped with oracle $O \subseteq \mathbb{N}$. Problem $A$ is many–one reducible to $B$, written as “$A \leq_m B$”, if there exists a computable total function $f$ satisfying the following condition: $x$ belongs to $A$ if and only if $f(x)$ belongs to $B$. $A$ is (Turing–) reducible to $B$ (“$A \leq_t B$”) if $A$ can be decided by a Turing machine with oracle access to $B$. The Turing degree of $A$ is the set of all problems $B \equiv A$ equivalent to (i.e. reducible both to and from) $A$.

#### 3.1.1 Power of the Halting Oracle

By querying $H$, an oracle machine can decide the termination of a given Turing Machine $M$ (on a given specific input or, equivalently, on the empty input).
Fact 3.3. Every semi-decidable problem (or its complement) becomes decidable relative to \( H \).

On the other hand, the \( H \)-oracle is provably not sufficient (cf. Example 3.12) to decide the following problem even more relevant to automated software verification than \( H \) (cf. Section 1.1.2).

Definition 3.4. \( \text{Tot} \) is the question of whether a given Turing machine \( M \) terminates on all possible inputs. \( \text{Fin} \) is the question of whether the language accepted by a given Turing machine \( M \) is finite (i.e. if \( M \) terminates on at most finitely many inputs).

A characterization of computation with the support of a Halting oracle is given by

Lemma 3.5 (Shoenfield’s Limit). A function \( f : \subseteq \mathbb{N} \rightarrow \mathbb{N} \) is computable relative to \( H \) iff \( f(n) = \lim_m g(n,m) \) is the pointwise limit of an ordinarily computable function \( g : \text{dom}(f) \times \mathbb{N} \rightarrow \mathbb{N} \).

Since this claim can be concluded ‘bitwise’ from Theorem 3.9 below, we omit a separate proof but refer to, e.g., [Soar87, §III.3.3] and point out a putative pitfall. Suppose one naively simulates computation of \( n \mapsto f(n) \) in the following way:

For each query “\( x \in H? \)” answer no and, while continuing the simulation (and printing the result when it terminates), start simultaneously looking for \( x \in H \) (semi-decidable).

If it turns out that \( x \) does belong to \( H \), then restart the simulation with the corrected answer yes.

Then the sequence of outputs \( g(n,1), g(n,2), \ldots \) might never attain the correct value because the algorithm for \( f(n) \) can lead to infinitely many oracle queries when given incorrect answers. More specifically, consider the following computation: Upon input of \( n \), query “\( x \in H? \)” for \( x = n, n+1, \ldots \) and for the first positive answer output \( x \) and terminate. The Padding Lemma (recall Section 1.1.1) ensures that this yields a total function, but (a careless distribution of priorities during dove-tailing in) the above naive simulation may never terminate!

Closely related is the area of revising computation, also known as Limiting [Gold65], Trial-and-Error [Putn05], Inductive [Burg04], or General [Schm02] Turing Machines. Indeed, even semi-decidability can be regarded as a relaxation of decidability which permits finitely but unboundedly many one-sided errors in the form of false negatives: a machine may report “\( (M) \notin H \)” unless and until termination of \( M \) is detected. Let us generalize this to two-sided errors:

Definition 3.6. A revising machine may make an infinite number of steps but has to write its result onto some tape in the form of a string. This output may be reverted an arbitrary yet finite number of times:

A revising name for a (delimited) string \( \sigma \in \{0,1\}^* \circ \{\cdot\} \), \( \bar{\sigma} = (\sigma_1, \ldots, \sigma_{N-1}, \cdot) \), is a sequence \( (\vec{\tau}_m)_m \) of (delimited) strings \( \vec{\tau}_m \in \{0,1\}^* \circ \{\cdot\} \) such that \( \bar{\sigma} = \lim_m \vec{\tau}_m \), i.e. such that

\[
\forall n \leq N \ \exists M \ \forall m \geq M : \tau_{n,m} = \sigma_n
\]

(pointwise convergence); equivalently: uniform convergence \( \exists M \ \forall n \geq M : \vec{\tau}_m = \bar{\sigma} \).

By Lemma 3.5 a function \( f : \subseteq \mathbb{N} \rightarrow \mathbb{N} \) is revisingly computable (i.e. computable by a machine with the above semantics) iff it is computable relative to \( H \).

Motivation for Revising Output

Revising machines can be regarded as models that take into account that the display of, say, a video terminal may change; and, even if it eventually becomes fixed, we can never be sure when.

Recall the two most basic ascii control characters, which even the earliest text display consoles understood: BS and CR. The first, called “backspace”, moves the cursor left by one position, allowing the last printed symbol to be overwritten; the second, “carriage return”, is the command to restart output from the beginning (of the present line). 

\[\text{[12]}\]The name seems to be more historical folklore than to indicate the discoverer of the result.
Example 3.7. The character sequence

G o o d b y e CR

Hello, Mrs BS BS BS world

will display as: Hello, world.

Let us consider a non-terminating program (such as an operating system) generating an infinite sequence of characters including BS and CR; how do they appear on an (unbounded, one-line) display? Let us make

Requirement 3.8. Each character position has to settle down eventually, leading ultimately to the display of a finite string \( \vec{\sigma} \) (without BS and CR).

This kind of output is obviously equivalent to the (classical) generation of a revising name for \( \vec{\sigma} \): every change from \( \vec{\tau}_m \) to \( \vec{\tau}_{m+1} \) can be effectively expressed as a sequence of 'editing operations' involving BS; and conversely, by considering, for \( m = 1, 2, \ldots \), the first \( m \) characters of \( \vec{\sigma} \) and applying the edit instructions it contains, we obtain a string \( \vec{\tau}_m \) converging to \( \vec{\sigma} \) due to Requirement 3.8.

3.1.2 Arithmetical Hierarchy

The question \( H =: \emptyset' \) whether a given Turing machine terminates (on the empty input, say), is undecidable (relative to a decidable oracle like \( \emptyset \)) but, relative to \( \emptyset' \), it is decidable. The proof relativizes, that is, it carries over to arbitrary oracles: Termination of a given Turing machine with oracle \( H \), that is the jump \( H' =: \emptyset'' \) of the ordinary Halting problem, is undecidable relative to \( \emptyset' \) yet decidable relative to \( \emptyset'' \). Iteration

\[
\emptyset^{(d)} := \{ (M, x) | M^{\emptyset^{(d-1)}} \text{ terminates on input } x \}
\]

leads to KLEENE’s Arithmetical Hierarchy

\[
\emptyset \preceq_m \emptyset' = H \preceq_m \emptyset'' = H' \preceq_m \ldots \preceq_m \emptyset^{(d)} \preceq_m \ldots ;
\]

where “\( A \preceq_m B \)” means “\( A \preceq_m B \land B \not\preceq A \)”. A syntactical characterization of the levels of this hierarchy was obtained by E. POST:

Theorem 3.9 (Normal Form). For \( L \subseteq X := \mathbb{N} \) and \( d \in \mathbb{N} \), the following are equivalent:

a) \( L \) is semi-decidable relative to \( \emptyset^{(d-1)} \)

b) There exists a decidable \( R \subseteq X \) such that

\[
L = \{ x \in X | \exists y_1 \in \mathbb{N} \ \forall y_2 \in \mathbb{N} \ \exists y_3 \in \mathbb{N} \ldots \exists_{d, y_d} \in \mathbb{N} : \langle x; y_1, \ldots, y_d \rangle \in R \} .
\]

Here \( \langle \cdot \rangle : \mathbb{N}^* \to \mathbb{N} \) denotes a computable tupling function and \( \theta_d \) either the universal or the existential quantifier, depending on \( d \)’s parity.

c) \( L \preceq_m \emptyset^{(d)} \).

Obviously, \( L \) is decidable relative to an oracle \( O \) if both \( L \) and its complement \( X \setminus L \) are semi-decidable relative to \( O \).

Definition 3.10. Write \( \Sigma_0 = \Pi_0 = \Delta_0 = \Delta_1 \) for the class of all decidable problems. With \( d \in \mathbb{N} \), \( \Sigma_d \) is the class of all problems \( L \) of the form \([3.3] \); equivalently: of all problems semi-decidable relative to \( \emptyset^{(d-1)} \). \( \Pi_d \) denotes the class of complements of languages in \( \Sigma_d \); and \( \Delta_d := \Sigma_d \cap \Pi_d \) the class of problems decidable relative to \( \emptyset^{(d-1)} \).

Theorem 3.9 and Equation (3.2) translate to this picture:

\[
\Delta_1 \subseteq \Sigma_1 \subseteq \Delta_2 \subseteq \Sigma_2 \subseteq \ldots \subseteq \Delta_d \subseteq \Sigma_d \subseteq \ldots
\]

We shall encounter and study real variations of this hierarchy both for the BCSS model and in Recursive Analysis in Sections 5.4 and 4.1, respectively.
Proof (Theorem 3.9). \( \text{c} \Rightarrow \text{a} \): \( \emptyset^{(d)} \) is semi-decidable relative to \( \emptyset^{(d-1)} \).

\( \text{b} \Rightarrow \text{c} \) holds by induction on \( d \), starting with \( d = 1 \) by definition. For \( d > 1 \), consider

\[
K := \{ \langle x, y_1 \rangle \mid \forall y_2 \exists y_3 \ldots \theta_d y_d : \langle x, y_1, y_2, \ldots, y_d \rangle \in R \}.
\]

\( X \setminus K \in \Sigma_{d-1} \) is, by induction hypothesis, many-one reducible to \( \emptyset^{(d-1)} \). Let \( f \) denote such a computable reduction function. Then some fixed \( x \in X \) belongs to \( L \) iff there exists some \( y_1 \in \mathbb{N} \) such that \( f(\langle x, y_1 \rangle) \in \emptyset^{(d-1)} \); that is, iff the oracle Turing machine \( M^{\emptyset^{(d-1)}}_\emptyset \) searching for such a \( y_1 \) terminates. The computable mapping \( x \mapsto \langle M^{\emptyset^{(d-1)}}_x \rangle \) therefore constitutes a reduction of \( L \) to \( \emptyset^{(d)} \).

\( \text{a} \Rightarrow \text{b} \) Let \( L \) be semi-decidable relative to \( K := \emptyset^{(d-1)} \) by some oracle machine \( M \). On input of \( x \in L \), \( M \) makes finitely many oracle queries “\( z \in K? \)” where \( z \in O_+ \cup O_- \); and \( O_+ \subseteq K \) denote the set of those answered positively, \( O_- \subseteq \neg K \) those answered negatively. In fact, \( x \in L \) iff there exist finite sets \( O_+ \subseteq K \) and \( O_- \subseteq \neg K \) such that \( M \) makes only queries to \( O_+ \cup O_- \) and terminates. Now it is easy to construct an oracle-free machine \( \tilde{M} \) that semi-decides the language

\[
\tilde{L} := \{ \langle x, O_+, O_- \rangle : M \text{ terminates on } x, \text{ making oracle queries only to } O_+ \cup O_- \}.
\]

(Here we refer to a straightforward encoding of finite sets \( O \subseteq \mathbb{N} \) into integers such that \( \langle O \rangle \geq \max O \).) Therefore, \( \tilde{L} \in \Sigma_1 \); and

\[
L = \{ x \mid \exists (O_+, O_-) \in \mathbb{N} : \langle x, O_+, O_- \rangle \in \tilde{L} \land \ O_+ \subseteq K \land \ O_- \subseteq \neg K \}.
\]

Since \( K = \emptyset^{(d-1)} \) is semi-decidable relative to \( \emptyset^{(d-2)} \), \( K \in \Sigma_{d-1} \) by induction hypothesis. Condition “\( O_+ \subseteq K \)” is equivalent to “\( \forall o \leq \langle O_+ \rangle : o \notin O_+ \lor o \in K \)” (since \( \max O \leq \langle O \rangle \), see above): a formula whose first (i.e. the universal) quantifier has a bounded range. Regarding condition “\( O_- \subseteq \neg K \)”, its negation is equivalent to “\( \exists o : o \in O_- \land o \in K \)” (since \( \max O \leq \langle O \rangle \)). Now apply Lemma 3.11 below.

Lemma 3.11 (Closure Properties). a) For \( K_1, K_2 \subseteq \Sigma_d, \ K_1 \cup K_2, K_1 \cap K_2 \in \Sigma_d \).

b) If \( K \in \Sigma_d \), then \( \{ x \in X \mid \exists y : \langle x, y \rangle \in K \} \in \Sigma_d \).

c) If \( K \in \Sigma_{d-1} \), then \( \neg K \in \Pi_{d-1} \subseteq \Sigma_d \).

d) If \( K \in \Sigma_d \), then \( \{ x \mid \forall y \leq x : \langle x, y \rangle \in K \} \in \Sigma_d \).

Concerning the latter, observe that

\[
\forall y \leq x \exists z : R(x, y, z) \iff \exists \langle z_1, \ldots, z_x \rangle \forall y \leq x : R(x, y, \overbrace{z_1, \ldots, z_x}^{= R(x, (z_1, \ldots, z_x))}),
\]

that is, quantifiers with bounded range may be swapped and partially eliminated.

The following generalizes Fact 3.3.

Example 3.12 (Complete Problems). a) It holds \( \emptyset^{(d)} \in \Sigma_d \). Conversely, every problem in \( \Sigma_d \) is many-one reducible to \( \emptyset^{(d)} \).

b) The problem \( \text{Fin} \) belongs to \( \Sigma_2 \); and every problem in \( \Sigma_2 \) can be many-one reduced to \( \text{Fin} \).

c) The problem \( \text{Tot} \) belongs to \( \Pi_2 \); and every problem in \( \Pi_2 \) can be many-one reduced to \( \text{Tot} \).

See e.g. \cite{Soar87} Theorem IV.3.2] . . .

In particular, \( \text{Tot} \) is not semi-decidable relative to \( H! \).
3.1.3 Hyperarithmetical and Analytical Hierarchy

The many similarities of the Arithmetical Hierarchy (3.4) with the Borel one (2.1) have led to the effective Borel Hierarchy discussed in Section 3.1.1. It also resembles the

Polynomial Hierarchy

In complexity theory, extending the famous classes $\mathcal{P}$ and $\mathcal{NP}$ by iterated oracles gives rise to a hierarchy which can equivalently be described by alternating quantifiers [Papa94]:

\[
\mathcal{P} = \Delta^p_1 \subseteq \mathcal{NP} = \Sigma^p_1 \subseteq \mathcal{P}^{\mathcal{NP}} = \Delta^p_2 \subseteq \mathcal{NP}^{\mathcal{NP}} = \Sigma^p_2 \subseteq \ldots
\]

where $\Sigma^p_d$ is the class of languages $L$ of the form

\[
L = \{ x \in \{0,1\}^* \mid \exists y_1 \in \{0,1\}^* \forall y_2 \in \{0,1\}^* \exists y_3 \in \{0,1\}^* \ldots \theta_d y_d \in \{0,1\}^* : |y_1|, \ldots, |y_d| \leq p(|x|) \land \langle x; y_1, \ldots, y_d \rangle \in P \}
\]

for some integer polynomial $p$ and a language $P \in \mathcal{P}$. (As opposed to strictness of the Arithmetical Hierarchy, that of the Polynomial one is a serious open question worth $1,000,000$—recall Exercise 2.52.) All classes $\Sigma^p_d$, $d \in \mathbb{N}$, are contained in $\text{PSPACE}$, the collection of problems decidable within polynomial space. A variant of Cook’s Theorem (PSPACE-hardness of TQBF) implies that PSPACE coincides with the class $\Sigma^p_{\text{poly}}$ of problems of the form

\[
L = \{ x \in \{0,1\}^* \mid \exists d \in \mathbb{N} \exists y_1 \in \{0,1\}^* \forall y_2 \in \{0,1\}^* \exists y_3 \in \{0,1\}^* \ldots \theta_d y_d \in \{0,1\}^* : d \leq p(|x|) \land |y_1|, \ldots, |y_d| \leq p(|x|) \land \langle x; y_1, \ldots, y_d \rangle \in P \}
\]

for some integer polynomial $p$ and $P \in \mathcal{P}$. In particular, $\Sigma^p_{\text{poly}}$ is closed under complement! Notice that, as opposed to Equation (3.5), the number $d$ of quantifier alternations here varies polynomially in the input length $|x|$.

Transfinite Arithmetical Hierarchy

In a similar way to Equations (3.5) and (3.6), extend (3.3) and consider languages of the form

\[
L_R = \{ x \in X \mid \exists d \in \mathbb{N} \forall y_1 \in \mathbb{N} \forall y_2 \in \mathbb{N} \exists y_3 \in \mathbb{N} \ldots \theta_d y_d \in \mathbb{N} : \langle x; y_1, \ldots, y_d \rangle \in R \}
\]

for decidable $R$. Equation (3.3) describes all problems in the finite Arithmetical Hierarchy $\Sigma_{\omega} = \Pi_{\omega} = \bigcup_{d<\omega} \Delta_d$; Equation (3.7) on the other hand leads to the class $\Sigma_{\omega+1} = \Sigma_\omega$, their complements populate $\Pi_\omega$: the first transfinite levels. Taking effective unions leads to higher classes $\Sigma_\alpha$ and $\Pi_\alpha$—as long as $\alpha$ runs through recursive ordinals, that is the hierarchy stalls beyond the Church–Kleene ordinal, although this $\omega^{CK}$ is still countable; cf. e.g. [Hinn78] Section IV.4.

Analytical Hierarchy

Running through all recursive ordinals $\alpha$, $\Sigma_\alpha$ exhausts exactly the class $\Delta^1_1$ of hyperarithmetical sets: in a sense the ‘ground level’ of the Analytical Hierarchy

\[
\Sigma_\alpha \subset \Delta^1_1 \subset \Sigma^1_1 \subset \Delta^1_2 \subset \Sigma^1_2 \subset \ldots \Delta^1_d \subset \ldots
\]

where $\Delta^1_d = \Sigma^1_d \cap \Pi^1_d$ and $\Pi_d$ consists of the complements of sets in $\Sigma^1_d$. Specifically, $\Sigma^1_1$ can be described as the class of sets $L \subseteq \mathbb{N}$ of the form

\[
L = \{ x \in \mathbb{N} \mid \exists b = (b_n)_n \in \{0,1\}^\omega \forall n \in \mathbb{N} : \langle x, n; b_1, \ldots, b_n \rangle \in R \}
\]

for some decidable $R \subseteq \mathbb{N}$ [Odi89 Proposition IV.2.5].
CHAPTER 3. HYPERCOMPUTATION

3.1.4 Post’s Problem

The levels of the Arithmetical Hierarchy consist of problems far ‘more undecidable’ than the Halting problem. They arise from iterative application of the (rather coarse) jump operator $O \mapsto O'$ on arbitrary language $O$. In particular, all problems $\emptyset^{(d)}$ are either decidable (namely for $d = 0$) or (for $d \geq 1$) reducible from—that is at least as difficult as—the Halting problem $H = \emptyset^{(1)}$. Investigation of the fine structure of Turing degrees dates back to 1944 with a question from E. Post:

**Question 3.13 (Post’s Problem).** Let language $P$ be semi-decidable (and thus reducible to the Halting problem) yet undecidable. Is $P$ then necessarily also reducible from the Halting problem?

The existence of such intermediate Turing degrees that need not be r.e. follows from a result by Kleene and Post from 1954. The original question, however, was fully answered, only 12 years after it had been posed, independently by Muchnik and Friedberg. Devising the finite injury priority sophistication of diagonalization, they proved the existence of r.e. Turing degrees strictly between those of $\emptyset$ and $\emptyset' = H$; cf. [Soar87, Chapters V to VII] or [ScPr98, Section 1].

Like the similarity between Arithmetical and Polynomial Hierarchy, Post’s question and its answer resemble (and precede) complexity theory investigations [Ladn75, Scho82] about problems $P \in \mathcal{NP} \setminus \mathcal{NP}$-completeness (i.e., $P$ is not reducible to e.g. SAT).

Question 3.13 has implications for hypercomputation as follows: A (mathematical model or a physical device allowing for) solution to a problem $B$ also implies the solvability of any problem $A \preceq B$. A (hyper-) computer deciding $B$ is thus strictly more powerful than one for $A$—and more difficult to realize. So when aiming for hypercomputation, one might start with the more modest goal of solving some problem $P$ strictly easier than the Halting problem $H$. And such a $P$ to indeed exist amounts to the negative answer to Post’s Problem above.

However, being essentially based on diagonalization, proofs really establish only the existence of some $P$ with $\emptyset \preceq P \preceq_m H$ and $H \not\preceq T P$. The ‘most explicit’ form known consists of a Turing machine enumerating such a $P$, compare [ScPr98, Theorem 1.1]. However, this still very implicit representation is hardly useful for realizing, e.g. by an engineer, a hypercomputing device solving $P$. An explicit solution to Post’s Problem might be obtained from Turing machines with few states: these are conceivably incapable of universal computation while still having undecidable Halting problem [MaSe05]. Whereas this question remains open for the discrete case, a real counterpart will be answered in Section 5.3 below. There, the explicit set $Q$ of rational numbers turns out to be semi-decidable, undecidable, yet strictly easier than the real Halting problem $\mathbb{H}$.

3.2 Physical Hypercomputers

The Church–Turing Hypothesis mentioned in Section 1.4 is (often interpreted in a way) as to prohibit physical hypercomputation. However, we have already pointed out it is merely a hypothesis—in fact, rather an ambiguous if not doubtful one (Section 1.5). As a matter of fact, some recent approaches do support the existence (cf. Section 1.5.3) of hypercomputers or, more precisely, the principal (although admittedly not practical) solvability of the Halting Problem by physical systems. These suggestions give rise to models of hypercomputation other than the oracle Turing machine in Section 3.3.

3.2.1 Relativistic Hypercomputation

According to classical (i.e., Newtonian) physics, and in accordance with common experience, velocities add the naive way: An observer jogging with speed $v$, inside a train in turn running at $u$, perceives the scenery passing by at $u + v$. However, if that was strictly correct, light emitted with $v = c$ from a torch inside that train would result in an overall propagation at $u + c = c' > c$. This contradicts MAXWELL’s theory of Electrodynamics as well as Michelson and Morley’s famous experiment, which demonstrated the speed of light $c$ to be the same independently of the frame.
of reference. Such inconsistency was resolved in 1905 by A. Einstein [Eins05]. His theory of Special Relativity revealed that velocities \( u \) and \( v \) add to \( \frac{u + v}{1 + (uv/c^2)} \). While this falls reasonably close to the classical formula \( u + v \) for \( u, v \ll c \), deviations from the latter must be taken into account surprisingly often—even in practical applications like GPS. For instance, the concept of absolute time has to be abandoned in favor of a subjective one: a moving clock moves slower than a resting one—and vice versa: the famous Twins’ Paradox. According to General Relativity, also discovered and developed into a sound theory by Einstein [Eins16, Eins61], the four-dimensional space–time fabric is further distorted by gravitation, affected dynamically by masses moving within space–time themselves. The (now) so-called Malament–Hogarth Space–Times permit an observer to watch, within finite time (subject to the observer’s clock), the entire infinite future of (and subject to) an object like a Turing machine. This observation and its application for solving the Halting Problem has been credited to I. Pitowski [Hoga92]. Many details of this approach are discussed in [Hoga92, EtNe02, ShPi03, NeAn06]. For instance, one has to consider

- whether the Space–Time under consideration complies with causality;
- if a light signal emitted by the Turing Machine upon (and only in the case of) termination can actually reach the observer;
- where such Space–Times (being hard to create) are to be found, e.g. in the vicinity of certain Black Holes?

Traveling to the closest confirmed Black Hole, residing next to star V 4641, takes at least 1600 years: Too long for an instance of the Halting Problem to be solved practically. Thus the merit of this approach is more a principal rejection of the Church–Turing Hypothesis.

Investigation of principal computing capabilities constitutes the central purpose of Theoretical Computer Science. Indeed, J. van Leeuwen and J. Wiedermann have extended relativistic computation studies from computability to complexity [WiLe02].

### 3.2.2 Quantum Mechanical Hypercomputation

Non-classical or ‘new’ physics builds and extends on two pillars: Relativity Theory and Quantum Mechanics. The latter recognizes that elementary particles are in many ways different from the ‘tiny balls’ which one often imagines matter to be built up from. Such naivety makes Quantum Mechanics seem mysterious, but if one drops this implicit mental image and considers only outcomes of measurements, it becomes quite reasonable; see also [23].

In the famous double-slit experiment, for instance, once the particle (or photon) has arrived and been detected on the screen, it simply makes no sense to ask which of the two slits it passed through—unless you measure it (which, as it turns out, one cannot, without affecting the result on the screen). Similarly, for two entangled particles, the concept of individual ones simply ceases to be applicable: wondering (rather than measuring) which state ‘the left one’ may be in, causes spooky effects like instantaneous state collapses and attaching of the suffix ‘paradox’ to the the names of Einstein, Podolski, and Rosen; but if one speaks, in the best tradition of Einstein, only of measurements and the probabilities of their outcomes, the cause of headaches dissolves into a mere (although indeed spatially long-ranging) correlated distribution [Ludw54].

The computational power of operating on entangled binary systems (called qubits) has been conjectured by R. Feynman and actually demonstrated by P. Shor, L. Grover, D. Deutsch, and R. Jozsa in the form of famous algorithms named after them. For specific problems (like Integer Factorization), such quantum computers may be asymptotically exponentially faster than the best known classical algorithms. However, with respect to computability (as opposed to complexity), they can be simulated by an ordinary Turing machine [Grus09, p.3 footnote 1] and thus are not hypercomputers!

Thevertheless, two other ways of exploiting quantum mechanics have recently been proposed for actually solving the Halting Problem: one [CDS01, CaPa02, ACP04] in an intricate randomized

---

13 Admittedly, probabilities are not so much in Einstein’s sense but cannot be avoided by virtue of Bell’s Inequality
sense, the other one [Kieu02, Kie03a, Kie03b, Kie04a, Kie04b] is based on adiabatic relaxation to a system’s ground state. Both are of course hotly disputed; partly due to ambiguities as, for example, in the Church–Turing Hypothesis, and due to confusing principal feasibility with practicality. As a step towards clarifying some obscurities, [24] has pointed out a common link between the two approaches based on the following quotes:

- “Our quantum algorithm is based on [...] our ability to implement physically certain Hamiltonians having infinite numbers of energy levels” [Kie03b, top of Section 6.3];
- “The key ingredients are the availability of a countably infinite number of Fock states, the ability to construct/simulate a suitable Hamiltonian” [Kie03a, end of Section 4];
- “The new ingredients built in our ‘device’ include the use of an infinite superposition (in an infinite-dimensional Hilbert space) which creates an ‘infinite type of quantum parallelism’” [CaPa02, p.123 Section 5].

Indeed, quantum systems generally ‘live in’ (or, more formally, are described by trace-class operators on) an infinite-dimensional Hilbert space $\mathcal{H}$. A ‘standard’ quantum computer, whose power equals that of the Turing machine [Beni80], operates on an (unbounded) finite number of qubits and thus in an at most countably infinite Hilbert space $\mathcal{H}'$. But the infinitely many dimensions of $\mathcal{H}$ provide room for an infinite number of direct copies of $\mathcal{H}'$ by virtue of a pairing function. Put differently, where we can run one universal quantum computer, one can also run infinitely many of them simultaneously!

The power of infinite parallelism as a model of computation will be discussed in Section 3.3.3. Again, as in Section 3.2.1, we do not claim that this approach is in any way practical (which it is even less than ‘standard’ quantum computing regarding the many difficulties it faces in reality); instead, its purpose is to point out in what ways and due to what ambiguities the Church–Turing Hypothesis may principally fail.

### 3.2.3 Hypercomputers in Classical Physics

Sections 3.2.1 and 3.2.2 described ‘solutions’ to the Halting Problem based on ‘new’ physics. So how about classical physics: does the Church–Turing Hypothesis hold there? Surprisingly, the answer is negative! It exploits (in agreement with [Schr96]) the fact that classical physics is not just a sub-theory of Quantum or Relativity Theory but an idealization of reality of its very own.

In Classical Mechanics, for instance, a solid is not built up from elementary particles but from a continuum. Therefore, a cuboid can be carved into a comb with infinitely many teeth of decreasing width and distance, compare Figure 3.1. Moreover, by breaking off tooth no. $n$ iff $n \notin H$, we arrive at an encoding of the Halting Problem in an object compatible with classical physics. It may be decoded (and thus used to solve the Halting Problem) by probing the presence of a certain tooth with a wedge. This realizes an $H$–oracle. In order to obtain a hypercomputer, it can be combined with a Universal Turing Machine which, too, is feasible within Newtonian Mechanics [FrTo82].

A different but related construction was proposed in [BeTu06]. Here, $H$ is encoded in a marble run and probed (queried for “$n \in H$?”) by having a ball enter at computable speed $v(n)$.

Of course the difficulty (in addition to practicality issues of manufacturing an ideal infinite comb and wedge) consists in obtaining the object encoding $H$. Its existence (Section 1.5.3) may be compatible with classical physics but we definitely cannot construct it from, say, an undamaged infinite comb or even a full cuboid.

Tracing a light ray in 3D through a finite system of mirrors and hyperbolic lenses has been shown to be equivalent to Turing computation [RTY94]. This pertains to classical geometrical optics where a beam has no width, travels instantaneously, and is not subject to scattering or attenuation. In this sense, sending a light ray through this system and observing which of two screens it arrives on is equivalent to the Halting Problem: an optical hypercomputer.
3.2.4 N-Body Problem

Can a Turing machine at least simulate a physical system as simple as \( N \) particles moving under the effect of a mutual inverse quadratic potential? This includes as small a case as charged molecules or (classical) ions subject to Coulomb’s law; it also includes the very large example of planets acting under Newton’s gravitation. In combination with Newton’s Law, the resulting motion is described by a coupled system of nonlinear ordinary differential equations (ODEs)

\[
m_i \cdot \ddot{x}_i = \vec{F}_i(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n), \quad i = 1, \ldots, n
\]  

(3.9)

where the inhomogeneous term is given in 3D by

\[
\vec{F}_i(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = q_i \cdot \sum_{j \neq i} q_j \cdot \frac{x_j - x_i}{\|x_j - x_i\|^3}
\]

(3.10)

with \( q_i \) the (appropriately scaled) charge or mass of particle no. \( i \). (In [14, 15] we have contributed fast algorithms for evaluating these forces...)

It has been known since Poincaré that the solution \( \vec{x}_i(t) \) to (3.9) may behave unstably for \( N \geq 3 \). One natural condition to impose on the initial data \( \vec{x}_i(0) \) is that the particles do not collide, that is, \( \vec{x}_i(t) \neq \vec{x}_j(t) \) for all \( i \neq j \) and all \( t \). Famous results achieved by Painlevé (1897) and Sundman (1912) imply that, under this hypothesis and for \( N = 3 \), the solution does depend continuously and in fact computably on the initial data. On the other hand, for \( N = 5 \) or for \( N \to \infty \) in the plane, Xia and Gerver have shown non-collision singularities to occur [Xia92, Gerv91]. By the Main Theorem (recall Proposition 2.44), this prevents \( \vec{x}_i(t) \) from being in general computable from \( \vec{x}_i(0) \).

Similar arguments have been given by W.D. Smith [Smi06a] and A. Yao [Yao03]. But should a merely discontinuous physical system be considered a hypercomputer? A brief account of this notational issue will be given in Section 3.4.1.

3.3 Further Discrete Hypercomputers

Section 3.1 discussed Turing’s Oracle Machine as one model of hypercomputing. In addition, the approaches to physically realizing a hypercomputer in Section 3.2 naturally give rise to further such models. For two thorough surveys on this topic, refer to [Cope02, Ord06].

Observation 3.14. An important characteristic of the original Turing machine model is finiteness:

a) It has a finite control (the ‘program’, so to speak);
b) an initially blank, countable supply of memory cells

c) storing a finite amount of information each (e.g., a bit or an integer);

d) execution has to terminate within finitely many steps,

e) possibly exploiting finite parallelism (as for a nondeterministic machine).

Hypercomputation emerges not only from adding an extension to, but also by removing some of these restrictions from a TM. Oracle machines (Section 3.1), for instance, lift Condition b). The BCSS model (Section 2.3) relaxes c). The ‘Halting comb’ or ‘marble run’ encoding $H$ in Section 3.2.3 can be seen as an infinite control (a). Relativistic computers (Section 3.2.1) allow infinitely many steps (d) to be observed; they lead to Infinite–Time Machines in Section 3.3.1. The computational power obtained from dropping Condition e) is investigated in Sections 3.3.2 and 3.3.3.

### 3.3.1 Infinite Time

Recalling Section 3.2.1, Malament–Hogarth Space–Times permit observation, within finite time, of an infinite number of steps performed by a TM $M$. In other words, the TM’s computation seems to accelerate at a geometric rate, reaching infinity within finite time.

But what happens after it has performed these steps no.s 1, 2, . . . , $n$, . . . , $\infty$? Most naturally (mathematically speaking), $\infty$ means the first infinite ordinal $\omega$, after which follow ordinals $\omega + 1$, $\omega + 2$, . . . , $\omega + n$, . . . , $\omega + \omega = 2\omega$, $2\omega + 1$, . . . , $3\omega$, . . . $\omega \cdot \omega = \omega^2$, $\omega^2 + 1$ and so on; compare the transfinite levels of Borel’s Hierarchy in Section 2.2.2.

This intuitive notion has been formalized to ordinal-length computations on so-called Infinite-Time TMs [HaLe00]. Such a machine may perform an infinite number of steps yet terminate, say after the $\omega + 3$rd one; or it may not terminate. (The Halting Problem for infinite time TMs is of course undecidable to an infinite time TM.) That differs from, and extends, the model in Recursive Analysis (Section 2.4) where any valid computation takes exactly time $\omega$. For instance, an infinite time TM has the power to first read the entire contents of the input tape (e.g a rational sequence $(q_n)$), then perform some calculations, and finally output a resulting sequence $(p_m)$, all within $3\omega$ steps. In Recursive Analysis on the other hand, these three phases must execute simultaneously—and require any computation to be continuous (recall Proposition 2.44).

### 3.3.2 Fair Nondeterminism

Regarding Observation 3.14e), even just nondeterminism can give rise to super-Turing computation, as has been pointed out in [BST89]. Recall

**Definition 3.15.** A nondeterministic TM $N$ accepts input $x$ if there exists a sequence of nondeterministic choices for which $N$ on $x$ reaches an accepting state.

Sections 5.1.1 and 5.1.1 will consider two kinds of nondeterministic hypercomputation on real numbers. Definition 3.15 so far actually amounts to “fair nondeterminism” in the sense of VAN EMDE BOAS, SPAAN, and TORENVLIET [BST89]. It does allow the Halting Problem to be solved (and thus counts as hypercomputing) in that a nondeterministic TM can easily be constructed to accept $n \in \mathbb{N}$ iff $n \in H$! In classical nondeterminism, which can be simulated by a deterministic TM, the following (perhaps subtle) further condition is imposed:

**Definition 3.15 (cont.)** The nondeterministic TM $N$ decides language $L$ if it accepts exactly inputs belonging to $L$ and if furthermore all possible nondeterministic choices of $N$ lead to an eventually terminating computational path.

By application of König’s Lemma this implies that, for every input $x$, the running time of $N$ on $x$ remains bounded even independently of its nondeterministic choices.
3.3. FURTHER DISCRETE HYPERCOMPUTERS

Figure 3.2: Sample Initial and Two Successor Configurations of Life.

3.3.3 Infinite Parallelism

The prospering field of Parallel Computing knows and has agreed upon a small collection of models as theoretical abstractions for devising and analyzing new algorithms for various actual parallel machines [Atal99, Sections 45.2 and 47.2]. With respect to their principal power, that is concerning computability rather than complexity, they are of course all equivalent to the TM.

However, on the issue of infinite parallelism (motivated of course by Section 3.2.2), seemingly no such agreement has been established, cf. e.g. [EdWe03, p.284]; and in fact no equivalence, either, as it will turn out. For instance, what kind of countably infinitely many individual computers are to operate concurrently: TMs or finite automata? In the first case, do they all execute the same program? When is the result to be read off? The answers to these questions fundamentally affect the capabilities of the resulting system.

Infinite Cellular Automata

Consider parallelism in an infinite cellular automaton in the plane. More specifically, we refer to CONWAY’s famous Game of Life [BCG04, Ch. 25] where in each step, any cell’s successor state is determined concurrently by its present state as well as by those of its eight adjacent ones’ as follows (cf. Figure 3.2):

- A dead cell with exactly three neighbors alive becomes alive, too; otherwise it remains dead.
- A living cell with two or three neighbors alive stays alive; otherwise (0,1,4...8 living neighbors, that is) it dies.

Definition 3.16. Starting from a given initial configuration, Life terminates if the sequence of successor configurations eventually stabilizes. This resulting configuration is called rejecting if it is empty (every cell dead), otherwise accepting.

An initial configuration is finite if, out of the infinite number of cells, only finitely many are alive.

Theorem 3.17. Life with finite initial configuration is a system with infinite parallelism yet equivalent to the TM. More precisely:

i) Given a finite initial configuration, its evolution through Life can be simulated by a TM.

ii) There is a finite initial configuration capable of simulating the Universal (and thus any) TM.

Simulation here means: The TM terminates/accepts/rejects iff Life terminates/accepts/rejects.

Proof. Life is easily programmed on a Turing Machine, for example by beginning with the source code for one of the many correct implementations of Life available on the Internet. The converse Claim ii) is a famous result based on a complicated construction; see [BCG04, Ch. 25] for details.
In particular, Life matches but does not exceed the computing capabilities of a TM; cf. [Atal99, Section 26.4(1)].

The finiteness of the initial configuration is of course crucial here. One can indeed see that infinite initial configurations in ii) correspond to non-blank memory contents and thus to the dropping of both Conditions e) and b) in Observation 3.14.

### Infinite Turing Concurrency

In order to focus on the power obtained from infinite parallelism only (that is, by removing just Condition e), now imagine that the finite automata are replaced by TMs. Strictness of Chomsky’s Hierarchy implies that a single TM is provably more powerful than a single automaton [Atal99, Section 25.3]. One may therefore expect that the capabilities of an infinite number of TMs exceed those of an infinite number of automata (and thus actually lead to hypercomputation); by how much, however, depends subtly on ambiguities which we shall remove by proposing the following

**Definition 3.18.** Fix a problem \( L \subseteq \mathbb{N} \) and a countably infinite family \((M_k)_{k \in \mathbb{N}}\) of TMs. This family solves \( L \) if

i) each \( M_k \), upon input of any \( x \in \mathbb{N} \), eventually terminates and

ii) for each \( x \in \mathbb{N} \) it holds: \( x \in L \) iff at least one \( M_k \) outputs “1” (accepts).

Even though each individual \( M_k \) halts, the time required to do so may depend on \( k \); so it can take infinitely long for the entire family \((M_k)_{k \in \mathbb{N}}\) to terminate. In order to get the entirety of all their answers within finite time, the following additional requirement is therefore important:

**Definition 3.18 (continued)**

iii) Upon input of any \( x \in \mathbb{N} \), all \( M_k \) terminate within finite time bounded independent of \( k \).

While this seems sensible at first glance, a second thought reveals that, even with this restriction, the resulting notion of ‘infinitely parallel computability’ is still unreasonable: simply because any problem \( L \subseteq \mathbb{N} \) becomes trivially solvable by an appropriate family \((M_k)_{k \in \mathbb{N}}\). To this end let the program executed by \( M_k \) store the constant “1” if \( k \in L \) and the constant “0” otherwise. Let its main part then operate as follows: Upon input of \( x \in \mathbb{N} \) test whether \( x = k \); if so, output the stored constant, otherwise output “0”; then terminate.

Observe that, in accordance with iii), the test “\( x = k \)” can indeed be performed in time depending only on \( x \) but not on \( k \) by comparing only the initial segments up to the length of \( x \). The trick lies of course in the family \((M_k)_{k \in \mathbb{N}}\) solving \( L \) being shown to merely exist. More precisely, i)–iii) fail to require that the descriptions of all \( M_k \) and their constants be computable from \( k \). Therefore, we finally include this uniformity condition as well.

**Definition 3.18 (concluded)**

iv) A TM \( M_0 \) is capable of generating, upon input of \( k \), (the encoding of) \( M_k \).

### Computational Power of Infinite Turing Concurrency

It turns out that Definition 3.18(i-iv) indeed yields an interesting non-trivial way of hypercomputation. More precisely, we show that, in this sense, infinite Turing parallelism is capable of solving exactly class \( \Sigma_1 \) in the Arithmetical Hierarchy (Section 3.1.2), that is all classically semi-decidable problems. In particular, the Halting Problem \( H \) becomes solvable but \( \text{Tot} \) does not (Example 3.12c).

**Theorem 3.19.** The Halting Problem is solvable by an infinity of TMs working in parallel in the sense of Definition 3.18(i-iv).
3.4. CRITIQUE OF HYPERCOMPUTING

Proof. For each \( k \in \mathbb{N} \), consider the TM \( M_k' \) working as follows: Given \( M \) and \( x \), it simulates \( M \)'s operation on \( x \) for the first \( k \) steps; if \( M \) halts within these steps, then \( M_k' \) outputs “1” and terminates, otherwise it outputs “0” and terminates. In other words, \( M_k' \) is basically a Universal Turing Machine \( U(\cdot; k) \) with an additional counter for the number of steps simulated so far in order to abort as soon as this counter exceeds the prescribed threshold \( k \).

Observe that the family \( (M_k')_k \) thus defined satisfies i) and ii) from Definition 3.18. Moreover, since \( M_k' \) is basically \( U \) with the last argument fixed to \( k \), one easily confirms that (iv) an appropriate TM \( M_{k0} \) can indeed generate from \( k \) an encoding of this \( M_k' \). And finally it is known \([\text{DuKo00}, \text{Proposition 1.17}]\) that simulation by a Universal TM is possible with at most quadratical overhead, i.e., \( M_k' \) can be achieved with running time of \( t(n) \leq c \cdot (n \cdot k)^2 \) for some constant \( c \).

Here, \( n \) denotes the length of the input to \( M_k' \), that is, of the joint binary encodings of \( x \) and \( M \).

Now let \( M_k \) be the TM obtained from applying Linear Speed-Up Lemma 3.20 given below to \( M_k' \) with \( C := k^3 \). It follows that \( M_k \) has a running time bounded independently of \( k \), that is, it complies with iii) while still satisfying i), ii), and iv).

In order to achieve Property iii) in the above proof, the crucial ingredient is the classical construction given below. In analogy to MOORE’s empirical law of technological progress, it basically says that a TM can be accelerated by any constant factor.

**Lemma 3.20 (Linear Speed-Up).** For each \( C \in \mathbb{N} \) and any TM \( M' \) of time complexity \( t(n) \), there exists another TM \( M \) simulating \( M' \) within running time \( n + t(n)/C \). \( M \) can be obtained computationally from \( M' \); i.e., there is a fixed further TM which, given an encoding of any \( M' \) and \( C \), outputs an encoding of \( M \) as above.

Proof. See for instance [\text{Atal99, Theorem 24.5(b)}].

Every semi-decidable problem is the termination problem of an appropriate TM [\text{Soar87, Theorem II.1.2}]. Theorem 3.19 thus implies that infinite Turing parallelism can indeed solve \( \Sigma_1 \). In fact, the converse holds as well:

**Theorem 3.21.** A problem \( L \subseteq \mathbb{N} \) is solvable in the sense of Definition 3.18(i-iv) iff semi-decidable.

Proof. It remains to consider the case that \( L \) is solvable by some parallel family \( (M_k)_k \) according to Definition 3.18. We are going to semi-decide \( L \) on a single TM by means of the following simulation: Upon input of \( x \in \mathbb{N} \) and for each \( k \in \mathbb{N} \),

- obtain from \( M_0 \) a description of \( M_k \) by virtue of iv)
- and simulate \( M_k \) on input \( x \). (Observe its termination according to Property i)
- If output is “1”, halt; otherwise proceed with next \( k \).

This algorithm indeed terminates iff at least one \( M_k \) outputs “1”, that is (ii), iff \( x \in L \).

3.4 Critique of Hypercomputing

The feasibility of hypercomputation is disputed in an almost dogmatic fashion, with hypocrisy on the part of advocates and trivialization by critics. Disputes are further heated by mutual misunderstandings caused by cultural and semantical differences (e.g. regarding the notion of a ‘proof’, to start with) implicit in the various scientific disciplines involved: computer scientists, physicists, philosophers, mathematicians; not to mention hobby researchers attracted by the popularity of the subject. The ambiguity of the Church–Turing Hypothesis and its various nonequivalent formulations (Section 1.4) is added to all this; see for instance [\text{Cope02}].
3.4.1 Harnessability

Is a physical process which cannot be simulated by a TM considered a violation of the hypothesis? Or does one require the process to be in addition capable of universal computation, that is to in turn simulate a TM? That distinction was pointed out explicitly in—and seemingly not before—[Ord02, Section 2.2]. (It is also closely related to the question of whether solving some undecidable problem necessarily induces hypercomputation, see Section 3.1.4.) Interpreted in the first way, the hypothesis requires any physical process on real numbers to be continuous by Proposition 2.44. This indeed complies with ARISTOTLE’s and LEIBNIZ’ “Natura non facit saltus”; yet nowadays we know for instance of quantum leaps and discontinuous behavior even within classical mechanics (compare Section 3.2.4). As pointed out in Section 1.5 the mathematical theory of quantum gravitation also contains terms which cannot be calculated.

But should we really consider every uncomputable physical system a hypercomputer? In a related context, does a randomized TM already count as hypercomputation in that it can, by simply printing a infinite random sequence of bits, do something a deterministic one cannot do: certainly that string will be uncomputable!

Following [Ord02, Section 2.2], a more strict interpretation of the physical Church–Turing Hypothesis would not claim that every system is simulatable by a TM, but only those harnessable for classical computing: in the sense of being able to, conversely, simulate a (universal) TM; compare Hypothesis 1.1. On the other hand, one may argue that a system incapable of universal classical computation can be combined with a physical TM [FrTo82, Beni80] to yield a hypercomputer in the stricter sense.

3.4.2 Practical Feasibility

Can we actually build a hypercomputer, based on the suggestions from Section 3.2?

The specific approach of T. Kieu [Kieu02, Kie03a, Kie03b, Kie04a, Kie04b] has been credited by W.D. SMITH [Smi06b] with being conceptually seminal indeed but suffering from several flaws, some of which he succeeded in mending, while others seem to be seriously wrong. Regarding the general underlying perspective of infinite quantum parallelism (Section 3.2.2), realizability is of course even more questionable than that of its finite variant in ‘standard’ Quantum Computing [Grus99, Section 7.2].

The general relativistic approach and some of its criticisms are thoroughly discussed in [ShPi03]. For instance, a non-terminating TM flying into the Black Hole, although observed within finite time, will have to run indefinitely and access a truly infinite working tape—which seems hard to realize. On the other hand, the same applies to the semi-decidability of the Halting Problem, that is to ordinary (as opposed to hyper-) computation.

3.4.3 Relevance

Independently of the status of the Church–Turing Hypothesis, we are well advised to continue studying the principal capabilities of models of computation: a core topic of Theoretical Computer Science. Too narrow a focus on realizability and practicality would as a final consequence force us to abandon even the Turing model with its unbounded working tape and running time and to resort to an ultrafinitistic point of view. Imposing such kinds of restrictions on computer science corresponds in mathematics to what DAVID HILBERT wrote in 1928 about the restrictions demanded by BROUWER: it “would be the same, say, as proscribing the telescope to the astronomer or to the boxer the use of his fists.”

3.5 Real Hypercomputation

The main contribution of the present work, Part II combines real computability with hypercomputation: it studies models of real number computation beyond the Church–Turing Hypothesis.
Technically speaking, this field benefits from the synthesis of logical properties (from discrete hypercomputation) with the algebraic and topological ones induced from the scintillating structure of real numbers. The goal is a taxonomy of which problems are, and which are not, solvable by which kinds of real hypercomputers. It leads to a very elegant, beautiful, and rich theory with many interesting theorems.

We focus on extensions of the two major models: Recursive Analysis (Chapter 4) and BCSS machines (Chapter 5).
Part II

Real Hypercomputation
Chapter 4

Arithmetical Hierarchies in Recursive Analysis

The present chapter describes real counterparts to the classical Arithmetical Hierarchy of subsets of integers (Section 3.1.2) in Recursive Analysis. (Chapter 5 below will deal with BCSS hypercomputers...)

4.1 Effective Borel Hierarchy

Many similarities between the Borel (Section 2.2.2) and the Arithmetical hierarchy were observed long ago and led to the study of effective descriptive set theory [Hinm78, Mosc80]. For instance, Section 2.4.4 can be regarded as equipping the class \( \Sigma^0_1 \) of open subsets of \( \mathbb{R}^k \) with a notion of effectivity. Moreover, it extends straightforwardly from \( \mathbb{R}^k \) and \( \mathbb{N} \) (with discrete topology) to much more general separable effective metric spaces including \( \{0, 1\}^\omega \) and \( \mathbb{N}^\omega \).

**Definition 4.1 (Effective Borel Hierarchy).** For a second-countable topological space \( X \), fix an enumeration \( (B_p)_{p \in \mathbb{N}} \) of a basis of its open sets.

a) Such an open set \( X \subseteq U \) is r.e. iff a Turing machine can output a sequence \( (p_n) \subseteq \mathbb{N} \) such that \( U = \bigcup_n B_{p_n} \). \( \Sigma^0_1(2^X) \) is the class of r.e. subsets of \( X \).

b) For \( d \in \mathbb{N} \), \( \Sigma^0_d(2^X) \) consists of all sets

\[
\bigcup_{m_1} \bigcap_{m_2} \cdots \bigcirc_{m_d} B_{p(m_1,\ldots,m_d)}
\]

for a Turing–computable (multi-)sequence \( p = (p_{m_1,\ldots,m_d}) \) of integers. Here “\( \bigcirc_{m} S_m \)” means “\( \bigcup_{m} S_m \)” for odd \( d \) and “\( \bigcap_{m} (X \setminus S_m) \)” for even \( d \).

c) \( \Pi^0_d(2^X) \) are the complements of sets in \( \Sigma^0_d(2^X) \); and \( \Delta^0_d(2^X) := \Sigma^0_d(2^X) \cap \Pi^0_d(2^X) \).

Observe that this Definition agrees with the classical Definition 3.10 for \( X = \mathbb{N} \) with the singletons \( B_p := \{p\} \) as topological basis. The above notions have been studied in detail in [Mosc80, Chapter 3] and [Brat05]; specifically for \( \Pi^0_2(2^\mathbb{R}) \) see e.g. [Weih00, Exercises 4.3.17+18]. In fact it obviously holds \( \Sigma^0_d \subseteq \Sigma^0_{d+1} \) for any \( X \), justifying the name Effective Borel Hierarchy. Conversely, since every integer sequence \( p \) is computable relative to an appropriate oracle \( O \), each \( S \in \Sigma^0_d \) belongs to a relativized class \( \Sigma^0_d[O] \).

\[\text{We deliberately deviate from the usual notation } \Sigma^0_1(\mathbb{R}) \text{ in order to avoid conflicts with the class of lower-computable real numbers in Section 4.2.}\]
Remark 4.2. A set $A$ on the ground level $\Delta_1$ of Baire space $X = \mathbb{N}^\omega$, although decidable relative to some appropriate oracle $O \subseteq \{0, 1\}^*$, is in general not the section $B_O = \{\bar{\alpha} : (\bar{\alpha}, O) \in B\}$ of some decidable subset $B$ of $X \times \{0, 1\}^*$; cf. [Mosc80, Exercise 3F.9].

On the other hand, for the respective ground levels $\Delta_1$ of (connected) Euclidean and of (compact) Cantor space, an oracle is not needed at all: For $X = \mathbb{R}^k$ it holds $\Delta_1 = \{0, \mathbb{R}^k\} = \Delta_1$; and for $X = \{0, 1\}^\omega$ it holds $\Delta_1 = \Delta_1$ because every clopen subset of a compact space coincides with a finite union of basic open sets and is thus r.e. (In that respect, the effective Borel Hierarchy on Baire space $X = \mathbb{N}^\omega$ behaves more generically.) In particular there is no generic transfer (meta-)theorem from product spaces to $\mathbb{R}$. Technically this fails because a total representation of $\mathbb{R}$ equivalent to $\rho$ cannot exist [Weih00, Theorem 4.1.15.1].

The strictness of this hierarchy requires additional conditions on $X$; it fails for instance on finite spaces.

Fact 4.3. For $d \in \mathbb{N}$ and for our prototype examples $X = \mathbb{R}^k$, $X = \{0, 1\}^\omega$, and $X = \mathbb{N}^\omega$, it holds

a) $\Sigma_d \cup \Pi_d \subsetneq \Delta_{d+1}$.

b) Even stronger, $\Sigma_d \not\subseteq \Delta_d$.

Proof. Cf. e.g. [Hinm78, Theorem III.1.9 and Exercise III.1.28].

That is, the light-face (normal print) classes strictly grow with respect to both computational and topological complexity (gauged in terms of the classical, bold-face Borel Hierarchy).

![Figure 4.1: Classical (bold-face) and effective (light-face) Borel Hierarchy on subsets of $\mathbb{R}^k$ and of $\{0, 1\}^\omega$.](image)

Regarding computational complexity, recall (Theorem 3.9) that in the case of the discrete space $X = \mathbb{N}$—where every subset belongs to $\Sigma_1$ anyway—$\Sigma_2$ coincides with the relativization of $\Sigma_1$ to $\emptyset'$. This differs considerably in the Euclidean case:

Theorem 4.4. For subsets of $X = \mathbb{R}^k$, it holds

a) $\Sigma_1[\emptyset'] \subsetneq \Sigma_2 \cap \Sigma_1$.

b) $\Pi_2 \cap \Sigma_1 \subsetneq \Sigma_1[\emptyset']$.

Section 4.4.4 establishes an elegant characterization of $\Sigma_1[\emptyset']$. We do not (yet) known the answer to

Question 4.5. Where is $\Pi_2 \cap \Sigma_1$ located compared to $\Sigma_1[\emptyset']$? Where does $\Sigma_2 \cap \Sigma_1$ lie in comparison with $\Sigma_1[\emptyset']$?

Proof. Proof (Theorem 4.4). (An even uniform strengthening of) the inclusion in Claim a) will be established in Theorem 4.68a) below. The strictness of this inclusion follows from taking complements in Proposition 4.7 below. Claim b) is included in the (uniform) Theorem 4.68. 

\[ \square \]
4.1. EFFECTIVE BOREL HIERARCHY

4.1.1 Transfinite Levels

The finite Borel Hierarchy \( \Sigma_{<\omega} := \bigcup_{d<\omega} \Sigma_d \) is not closed under countable unions: for \( S_d \in \Sigma_d \), \( \bigcup_d S_d \) may belong to transfinite Borel class \( \Sigma_{d+1} \), recall Section \( \ref{sec:transfinite} \). Similarly, the Arithmetical Hierarchy on \( X = \mathbb{N} \) was extended to recursive transfinite ordinal levels \( \Sigma_\alpha \) up to the hyperarithmetical class \( \Delta^1_1 \); recall Section \( \ref{sec:hyperarithmetical} \).

The same applies to the light-face hierarchies on other spaces like \( X = \mathbb{R}^k \), \( X = \{0,1\}^\omega \), and \( X = \omega^\omega \) \cite[THEOREM IV.4.15]{Hinm78}. On the other hand (as opposed to Fact \( \ref{fact:finite-borel} \)) they (and their topological complexities) stall at \( \Delta^1_1 \).

4.1.2 Representing the Borel Hierarchy

Recall (Remark \( \ref{rem:tte} \)) that TTE aims at turning notions of computability into representations. Example \( \ref{ex:encoding-borel} \) is exactly in that spirit, using Fact \( \ref{fact:encoding-borel} \) for \( r.e. \) open sets as a guide to defining an encoding of the class \( \Theta^k = \Sigma_1(\mathbb{R}^k) \) of all open subsets of \( \mathbb{R}^k \). Taking complements immediately leads to the natural representation \( \psi_S^k \) of the class \( \Pi_1 \) of closed subsets of \( \mathbb{R}^k \). V. Brattka in \cite[DEFINITION 3.1]{Brat05} has extended these canonically to higher levels of Borel's Hierarchy:

**Definition 4.6.** Fix a second-countable topological space \( X \) with enumeration \( (B_i)_{i \in \mathbb{N}} \) of a basis of its open sets. Let

\[
\begin{aligned}
\delta_{\Sigma_1} &\text{ encode } U \subseteq \Sigma_1 \text{ as a sequence } p = (p_m) \text{ of integers such that } U = \bigcup_m B_{p_m}, \\
\delta_{\Pi_1} &\text{ encode } A \subseteq \Pi_1 \text{ as a sequence } p = (p_m) \text{ of integers such that } A = \bigcap_m (X \setminus B_{p_m}), \\
\delta_{\Sigma_2} &\text{ encode } S \subseteq \Sigma_2 \text{ as a double sequence } p = (p_{m,n}) \text{ of integers such that } S = \bigcup_m \bigcap_n (X \setminus B_{p_{m,n}}), \\
\delta_{\Pi_2} &\text{ encode } T \subseteq \Pi_2 \text{ as a double sequence } p = (p_{m,n}) \text{ of integers such that } T = \bigcap_m \bigcup_n B_{p_{m,n}}, \\
\vdots \\
\delta_{\Sigma_d} &\text{ encode } S \subseteq \Sigma_d \text{ as a multi-sequence } p = (p_{m_1,\ldots,m_d}) \text{ of integers such that } S \text{ coincides with Equation } \ref{eq:sigma-d}, \\
\delta_{\Pi_d} &\text{ encode } T \subseteq \Pi_d \text{ as a multi-sequence } p = (p_{m_1,\ldots,m_d}) \text{ of integers such that } T \text{ coincides with Equation } \ref{eq:sigma-d}.
\end{aligned}
\]

Put differently, a \( \delta_{\Sigma_d} \)-name of \( S \) \( \in \Sigma_d \) consists inductively of the countable product (Fact \( \ref{fact:inductive} \)) of \( \delta_{\Pi_{d-1}} \)-names of sets \( T_m \in \Pi_{d-1} \) with \( S = \bigcup_m T_m \); and a \( \delta_{\Sigma_d} \)-name of \( S \) \( \in \Sigma_d \) is simultaneously an \( \delta_{\Pi_d} \)-name of \( T := X \setminus S \in \Pi_d \). The inductive start corresponds to \( \delta_{\Sigma_1} = \theta_\omega \) and \( \delta_{\Pi_1} = \psi_\omega \).

Observe the compatibility of these notions with Definition \( \ref{def:co-compact} \): a set \( S \subseteq \Sigma_d \) belongs to \( \Sigma_d \) iff \( S \) is \( \delta_{\Sigma_d} \)-computable; similarly for \( \Pi_d \), \( \Pi_d \), and \( \delta_{\Pi_d} \). Now we are ready to establish the previously announced

**Proposition 4.7.** There exists a compact \( A \subseteq \mathbb{R} \) which belongs to both \( \Pi_2 \) and \( \Sigma_2 \) but not to \( \Pi_1[\emptyset] \).

This disproves \cite[CONJECTURE 2.15]{Conj}. Roughly speaking, both \( \Pi_2 \) and \( \Sigma_2 \) involve a countable intersection and a countable union, whereas \( \emptyset \) makes feasible only one quantifiable.

**Proof.** We will first construct \( A \subseteq [0, \infty) \) as an unbounded set of countably many isolated points; this can then be turned into a compact one (without affecting its effectivity properties) by taking the image under the computable function \( 1/\exp \) and including \( \{0\} \).

Let \( L = \{x \in \mathbb{N} : \forall y \in \mathbb{N} \exists z \in \mathbb{N} : (x,y,z) \in R\} \) denote a discrete language complete for Kleene’s class \( \Pi_2(2^\omega) \) with decidable \( R \subseteq \mathbb{N} = \{1,2,\ldots\} \); \textsc{Tot} for instance will do fine (Example \( \ref{ex:tot} \)). For \( x \in L \) let \( z(x,y) \) denote the smallest \( z \in \mathbb{N} \) such that \( (x,y,z) \in R \); \( z(x,y) := \perp \) if \( x \notin L \). Finally let \( r(x) := \sum_{y=1}^\infty 2^{-\sum_{i=1}^y z(x,i)} \in (0,1] \) (or \( r(x) = \perp \)). Interestingly, \( x \mapsto x - r(x)/2 \)
defines a Cauchy–computable sequence on $L \subseteq \mathbb{N}$: by searching for $z \in \mathbb{N}$ with $\langle x, y, z \rangle \in R$ one can, given $x \in L$ and $y \in \mathbb{N}$, determine $z(x, 1), \ldots, z(x, y)$ and thus obtain (at least) the first $y$ binary digits of $r(x)$. Now consider the closed set $A := \{ x - r(x)/2 : x \in L \}$.

- Suppose $A \in \Pi^1_1[\emptyset']$. Then one could, given $x \in \mathbb{N}$ and relativized to $\emptyset'$, obtain a $\psi_1$–name of $A \cap [x - \frac{1}{2}, x]$ \cite[Theorem 5.1.13.2]{Weih00} which permits, relative to $\emptyset'$, semi-decision of whether $A \cap [x - \frac{1}{2}, x] = \emptyset$ (Fact 2.51); equivalently: whether $x \in L$—contradicting that $\Pi^1_2$–complete $L$ cannot be semi-decided relative to $\emptyset'$ (Theorem 3.9).

- Note that, for fixed $x \in \mathbb{N}$, every $z \in \mathbb{N}$ has a unique and computable decomposition

$$z = \sum_{i=1}^{y} z(x, i) + z' \text{ where } z(x, 1), \ldots, z(x, y) \neq \bot \land \langle x, y+1, 1 \rangle, \ldots, \langle x, y+1, z' \rangle \notin R \ (4.2)$$

found by bounded search. Define the open intervals $U_{x,z} \subseteq \mathbb{R}$ by

$$U_{x,\sum_{i=1}^{y} z(x,i) + z'} := \left( x - \sum_{j=1}^{y} 2^{-\sum_{i=1}^{y} z(x,i)-1} - 2^{-\sum_{i=1}^{y} z(x,i)-z'}, x - \sum_{j=1}^{y} 2^{-\sum_{i=1}^{y} z(x,i)-1} \right)$$

as illustrated in Figure 4.2. Observe that $U_{x,z+1} \subseteq U_{x,z} \subseteq U_{x,1} = [x - \frac{1}{2}, x]$ for all $x, z \in \mathbb{N}$; indeed, when $z'$ reaches $z(y+1)$, $U_{x,\sum_{i=1}^{y+1} z(i)}$ has as its upper interval point the center of
4.2. ARITHMETICAL HIERARCHY OF REAL NUMBERS

Recall that in Recursive Analysis (as opposed to the BCSS model) the computability of even single reals is a non-trivial property: the number $\sum_{n\in\mathbb{N}} 2^{-n}$ lacks computability but is naively computable (Example 2.37c) and also becomes obviously computable relative to the Halting oracle. This is by no means a coincidence, as Ho has shown

**Proposition 4.8 ([Ho99, Theorem 9]).** For an arbitrary oracle $O$, a real number $x$ is naively computable (i.e. $|x - q_n| \to 0$ for a computable rational sequence $(q_n)_n$) relative to $O$ if and only if $x$ is Cauchy-computable (i.e. $|x - p_m| < 2^{-m}$ for a computable rational sequence $(p_m)_m$) relative to $O$.

That is, a jump makes the difference between approximations with or without effective error bounds: fast $2^{-n}$ convergence versus ultimate convergence. This result can be regarded as a continuous version of Shoenfield’s Limit Lemma [3.5], replacing integer-valued $f : \mathbb{N} \to \mathbb{N}$ with rational-valued sequences $g : \mathbb{N} \to \mathbb{Q}$. The difference is that (naive) convergence means ultimate stabilization in the first case but not in the latter:

$$\exists N \forall n \geq N : f(n) = f(N) \quad \forall m \exists N \forall n \geq N : |g(n) - g(N)| \leq 2^{-m}.$$  

A proof of Proposition 4.8 will be an immediate consequence of a more general result in Section 4.4. Zheng and Weihrauch in [ZhWe01] have extended Ho’s above result from a single jump to arbitrary levels of the Arithmetical Hierarchy:

**Theorem 4.9.** For $x \in \mathbb{R}$, $d \in \mathbb{N}$, and oracle $O$, the following are equivalent:

- $x$ has a binary expansion recursive relative to $O^{(d)}$
- $x$ is computable relative to $O^{(d)}$
- $x = \lim_{n} q_{n}$ for a rational sequence $q_{n}$ computable relative to $O^{(d-1)}$
- $x = \lim_{m} q_{(n,m)}$ for a sequence $q_{(n,m)} \subseteq \mathbb{Q}$ computable relative to $O^{(d-2)}$
- ...
- $x = \lim_{m} q_{(n_{1},\ldots,n_{d})}$ for a rational sequence computable relative to $O$.

Similar characterizations hold for left-computable real numbers:
Theorem 4.10. For \( x \in \mathbb{R} \), oracle \( \mathcal{O} \), and \( d \in \mathbb{N} \), the following are equivalent:

- \( x = \sup_n q_n \) for a rational sequence \( q_n \) computable relative to \( \mathcal{O}^{(d-1)} \)
- \( x = \sup_m \inf_n q_{(n,m)} \) for a sequence \( q_{(n,m)} \subseteq \mathbb{Q} \) computable relative to \( \mathcal{O}^{(d-2)} \)
- \( \ldots \)
- \( x = \sup_n \ldots \lim g_{(n_1, \ldots, n_d)} \) for a rational sequence computable relative to \( \mathcal{O} \)

where the last “\( \sup \)” means “\( \inf \)” for \( d \) even and “\( \sup \)” for odd \( d \).

Recall that by Example 2.41 and as opposed to the first item of Theorem 4.9, real lower approximable numbers in the sense of Theorem 4.10 cannot be characterized by semi-decidable binary expansion even for \( d = 1 \).

Definition 4.11 (Arithmetical Hierarchy of Real Numbers). Let the class \( \Delta_d(\mathbb{R}) \) consist of those \( x \in \mathbb{R} \) that satisfy one (and thus all) conditions from Theorem 4.10 and let \( \Sigma_d(\mathbb{R}) \) contain those \( x \in \mathbb{R} \) that satisfy one (and thus all) conditions from Theorem 4.10.

This gives rise to a hierarchy similar to Equation (3.4):

\[
\Delta_1(\mathbb{R}) \subseteq \Sigma_1(\mathbb{R}) \subseteq \Delta_2(\mathbb{R}) \subseteq \Sigma_2(\mathbb{R}) \subseteq \ldots \subseteq \Delta_d(\mathbb{R}) \subseteq \Sigma_d(\mathbb{R}) \subseteq \ldots \quad (4.3)
\]

In the spirit of TTE (recall Remark 2.59), Theorems 4.9 and 4.10 canonically lead to

Definition 4.12 (Arithmetical Hierarchy of Real Representations). Consider the representations of \( \mathbb{R} \) where a real \( y \) is encoded as

- \( \rho \) : a rational sequence \( (p_m) \) such that \( |y - p_m| < 2^{-m} \) (i.e. fast convergence)
- \( \rho < \) : a rational sequence \( (p_m) \) such that \( y = \sup_m p_m \) (i.e. lower approximation)
- \( \rho' \) : a rational sequence \( (p_m) \) such that \( y = \lim_m p_m \) (i.e. ultimate convergence)
- \( \rho' < \) : a rational sequence \( (p_m) \) such that \( y = \sup_m \inf p_m(n,n) \)
- \( \rho'' \) : a rational sequence \( (p_m) \) such that \( y = \lim_{m,n} p_m(n,n) \)
- \( \rho'' < \) : a rational sequence \( (p_m) \) such that \( y = \sup_m \inf_k \sup p_m(n,n,k) \)

\( \ldots \)

- \( \rho^{(d)} \) : a rational sequence \( (p_m) \) such that \( y = \lim_{n_1, n_2, \ldots, n_d} \lim \cdots \lim p_{(n_1, n_2, \ldots, n_d)} \)
- \( \rho^{(d)} < \) : a rational sequence \( (p_m) \) such that \( y = \sup_{n_1, n_2, \ldots, n_d} \inf_{n_1, n_2, \ldots, n_d} \lim p_{(n_1, n_2, \ldots, n_d)} \)

\( \ldots \)

- \( \rho^{(d)} > \) : a \( \rho^{(d)} \)-name for \( -y \).

\( \rho' \) of course coincides with the naive Cauchy-representation \( \rho_{\text{CN}} \).

Observation 4.13. These representations constitute a hierarchy

\[
\rho \preceq \rho < \preceq \rho' \preceq \rho' < \preceq \rho'' \preceq \ldots \preceq \rho^{(d)} \preceq \rho^{(d)} < \ldots
\]

which corresponds to—and is as strict as—the Arithmetical Hierarchy of real numbers in Equation 4.3: \( x \in \mathbb{R} \) is obviously \( \rho^{(d)} \)-computable (\( \rho^{(d)} < \)-computable) iff \( x \in \Delta_{d+1}(\mathbb{R}) \) (\( x \in \Sigma_{d+1}(\mathbb{R}) \)).

It follows that \( x \in \mathbb{R} \) is \( \rho^{(d)} \)-computable iff \( x \) is both \( \rho^{(d)} < \)-computable and \( \rho^{(d)} > \)-computable.

Closer examination of the proof of \textbf{Lemmas 3.2 and 3.3} reveals the above claim to hold even uniformly; that is one has, as an extension of Fact 2.61 the following

Lemma 4.14. a) For each \( d \in \mathbb{N} \), it is \( \rho^{(d)} \equiv \rho^{(d)} \land \rho^{(d)} \).

b) Let \( \rho^{(d)} < \) denote the representation encoding \( x \in \mathbb{R} \) as \( (q_n) \subseteq \mathbb{Q} \) with \( x = \lim \inf_n q_n \). Then it holds \( \rho^{(d)} \equiv \rho^{(d)} \) (\( \rho^{(d)} \leq \rho^{(d)} \) being the trivial direction).
4.2. ARITHMETICAL HIERARCHY OF REAL NUMBERS

4.2.1 Some Relations to the Effective Borel Hierarchy

The open interval \((a, b)\) belongs to \(\Sigma_1(2^\mathbb{R})\) iff \(a \in \Pi_1(\mathbb{R})\) and \(b \in \Sigma_1(\mathbb{R})\) \cite{Weih00} Example 5.1.17.2a; the closed interval \([a, b]\) belongs to \(\Pi_1(2^\mathbb{R})\) iff \(a \in \Sigma_1(\mathbb{R})\) and \(b \in \Pi_1(\mathbb{R})\) \cite{Weih00} Example 5.1.3.2b. \cite{ZhWe01} Corollary 6.6 partly generalizes this to the second level of the Arithmetical Hierarchy, namely to intervals in \(\Pi_2(2^\mathbb{R})\): the open case turns out quite surprisingly:

Fact 4.15. a) For a closed interval \([a, b]\) \(\in \Pi_2(2^\mathbb{R})\), \(a \in \Sigma_2(\mathbb{R})\) and \(b \in \Pi_2(\mathbb{R})\).

b) Every open interval \((a, b)\) \(\in \Pi_2(2^\mathbb{R})\) belongs even to \(\Sigma_1(2^\mathbb{R})\).

Claims a) and b) of the next Lemma are folklore and well-known \cite{Weih00} Lemma 5.2.6.2. They imply the above facts about intervals on the first level of the effective Borel Hierarchy. Claims c) and d) concern extensions to the second level.

Lemma 4.16. Consider the function \(\sup : 2^\mathbb{R} \to \mathbb{R} \cup \{\pm \infty\}\) (with the convention \(\sup \emptyset = -\infty\)).

a) Restricted to \(\Sigma_1(\mathbb{R})\), it is \((\delta_{\Sigma_1} \to \rho_<)\)-computable.

b) Restricted to \(\Pi_1([0, 1])\), it is \((\delta_{\Pi_1} \to \rho_>\)–computable.

c) Restricted to \(\Sigma_2([0, 1])\), it is \((\delta_{\Sigma_2} \to \rho'_<)\)-computable.

d) Restricted to \(\Sigma_1([0, 1]) \subseteq \Pi_2([0, 1])\), it is \((\delta_{\Pi_2} \to \rho'_>\)–discontinuous.

Similarly for \(\inf\) with \(\rho_<\) and \(\rho_>\) (or \(\rho'_<\) and \(\rho'_>\)) exchanged…

Question 4.17. Nonuniformly, does it hold \(\sup S \in \Pi_2(\mathbb{R})\) for every \(S \in \Pi_2(2^\mathbb{R})\)? Or at least for every \(S \in \Pi_2 \cap \Sigma_1\)?

Proof (Lemma 4.16). c) Let \(S = \bigcup_n A_n\) be given by \(\psi_\beta\)–names of closed \(A_n \subseteq [0, 1]\). For each \(n\), compute by virtue of b) a rational sequence \((q_{n,m})_m\) with \(\inf_m q_{n,m} = \sup A_n\). According to Claim 4.18 below, \((q_{n,m})\) is then a \(\rho'_<\)–name for \(\sup S = \sup_n \inf_m q_{n,m}\).

Claim 4.18. For \(S_i \subseteq \mathbb{R}\), \(i\) ranging over an arbitrary index set \(I\), it holds \(\sup(\bigcup_i S_i) = \sup \sup S_i\) and \(\inf(\bigcap_i S_i) = \inf \inf S_i\).

Moreover, it is \(\sup(\bigcap_i S_i) \leq \sup S_i\) and \(\inf(\bigcup_i S_i) \geq \inf S_i\).

It is easy to find (even topologically very simple) examples of strict inequalities in the case of intersection.

Proof. Since \(S_i \subseteq \bigcup_j S_j\), \(\sup S_i \leq \sup \bigcup_j S_i\) for each \(i\) by monotonicity; and hence \(\sup_i \sup S_i \leq \sup(\bigcup_j S_j)\). Conversely every \(x \in \bigcup_i S_i\) has \(x \leq \sup_i \sup S_i\) (because \(x \in S_i\) for some \(i\)); therefore \(\sup(\bigcup_i S_i) \leq \sup \sup S_i\).

The next lemma can be regarded as a partial converse to Lemma 4.16. Again, Claims a) and b) are folklore.

Lemma 4.19. a) The mapping \(\mathbb{R}^2 \ni (a, b) \mapsto (a, b) \in \mathcal{O}\) is \((\rho_\times \rho_\to \delta_{\Sigma_1})\)-computable.

b) The mapping \(\mathbb{R}^2 \ni (a, b) \mapsto [a, b] \in \mathcal{O}\) is \((\rho_\times \rho_\to \delta_{\Pi_1})\)-computable.

c) The mapping \(\mathbb{R}^2 \ni (a, b) \mapsto [a, b] \in \mathcal{O}\) is \((\rho'_\times \rho'_\to \delta_{\Pi_1})\)-computable.

In view of Fact 4.15 the mapping \(\mathbb{R}^2 \ni (a, b) \mapsto (a, b) \in \mathcal{O}\) cannot be \((\rho'_\times \rho'_\to \delta_{\Sigma_1})\)-computable.

Proof. b) For \(a = \sup_n q_n\) and \(b = \inf_m p_m\) output open rational intervals (exhausting) \((-\infty, q_n)\) and \((p_m, +\infty)\): these cover exactly (and hence constitute a \(\delta_{\Sigma_1}\)–name of) \((-\infty, a) \cup (b, +\infty)\); even in the degenerate cases \(a = b\) and \(a > b\).

c) For notational simplicity we consider only \(a = \sup_n q_n\) and \(b = \inf_m q_{n,m}\) as given and fix \(b = 0\). We may w.l.o.g suppose that \(\inf_m q_{n,m} < \inf_m q_{n+1,m} < a\) holds: apply Claim 4.24. According to Item a), \(\delta_{\Sigma_1}\)–compute \(U_n := (\inf_m q_{n,m}, 1/n)\) and observe that \(\bigcap_n U_n = [a, 0]\) holds.
4.2.2 Transfinite Levels

Similar to Section 4.1.1, one may extend the finite Arithmetical Hierarchy of real numbers (4.3) to transfinite levels. This is easy for the ambiguous classes by defining, in view of the first item in Theorem 4.9,

$\Delta^\alpha(\mathbb{R}) := \{ \sum_{n \in S} 2^{-n} \mid S \in \Delta^\alpha(\mathbb{N}) \}$ (4.4)

with $\Delta^\alpha(\mathbb{N})$ according to Section 3.1.3. However, an approach based on binary expansions does not work for classes $\Sigma^\alpha$. A definition based on extending the last item of Theorem 4.10 to an infinite number of “sup/inf” alternations seems more natural; but a multi-sequence $q(n_1,\ldots,n_d)$ of course makes sense only for finite $d$.

G. Barmpalias had the idea of letting $d$ be finite but vary; he defined, analogously to Equation (3.7),

$\Sigma^\omega(\mathbb{R}) := \{ \sup_d \sup_{n_1} \inf_{n_2} \sup_{n_3} \ldots \lim_{n_d} q(n_1,\ldots,n_d) \mid (q_n)_n \subseteq \mathbb{Q} \text{ computable} \}$

This approach generalizes to higher recursive ordinals and indeed yields a proper hierarchy [Barm03, Theorem 4]. For the ambiguous classes it coincides with (4.4); and, just as for the finite levels $\alpha < \omega$, it holds $\sum_{n \in S} 2^{-n} \in \Sigma^\alpha(\mathbb{R})$ if (but not only if) $S \in \Sigma^\alpha(\mathbb{N})$ [Barm03, Proposition 3].

4.3 Hypercomputing Real Functions

In addition to real numbers, [Ho99] has also studied real function computability (Definition 2.42a) relative to $\emptyset'$ and obtained, for instance, the following analogue of Fact 2.45 with non-effective uniform convergence:

**Proposition 4.20 ([Ho99, Corollary 17]).** A function $f : [0,1] \to \mathbb{R}$ is computable relative to $\emptyset'$ iff there exists a Turing–computable sequence of (degrees and coefficients of) rational polynomials $(P_m) \subseteq \mathbb{Q}[X]$ such that $\|f - P_m\| \to 0$.

Notice the similarity to Proposition 4.8.

In particular the Halting oracle, employed in this way, does not lift the continuity condition Proposition 2.44. More generally it immediately follows from the classical Fact 2.9 in combination with the relativization of Fact 2.45:

**Corollary 4.21.** A function $f : \mathbb{R} \to \mathbb{R}$ is continuous iff there exists an oracle $O \subseteq \mathbb{N}$ such that $f$ is computable relative to $O$.

Compare [Mosc80, Exercise 3D.21]...

4.3.1 Weak Function Evaluation and Continuity

The Main Theorem of Recursive Analysis (Proposition 2.44) requires any computable real function to be continuous. This raises

**Question 4.22.** Does real hypercomputation permit an effective evaluation of (certain) discontinuous functions?

In view of Corollary 4.21 the answer seems negative. However, a characteristic of real hypercomputation is that it naturally includes non-oracle ways of transcending the Church-Turing Hypothesis. Specifically, the present section investigates $(\rho^{(k)} \to \rho^{(d)})$–computable, $(\rho^{(k)} \to \rho^{(d)}_{<})$–computable, and $(\rho^{(k)}_{<} \to \rho^{(d)}_{<})$–computable real functions in the sense of Section 4.2. (Another interesting notion will be dealt with in Section 4.5.) Indeed, the representations $\rho^{(d)}$ for $d \geq 1$ and $\rho^{(d)}_{<}$ for $d \geq 0$ fail the admissibility hypothesis of Fact 2.62 and this kind of real hypercomputation does indeed open the door to ‘counterexamples’ to Proposition 2.44.
Example 4.23. Heaviside’s function (cf. Equation (2.7) and Figure 2.3) is both \((\rho_\prec \rightarrow \rho_\prec)\)-computable and \((\rho'_\prec \rightarrow \rho'_\prec)\)-computable.

Proof. Given \((q_n) \subseteq \mathbb{Q}\) with \(x = \sup_n q_n\), exploit \((\nu_\mathbb{Q} \rightarrow \nu_\mathbb{Q})\)-computability of the restriction \(h|_\mathbb{Q} : \mathbb{Q} \to \{0, 1\}\) to obtain \(p_n := h(q_n)\). Then \((p_n) \subseteq \mathbb{Q}\) has \(\sup_n p_n = h(x)\): In case \(x \leq 0\), \(q_n \leq 0\) and hence \(p_n = 0\) for all \(n\); whereas in case \(x > 0\), \(q_n > 0\) and hence \(p_n = 1\) for some \(n\).

Let \(x \in \mathbb{R}\) be given by a rational double sequence \((q_{i,j})\) with \(x = \sup_i \inf_j q_{i,j}\). By Claim 4.24 below we may w.l.o.g assume that \(\inf_j q_{i+1,j} \geq \inf_j q_{i,j}\). Now compute \(p_{i,j} := h(q_{i,j} - 2^{-i})\). Then in case \(x \leq 0\), it holds that \(\forall i:j \mid q_{i,j} \leq 2^{-i}\), i.e., \(p_{i,j} = 0\) and thus \(\sup_i \inf_j p_{i,j} = 0 = h(x)\). Similarly, in case \(x > 0\), there is some \(i_0\) such that \(\inf_j q_{i_0,j} > x/2\) and thus \(\inf_j q_{i,j} > x/2\) for all \(i \geq i_0\). For \(i \geq i_0\) with \(2^{-i} \leq x/2\), it follows that \(p_{i,j} = 1 \forall j\) and therefore \(\sup_i \inf_j p_{i,j} = 1 = h(x)\).

Claim 4.24. For real double-sequence \((q_{n,m})\) let \(\tilde{q}_{n,m} := \min\{q_{n,1}, \ldots, q_{n,m}\} + 2^{-m}\) and \(\check{q}_{n,m} := \max\{q_{1,m}, \ldots, q_{n,m}\} - 2^{-n}\). Then it holds

\[
\begin{align*}
\text{a)} & \quad \tilde{q}_{n,m+1} \leq \check{q}_{n,m} - 2^{-m-1} \\
\text{b)} & \quad \inf_m \tilde{q}_{n+1,m} \geq \inf_m \check{q}_{n+1,m} + 2^{-n-1} \\
\text{c)} & \quad \sup_n \inf_m \check{q}_{n,m} = \sup_n \inf_m q_{n,m}.
\end{align*}
\]

Proof. See the proof of [ZhWe01, Lemma 3.1].

So real function hypercomputation based on representations weaker than \(\rho\) does allow effective evaluation of some discontinuous functions. On the other hand, they still impose well-known restrictions (recall Section 2.2.3):

Fact 4.25. Consider \(f : \mathbb{R} \to \mathbb{R}\).

\[
\begin{align*}
\text{a)} & \quad \text{If } f \text{ is } (\rho \to \rho)\text{-computable, then it is continuous.} \\
\text{b)} & \quad \text{If } f \text{ is } (\rho \to \rho_\prec)\text{-computable, then it is lower semi-continuous.} \\
\text{c)} & \quad \text{If } f \text{ is } (\rho_\prec \to \rho_\prec)\text{-computable, then it is nondecreasing.} \\
\text{d)} & \quad \text{If } f \text{ is } (\rho' \to \rho')\text{-computable, then it is continuous.}
\end{align*}
\]

The claims remain valid under oracle-supported computation.

Claim a) is the Main Theorem; the others are proven similarly on the basis of discontinuity arguments. For b) see e.g. [WeZh00]. Establishing d) in [BrHe02, Section 6] caused some surprise. We briefly sketch the proofs as a preparation for those of Theorem 4.26 below.

Proof. a) Cf. the proof of Proposition 2.44

b) As in a), we prove \((\rho \to \rho_\prec)\)-uncomputability of the flipped Heaviside Function \(\overline{h} = 1 - h\) as a prototype lacking lower semi-continuity. Consider the \(\rho\)-name \(q_n := 2^{-n}\) for \(x = 0\) which the hypothetical Type-2 Machine transforms into a \(\rho_\prec\)-name for \(y = \overline{h}(x) = 1\), that is, a sequence \((p_m) \subseteq \mathbb{Q}\) with \(\sup_m p_m = y\); in particular, for some \(M \in \mathbb{N}\), \(p_m \geq \frac{2}{3}\) gets output, having read only \((q_n)_{n \leq N}\) for some \(N \in \mathbb{N}\). The latter finite segment is also the initial part of a valid \(\rho\)-name for \(\tilde{x} = q_N > 0\), whereas \((p_m)_{m \leq M}\) has sup \(\geq \frac{2}{3}\) and thus is not the initial part of a valid \(\rho_\prec\)-name for \(\check{y} = \overline{h}(\tilde{x}) = 0\); contradiction.

This proof for the case \(\overline{h}\) carries over to an arbitrary \(f : \mathbb{R} \to \mathbb{R}\) just like in a), that is, by replacing \(q_n = 2^{-n}\) with rational approximations to a general sequence \(x_n \in \mathbb{R}\) witnessing the violation of lower semi-continuity of \(f\) in that \(f(\lim_n x_n) > \lim inf_n f(x_n)\).
c) As in a) and b), we treat for notational simplicity the case of \( f : \mathbb{R} \to \mathbb{R} \) violating monotonicity in that \( f(0) = 1 \) and \( f(1) = 0 \); the general case can again be handled similarly. Feed the \( \rho \)-name \((q_n) = (0, 0, \ldots)\) for \( x = 0 \) into a machine which, according to the presumption, produces a sequence \((p_m) \subseteq \mathbb{Q}\) with \( \sup p_m = 1 \) and in particular \( p_M \geq \frac{2}{3} \) for some \( M \in \mathbb{N} \).

Up to output of \( p_M \), only \((q_n)_{n \in \mathbb{N}}\) has been read for some \( N \in \mathbb{N} \). Now consider the rational sequence \((\tilde{q}_n)\) consisting of \( N \) 0s followed by an infinity of 1s, that is, a valid \( \rho \)-name for \( \tilde{x} = 1 \). This new input will cause the machine to output a sequence \((\tilde{p}_m) \subseteq \mathbb{Q}\) coinciding with \((p_m)\) for \( m \leq M \); in particular, \( \tilde{p}_M \geq \frac{2}{3} \), contradicting the hypothesis that \((\tilde{p}_m)\) is supposed to satisfy \( \sup p_m = f(\tilde{x}) = 0 \).

d) Suppose that, in spite of its discontinuity at \( x = 0 \), \( \mathcal{H} \) be \((\rho' \to \rho')\)-computable by some Type-2 Machine \( \mathcal{M} \).

Consider the sequence \( q^{(1)} := (q_n^{(1)} \subseteq \mathbb{Q}, q_n^{(1)} := 1 \) which is by definition a valid \( \rho' \)-name for \( 1 =: x^{(1)} = \lim_n q_n^{(1)} \). Thus upon input of \( q^{(1)} \), \( \mathcal{M} \) will generate a sequence \( p^{(1)} \subseteq \mathbb{Q} \) as a \( \rho' \)-name for \( y^{(1)} = \mathcal{H}(x^{(1)}) = 0 \), that is, satisfying \( \lim_m p_m^{(1)} = 0 \); in particular, \( p_m^{(1)} \leq \frac{1}{3} \) for some \( m_1 \in \mathbb{N} \). Up to this output, \( \mathcal{M} \) has read only a finite initial part of the input \( q^{(1)} \), say, up to \( n \leq m_1 \).

Next consider the sequence \( q^{(2)} \subseteq \mathbb{Q} \) defined by \( q_n^{(2)} := 1 \) for \( n \leq m_1 \) and \( q_n^{(2)} := \frac{1}{2} \) for \( n > m_1 \); a valid \( \rho' \)-name for \( x^{(2)} = \frac{1}{2} \) which \( \mathcal{M} \) by presumption transforms into a sequence \( p^{(2)} \subseteq \mathbb{Q} \) with \( \lim_m p_m^{(2)} = y^{(2)} = \mathcal{H}(x^{(2)}) = 0 \); in particular, \( q_n^{(2)} \leq \frac{1}{3} \) for some \( m_2 > m_1 \). However, due to \( \mathcal{M} \)'s deterministic behavior and since \( q^{(1)} \) and \( q^{(2)} \) initially coincide, it still holds \( p_m^{(2)} \leq \frac{1}{3} \).

Now by repeating the above argument we obtain a sequence of sequences \( q^{(k)} \subseteq \mathbb{Q} \), each constant for \( n \geq n_k \) of value (and thus a valid \( \rho' \)-name for) \( x^{(k)} = 2^{-k+1} \) and transformed by \( \mathcal{M} \) into a sequence \( p^{(k)} \subseteq \mathbb{Q} \) satisfying \( p_m^{(k)} \leq \frac{1}{3} \) for \( i = 1, \ldots, k \) with strictly increasing \((n_k), (m_k) \subseteq \mathbb{N} \). The ultimate sequence \( q^{(\omega)} \subseteq \mathbb{Q} \), well-defined by \( q_n^{(\omega)} := q_n^{(k)} \) for \( n \leq n_k \) (and in fact the limit of the sequence of sequences \( q^{(k)} \) with respect to Baire’s Topology), therefore converges to \((\text{and is therefore a valid} \rho' \text{-name for}) x^{(\omega)} = 0 \); and it is mapped by \( \mathcal{M} \) to a sequence \( q^{(\omega)} \subseteq \mathbb{Q} \) satisfying \( q_m^{(\omega)} \leq \frac{1}{3} \) for infinitely many \( m \), contradicting the hypothesis that a valid \( \rho' \)-name for \( y^{(\omega)} = \mathcal{H}(x^{(\omega)}) = 1 \) should have \( \lim_m = 1 \).

Being of information-theoretic nature, the above arguments obviously relativize. \( \square \)

The main result of the present section is an extension of Fact \( 4.25 \) to one level up on the hierarchy of real representations from Definition \( 4.12 \). This might not be as surprising any more as Fact \( 4.25 \) in \( \text{BrHe02} \); nevertheless, even this additional step makes proofs significantly more involved.

**Theorem 4.26.** Consider \( f : \mathbb{R} \to \mathbb{R} \).

a) If \( f \) is \((\rho' \to \rho'_c)\)-computable, then it is lower semi-continuous.

b) If \( f \) is \((\rho'_c \to \rho'_c)\)-computable, then it is nondecreasing.

c) If \( f \) is \((\rho'' \to \rho'')\)-computable, then it is continuous.

The claims remain valid under oracle-supported computation.

We point out that the proofs of Fact \( 4.25 \) proceed by constructing an input for which a presumed machine \( \mathcal{M} \) fails to produce the correct output. They differ, however, in the ‘length’ of these constructions: for Claims a) to c), the counter-example inputs are obtained by running \( \mathcal{M} \) for a finite number of steps on a single, fixed argument; whereas in the proof of Claim d), \( \mathcal{M} \) is repeatedly started on an adaptively extended sequence of arguments. The latter argument may thus be considered as of length \( \omega \), the first infinite ordinal. Our proof of Theorem 4.26c) will be even longer and is therefore put into the following subsubsection.
4.3. HYPERCOMPUTING REAL FUNCTIONS

Proof of Theorem 4.26

As in the proof of Fact 4.25, we treat the special case of the flipped Heaviside Function $\overline{H}$ for reasons of notational convenience and clarity of presentation; the arguments can be immediately extended to the general case.

Claim 4.27. $\overline{H} : \mathbb{R} \to \mathbb{R}$ is not $(\rho' \to \rho'_\omega)$-computable.

Proof. Suppose a Type-2 Machine $\mathcal{M}$ $(\rho' \to \rho'_\omega)$-computes $\overline{H}$. In particular, upon input of $x^{(1)} = 1$ in the form of the sequence $q^{(1)} = (q_n^{(1)})$ with $q_n^{(1)} := 1$, $\mathcal{M}$ will output a rational double sequence $p^{(1)} = (p_{k,\ell}^{(1)})$ with $0 = y^{(1)} := \overline{H}(x^{(1)}) = \sup_{k,\ell} p_{k,\ell}^{(1)}$. Observe that $p_{n,\ell_1}^{(1)} \leq \frac{1}{3}$ for some $\ell_1$. When $p_{n,\ell_1}^{(1)}$ is written, $\mathcal{M}$ has only read a finite part of $(q_n^{(1)})$, say, up to $n_1$.

Now consider $x^{(2)} := \frac{1}{2}$, given by way of the sequence $q^{(2)}$ with $q_n^{(2)} := 1$ for $n < n_1$ and $q_n^{(2)} := \frac{1}{2}$ for $n \geq n_1$. Then, too, $\mathcal{M}$ will output a double sequence $p^{(2)}$ with $0 = y^{(2)} = \sup_{k,\ell} p_{k,\ell}^{(2)}$. Observe that, similarly, some $p_{n,\ell_2}^{(2)} \leq \frac{1}{3}$ is output when only a finite part of $(q_n^{(2)})$, say, up to $n_2$, has been read. Moreover, as $q^{(1)}$ and $q^{(2)}$ coincide up to $n_1$ and since $\mathcal{M}$ operates deterministically, $p_{n,\ell_2}^{(2)} = p_{n,\ell_1}^{(1)} \leq \frac{1}{3}$.

Figure 4.3: Illustration of the iterative construction employed in the proof of Claim 4.27

Continuing this process with $x^{(k)} := 2^{-k+1}$ for $k = 3, 4, \ldots$ as indicated in Figure 4.3, eventually yields a rational sequence $q^{(\omega)}$ with $\lim_n q_n^{(\omega)} := x^{(\omega)} = 0$, upon input of which $\mathcal{M}$ outputs a double sequence $p^{(\omega)}$ such that $p_{k,\ell}^{(\omega)} \leq \frac{1}{3}$ for all $k = 1, 2, \ldots$. In particular, $y^{(\omega)} := \sup_{k,\ell} p_{k,\ell}^{(\omega)} \leq \frac{1}{3}$ whereas $\overline{H}(x^{(\omega)}) = 1$ : contradiction.

Note that the above proof involves one-dimensionally indexed sequences $(q_n)$ for input and two-dimensionally indexed ones $(p_{k,\ell})$ for output. We now proceed a step further in proof difficulty, namely involving two-dimensional indices for both input and output in order to establish Item b).

Claim 4.28. Let $f : \mathbb{R} \to \mathbb{R}$ violate monotonicity in that $f(0) = 1$ and $f(1) = 0$. Then $f$ is not $(\rho'_\omega \to \rho'_\omega)$-computable.

Proof. We construct a $\rho'_\omega$-name for $x = 0$ from an iteratively defined sequence of initial segments of $\rho'_\omega$-names for $x = 1$:

Start with $q_{i,j}^{(1)} := 1$ for all $i, j$. Then $q^{(1)} = (q_{i,j}^{(1)})$ is obviously a $\rho'_\omega$-name for $x = 1$ and thus yields by presumption, upon input to $\mathcal{M}$, a $\rho'_\omega$-name $p_{k,\ell}^{(1)}$ for $f(1) = 0$, that is, with $0 = \sup_{k,\ell} p_{k,\ell}^{(1)}$. In particular, $p_{n,\ell_1}^{(1)} \leq \frac{1}{3}$ for some $\ell_1$.

Until output of $p_{n,\ell_1}^{(1)}$, $\mathcal{M}$ has read only finitely many entries of $q^{(1)}$: say, up to $i_1$ and $j_1$, that is, covered in Figure 4.4 by the light gray rectangle. Now consider $q^{(2)}$ defined as in this figure: Since $\inf_{i < i_1} q_{i,j}^{(2)} = 0$ for $i \leq i_1$ and $\inf_{i > i_1} q_{i,j}^{(2)} = 1$ for $i > i_1$, $\sup_{i,\ell} q_{i,j}^{(2)} = 1$, that is, this is again a valid $\rho'_\omega$-name for $x = 1$; and again, $\mathcal{M}$ will, according to the presumption, convert $q^{(2)}$ into
a \( \rho' \)-name \( p^{(2)} \) for \( f(1) = 0 \). In particular, \( p^{(2)}_{j,\ell_2} \leq \frac{1}{3} \) for some \( \ell_2 \); and, being a deterministic machine, \( \mathcal{M} \)’s operation on the initial part (dark gray) on which input \( q^{(1)} \) will first have generated the same initial output, namely \( p^{(2)}_{1,\ell_1} = p^{(1)}_{1,\ell_1} \leq \frac{1}{2} \).

Again, until output of \( p^{(2)}_{2,\ell_2} \), \( \mathcal{M} \) has read only a finite part of \( q^{(2)} \) of, say, up to \( \ell_2 > \ell_1 \) (light gray).

By now considering input \( q^{(3)} \) with \( \inf_j q^{(3)}_{i,j} = 0 \) for \( i \leq \ell_2 \) as in Figure 4.4 we arrive at \( p^{(3)} \) and \( \ell_3 \) with \( p^{(3)}_{1,\ell_3}, p^{(3)}_{2,\ell_3}, p^{(3)}_{3,\ell_3} \leq \frac{1}{3} \); and so on with \( i_3, q^{(4)}, p^{(4)}, \ell_4, i_4, \ldots \).

Finally observe that continuing these arguments eventually leads to a rational double sequence \( q^{(\omega)} = (q^{(\omega)}_{i,j}) \) which has \( \inf_j q^{(\omega)}_{i,j} = 0 \) for \( i \leq i_\omega = \omega \)—and is therefore a valid \( \rho' \)-name for \( x = 0 \) (rather than \( x = 1 \))—but gets mapped by \( \mathcal{M} \) to \( p^{(\omega)} = (p^{(\omega)}_{k,\ell}) \) with \( \inf_{k,\ell} p^{(\omega)}_{k,\ell} \leq p^{(\omega)}_{k,\ell_k} \leq \frac{1}{3} \) for all \( k \).

Since \( f(0) = 1 \), this contradicts our presumption that \( \mathcal{M} \) maps \( \rho' \)-names for \( x \) to \( \rho' \)-names for \( f(x) \).

The above proofs involving \( \rho' \) and \( \rho'_\varsigma \) proceeded by constructing an infinite sequence of inputs \( q^{(1)}, q^{(2)}, \ldots, q^{(\omega)} \) (each possibly a multi-indexed sequence of its own). In order to finally assert Claim c) involving \( \rho'' \), we will extend this method from length \( \omega \), the first infinite ordinal, to an even longer (but still countable) one.

Claim 4.29. \( \mathcal{T} : \mathbb{R} \to \mathbb{R} \) is not \((\rho'' \to \rho'')\)-computable.

Proof. Outwit a Type-2 Machine \( \mathcal{M} \), presumed to realize this computation, as follows:

i) Take \( q^{(1)} \) to be the constant double sequence 1, i.e., \( q^{(1)}_{i,j} := 1 \) for all \( i, j \). Being a \( \rho'' \)-name for 1, it is by presumption mapped to a \( \rho'' \)-name \( p^{(1)} \) for \( \mathcal{T}(1) = 0 \), that is, satisfying \( \lim_k \lim_{\ell} p^{(1)}_{k,\ell} = 0 \). In particular, almost every column \( \#k \) contains an entry \( \#\ell \) with \( p^{(1)}_{k,\ell} \leq \frac{1}{3} \).

Until output of the first such \( p^{(1)}_{k_1,\ell_1} \), \( \mathcal{M} \) has read only a finite part of \( q^{(1)} \)—say, up to \( i_1, j_1 \).

ii) Observe that this Argument i) equally applies to the scaled input sequence \( 2^{-m} \cdot q^{(1)} \) for any \( m \). So define \( q^{(2)}_{i,j} := q^{(1)}_{i,j} \) for \( j \leq j_1 \) (i.e., inherit the initial part of \( q^{(1)} \)) and \( q^{(2)}_{i,j} := \frac{1}{3} \) for \( j > j_1 \). Now upon input of this \( q^{(2)} \), \( \mathcal{M} \) will output \( p^{(2)} \) with, again, infinitely many \( p^{(2)}_{k,\ell} \leq \frac{1}{3} \), the first one—\((k_2, \ell_2)\), say—after having read \( q^{(2)} \) only up to some \((i_2, j_2)\). Furthermore, \( \mathcal{M} \)’s determinism implies \( p^{(2)}_{k_1,\ell_1} = p^{(1)}_{k_1,\ell_1} \leq \frac{1}{3} \).

By repeating for \( m = 2, 3, \ldots \), we eventually obtain—similarly to the proof of Claim 4.28—an input sequence \( q^{(\omega)} \) with \( q^{(\omega)}_{i,j} \) with \( \lim_k \lim_{\ell} q^{(\omega)}_{i,j} = 0 \), that is, a valid \( \rho'' \)-name for \( x = 0 \).
4.3. HYPERCOMPUTING REAL FUNCTIONS

Figure 4.5: The first infinitely long iterative construction employed in the proof of Claim 4.29

(rather than 1). This is mapped by $\mathcal{M}$ to $p^{(\omega)}$ with $p^{(\omega)}_{k_m,\ell_m} \leq \frac{1}{3}$ for all $m$. On the other hand, $p^{(\omega)}$ is by presumption a $\rho''$–name for $\bar{h}(0) = 1$. Therefore, there are infinitely many $m$ with $p^{(\omega)}_{m,\ell} \geq \frac{2}{3}$ for some $\ell > \ell_m$ and $p^{(\omega)}_{m,\ell_m} \leq \frac{1}{3}$; see the gray columns in the right part of Figure 4.5.

iii) Since this gives no contradiction yet, we proceed by considering the first such column $m$ containing an entry $\leq \frac{1}{3}$ as well as an entry $\geq \frac{2}{3}$. Take the initial part of the input $q^{(\omega)}$ — up to $(i_\omega, j_\omega)$, say, depicted in gray in the left part of Figure 4.6 — that $\mathcal{M}$ has read until output of both of them; extend it with $\frac{1}{2}s$ in top direction and with $1$s to the right. Feed this $\rho''$–name for $x = 1$ into $\mathcal{M}$ until output of an entry $p_{k,\ell} \leq \frac{1}{3}$ in some column $k$ beyond $m$. Then repeat extending to the right with $1$s replaced by $\frac{1}{2}s$ for a second entry $p_{k,\ell} \leq \frac{1}{3}$.

More generally, proceed similarly as in ii) and extend $d^{(\omega)}_{i_\omega, j_\omega}$ to the right with some $\rho''$–name $q^{(\omega)}$ for $x = 0$ in such a way as to obtain another column $m'$ with both entries $\leq \frac{1}{3}$ and $\geq \frac{2}{3}$; see the middle part of Figure 4.6. Again, $\mathcal{M}$ outputs the latter two entries having read only a finite part; say, up to $(i'_\omega, j'_\omega)$.

Now extend this part, too, with $\frac{1}{2}$ in top direction and with another $q''^{(\omega)}$ obtained, again, as in ii) for a third column $m''$ with both entries $\leq \frac{1}{3}$ and $\geq \frac{2}{3}$; and so on.
CHAPTER 4. ARITHMETICAL HIERARCHIES IN RECURSIVE ANALYSIS

This eventually leads to an input \( q^{(2\omega)} \) which, due to the extensions to the top, represents a \( \rho'' \)-name for \( x = \frac{1}{2} \) and is thus mapped, according to the presumption, to a \( \rho'' \)-name \( p^{(2\omega)} \) for \( \tilde{h}(\frac{1}{2}) = 0 \). In particular, almost every column of \( p^{(2\omega)} \) has almost every entry \( \leq \frac{1}{3} \) while maintaining infinitely many columns with preceding entries \( \leq \frac{1}{3} \) and \( \geq \frac{2}{3} \); see the right part of Figure 4.6. This ensures the existence of infinitely many columns in \( p^{(2\omega)} \) containing \( \leq \frac{1}{3} \), \( \geq \frac{2}{3} \), and \( \leq \frac{1}{3} \) in order. And again, even just a finite initial part of \( q^{(2\omega)} \) up to some \((i_{2\omega}, j_{2\omega})\) gives rise to the first such triple.

iv) Notice that the arguments in iii) similarly yield the existence of an appropriate, scaled counter-part \( \frac{1}{2} q^{(2\omega)} \) of \( q^{(2\omega)} \), of some \( \frac{1}{4} q^{(2\omega)} \), and so on, all leading to output containing infinitely many columns with alternating triples as above. We now construct input \( q^{(3\omega)} \) leading to output \( p^{(3\omega)} \) containing an infinity of columns, each with four entries \( \leq \frac{1}{3} \), \( \geq \frac{2}{3} \), \( \leq \frac{1}{3} \), and \( \geq \frac{2}{3} \). Take the initial part of \( q^{(2\omega)} \) leading to output of the first column with alternating triple in the above sense; then extend it with the initial part of the scaled version \( \frac{1}{2} q^{(2\omega)} \) leading to another column with such a triple; and so on. Observing that, due to the scaling, the \( q^{(3\omega)} \) obtained in this way represents a \( \rho'' \)-name for \( x = 0 \), and almost every column of the output \( p^{(3\omega)} \) representing \( \tilde{h}(0) = 1 \) contains entries \( \geq \frac{2}{3} \) in addition to the infinitely many columns with triples as above; see the left part of Figure 4.7.

![Figure 4.7](image)

Figure 4.7: Third, fourth, and fifth infinitely long iterative construction employed in the proof of Claim 4.29

v) Our next step is a \( \rho'' \)-name \( q^{(4\omega)} \) for \( x = \frac{1}{4} \) giving rise to \( p^{(4\omega)} \) with infinitely many columns containing alternating quintuples. This is obtained by repeating the arguments in iv) to obtain initial segments of (variants of) \( q^{(3\omega)} \), stacking them horizontally—in order to obtain an infinity of columns with alternating quadruples—while extending in top direction with \( \frac{1}{4} \); see the middle part of Figure 4.7. This forces \( M \) to output a \( \rho'' \)-name \( q^{(4\omega)} \) for \( \tilde{h}(\frac{1}{4}) = 0 \) having in almost every column almost every entry \( \leq \frac{1}{4} \), thereby extending the alternating quadruples to quintuples.

vi) Noticing that the vertical extension in v) was similar to step iii), we now take a step similar to iv) based on horizontally stacked initial parts of scaled counterparts of \( q^{(4\omega)} \) in order to obtain a \( \rho'' \)-name \( q^{(5\omega)} \) for \( x = 0 \) which \( M \) maps to some \( p^{(5\omega)} \) containing infinitely many alternating six-tuples.

Then again construct a \( \rho'' \)-name \( q^{(6\omega)} \) for \( x = \frac{1}{8} \) by horizontally stacking initial segments of (variants of) \( q^{(5\omega)} \) while extending them vertically with \( \frac{1}{8} \) and so on.
4.3. HYPERCOMPUTING REAL FUNCTIONS

Now for the bottom line: By proceeding with the above construction, one eventually obtains a rational double sequence \( q^{(\omega^2)} \) with \( \lim_k q_{1,k}^{(\omega^2)} = 0 \) for all \( i \) — that is, a \( \rho'' \)-name for \( x = 0 \) — mapped by \( M \) to some \( p^{(\omega^2)} \) containing (infinitely many) columns \( \# k \) with infinitely many alternating entries \( \leq \frac{1}{i} \) and \( \geq \frac{2}{i} \) — contradicting the hypothesis that, for \( \rho'' \)-names \( p = (p_k, x) \), \( \lim_k p_k, x \) is required to exist for every \( k \). \( \square \)

4.3.2 Hierarchy of Weakly Evaluable Functions

For every \((\alpha \rightarrow \beta)\)-computable function \( f : A \rightarrow B \), one may obviously replace representation \( \alpha \) for \( A \) by a stronger one and \( \beta \) for \( B \) by a weaker one while yet maintaining the computability of \( f \):

\[
    f \ (\alpha \rightarrow \beta)\text{-computable} \land \ \alpha \leq \alpha' \land \ \beta \leq \beta' \implies f \ (\alpha' \rightarrow \beta')\text{-computable}
\]

However, if both \( \alpha \) and \( \beta \) are made, say, weaker then the \((\alpha \rightarrow \beta)\)-computability of \( f \) may in general be violated. For \( \alpha = \beta = \rho \leq t \), though, we have seen in Example 4.23 that the implication "\( (\rho \leq t \rightarrow \rho \leq t) \Rightarrow (\rho' \leq t \rightarrow \rho' \leq t') \)" does hold at least for the case of \( f \) being Heaviside’s function. As the following result shows, it holds in fact for every \( f \):

**Theorem 4.30.** Consider \( f : \mathbb{R} \rightarrow \mathbb{R} \).

a) If \( f \) is \((\rho \rightarrow \rho)\)-computable, then it is also \((\rho' \rightarrow \rho')\)-computable.

b) If \( f \) is \((\rho \rightarrow \rho')\)-computable, then it is also \((\rho' \rightarrow \rho')\)-computable.

c) If \( f \) is \((\rho \leq t \rightarrow \rho')\)-computable, then it is also \((\rho' \leq t \rightarrow \rho')\)-computable.

d) If \( f \) is \((\rho' \rightarrow \rho')\)-computable, then it is also \((\rho'' \rightarrow \rho'')\)-computable.

e) If \( f \) is \((\rho'' \rightarrow \rho'')\)-computable, then it is also \((\rho''' \rightarrow \rho''')\)-computable.

The claims remain valid under oracle-supported computation.

This shows for example that, for \( d = 0, 1, \ldots \), the classes of \((\rho^{(d)} \rightarrow \rho^{(d)})\)-computable real functions \( f : \mathbb{R} \rightarrow \mathbb{R} \) form a hierarchy. According to Equation 4.3, this hierarchy is strict, as can be seen from the constant functions \( f \equiv c \) with \( c \in \Delta_{d+1} \).

As another consequence we conclude that the non-reducibility claims made in Observation 4.13 do not only mean uncomputability, but in fact discontinuity, of the reductions (recall Definition 2.36v):

**Corollary 4.31.** It holds

\[
    \rho \equiv \rho_\leq \rho_\leq t \leq t' \rho_\leq \rho_\leq t \leq t' \rho_\leq \rho'' .
\]

**Proof.** The positive claims were already contained in Observation 4.13. For a negative claim like "\( \rho_\leq \leq t', \rho'' \)”, suppose the contrary. Then according to Corollary 4.21 with the help of some appropriate oracle \( O \), one can convert \( \rho_\leq \)-names to \( \rho'' \)-names. As Heaviside’s function \( h \) is \((\rho' \rightarrow \rho')\)-computable by Example 4.23 and Theorem 4.30, composition with the presumed conversion implies \((\rho'' \rightarrow \rho'')\)-computability of \( h \) relative to \( O \) — contradicting Theorem 4.26. \( \square \)

**Proof (Theorem 4.30d).** Let \( f \) be \((\rho' \rightarrow \rho')\)-computable and \( x \) given by a \( \rho'' \)-name, that is, a rational sequence \( q = (q_n) \) with \( x = \lim_i \lim_m q_{i,j} \). For each \( i \), compute, according to the hypothesis, from the \( \rho' \)-name \( q_{i,j} = (q_{i,j})_j \) of \( x_i := \lim_j q_{i,j} \) a \( \rho' \)-name of \( f(x_i) \), that is, a sequence \( p = p_{(i,j)} = (p_{(i,j)})_j \) with \( f(x_i) = \lim_j p_{(i,j)} \). The continuity of \( f \) due to Fact 4.25c ensures

\[
    \lim_i \lim_j p_{(i,j)} = \lim_i f(x_i) = f(\lim_i x_i) = f(\lim_j \lim q_{i,j}) = f(x)
\]

this sequence \( p \) is a \( \rho'' \)-name for \( y = f(x) \). \( \square \)
Where the last proof exploited Fact 4.25a), the next one relies on Theorem 4.26c):

Proof (Theorem 4.30b). A \( \rho'' \)-name for \( x \in \mathbb{R} \) is a rational sequence \( a = (q_n) \) with \( x = \lim_i \lim_j \lim_k q_{(i,j,k)} \).

For each \( i \), exploit \((\rho'' \rightarrow \rho'')\)-computability of \( f \) to obtain, from the \( \rho'' \)-name \( q_{(i,\ldots)} \) of \( x_i := \lim_j \lim_k q_{(i,j,k)} \in \mathbb{R} \), a sequence \( p_{(i,\ldots)} \) with \( \lim_i \lim_k p_{(i,j,k)} \) as \( \rho'' \)-name of \( f(x_i) \). Similarly to case d), this sequence \( p \) constitutes a \( \rho'' \)-name for \( y = f(x) \) by continuity of \( f \) due to Theorem 4.26c). \( \square \)

Proof (Theorem 4.30e). Let \( f \) be \((\rho \rightarrow \rho')\)-computable. Its \((\rho' \rightarrow \rho')\)-computability is established as follows: Given \((q_n) \subseteq \mathbb{Q} \) with \( x = \lim_n q_n \), apply the assumption to evaluate \( f(q_n) \) for each \( n \) up to error \( 2^{-n} \); that is, obtain \( p_n \in \mathbb{Q} \) with \( |p_n - f(q_n)| < 2^{-n} \). Since \( f \) is continuous by Fact 4.25a), it follows that \( f(x) = \lim_n f(q_n) = \lim_n p_n \) so that \( (p_n) \) is a \( \rho' \)-name for \( y = f(x) \).

It is worth noting that the latter proof in fact works uniformly in \( f \), i.e., we have

Scholium 4.32. The apply operator \( C(\mathbb{R}) \times \mathbb{R} \ni (f,x) \mapsto f(x) \) is \((\rho \rightarrow \rho \times \rho' \rightarrow \rho')\)-computable.}

Similarly, Theorem 4.30b) follows from Lemma 4.33 below together with the observation that every \((\rho \rightarrow \rho)\)-computable \( f \) has a computable \((\rho \rightarrow \rho)\)-name \( \text{[WeZh00, COROLLARY 5.1(2) and THEOREM 3.7]} \); here, \([\rho \rightarrow \rho] \) denotes a natural representation for the space \( \text{LSC}(\mathbb{R}) \) of lower semi-continuous functions \( f : \mathbb{R} \rightarrow \mathbb{R} \) considered in \[\text{[WeZh00]. Specifically, a \((\rho \rightarrow \rho)\)-name for a \( f \) is an enumeration of all rational triples \((a,b,c)\) such that \( c < \min f[a,b] \)—the latter making sense as a lower semi-continuous function attains its minimum (though not necessarily its maximum) on any compact set. \( [\rho \rightarrow \rho] \) indeed is a representation for \( \text{LSC}(\mathbb{R}) \) because different lower semi-continuous functions give rise to different such collections \( \{(a,b,c) \in \mathbb{Q}^3 : \ldots\}; cf. \text{[WeZh00, Lemma 3.3].} \)

Lemma 4.33. \( \text{LSC}(\mathbb{R}) \times \mathbb{R} \ni (f,x) \mapsto f(x) \) is \((\rho \rightarrow \rho \rightarrow \rho \times \rho' \rightarrow \rho')\)-computable.

Proof. Let \((a_k,b_k,c_k)_n\) denote the given \([\rho \rightarrow \rho] \)-name of \( f \in \text{LSC}(\mathbb{R}) \) and \((q_n)_n\) the \( \rho' \)-name for \( x \in \mathbb{R} \). Our goal is to \( \rho' \)-compute \( y := f(x) \). Define the sequence \( p = (p_m)_m \subseteq \mathbb{Q} \cup \{+\infty\} \) by

\[
p_{(k,\ell,n)} := \begin{cases} 
\max \{ c_m : m \leq k \land [a_m,b_m] \supseteq [a_k,b_k] \} & \text{if } q_n \in (a_k,b_k) \land |b_k - a_k| = 2^{-\ell} \\
+\infty & \text{otherwise}
\end{cases}
\] (4.5)

From the given information, one can obviously compute \( p \). Moreover, this sequence satisfies

\[ \liminf p \geq y; \]

Let \( \epsilon > 0 \) be arbitrary. Since \( f \) is lower semi-continuous, its pre-image \( f^{-1}[ (y - \epsilon, \infty) ] \ni x \) is an open set and therefore contains an entire ball around \( x \). In fact, the center of this ball may be chosen as rational and its diameter of the form \( 2^{-L} \) for some \( L \in \mathbb{N} \); formally (see Figure 4.8):

\[ \exists K, L, K' \in \mathbb{N} : x \in (a_{K'}, b_{K'}) \subseteq [a_K, b_K] \subseteq f^{-1}[ (y - \epsilon, \infty) ] \land |b_K - a_K| = 2^{-L} \land a_{K'} = a_K + \frac{3}{2} \cdot 2^{-L-2} \land b_{K'} = b_K - \frac{3}{2} \cdot 2^{-L-2} \] (4.6)

where we have exploited the fact that every rational pair \((a,b)\) occurs in the list representing the \([\rho \rightarrow \rho] \)-name. Moreover, as it consists of all rational triples \((a,b,c)\) with \( c < \min f[[a,b]] \),

\[ \exists M \geq K: [a_K,b_K] = [a_M,b_M] \land c_M \geq \min f[[a_M,b_M]] - \epsilon \tag{*} \] (4.7)

where (*) is a consequence of \([a_K,b_K] \subseteq f^{-1}[ (y - \epsilon, \infty) ] \) in Equation (4.6). Finally,

\[ \lim n q_n = x \in (a_{K'}, b_{K'}) \Rightarrow \exists N: \forall n \geq N: \ q_n \in (a_{K'}, b_{K'}) . \tag{4.8} \]

So putting things together, for each \( n \geq N, \ell \geq L', \) and \( k \geq M \), either we have \( p_{(k,\ell,n)} = +\infty \geq y - 2\epsilon \); or we are in the first case of Equation (4.5), thus
4.3. HYPERCOMPUTING REAL FUNCTIONS

- \( q_n \in (a_k, b_k) \) with \( |b_k - a_k| \leq 2^{-\ell} \)
- \( q_n \in (a_K, b_K) \) by Equation \(4.8\)
- hence \( [a_k, b_k] \subseteq [a_K, b_K] \) by Equation \(4.6\) due to \( \ell \geq L' \); cf. Figure \(4.8\)
- So \( [a_k, b_k] \subseteq [a_M, b_M] \) by Equation \(4.7\)
- implying \( p_{(k,\ell,n)} \geq c_M \geq y - 2\epsilon \) by Equations \(4.9\) and \(4.10\) since \( k \geq M \).

Summarizing, it holds \( p_{(k,\ell,n)} \geq y - 2\epsilon \) for all \( (k,\ell,n) \in \mathbb{N}^3 \) not belonging to the finite set \( \{0,1,\ldots,N-1\} \times \{0,1,\ldots,L'-1\} \times \{0,1,\ldots,M-1\} \) of exceptions. Consequently, \( \liminf p \geq y - 2\epsilon \); even \( \liminf p \geq y \) because \( \epsilon > 0 \) was arbitrary.

![Figure 4.8: Nesting of some rational intervals of dyadic length contained in \( f^{-1}([y-\epsilon,\infty)) \).](image)

The parameters are chosen in such a way that, whenever \((a_k, b_k)\) meets some other \((a'_k, b'_k)\) of length \( |b_k - a_k| = 2^{-\ell} \) for \( \ell \geq L' := L + 2 \), then \([a_k, b_k]\) is entirely contained within the larger \([a_K, b_K]\).

- \( \liminf p \leq y \)
- Indeed: Since the \([\rho \to \rho_<]\)-name contains in particular all rational pairs \((a_k, b_k)\) and these intervals are dense in \(\mathbb{R}\), there exists for every \(\ell \in \mathbb{N}\) some \(k\) such that \(|b_k - a_k| = 2^{-\ell}\) and \(x \in (a_k, b_k)\). Furthermore, it holds \(q_n \in (a_k, b_k)\) for some sufficiently large \(n\) because \(\lim_n q_n = x\). We have thus infinitely many triples \((k,\ell,n)\) for which \(p_{(k,\ell,n)}\) is defined by the first case in Equation \(4.5\) and thus agrees with some \(c_m < \min f([a_m, b_m]) \leq f(x) = y\) as \(x \in (a_k, b_k) \subseteq [a_m, b_m]\).

Concluding, we have \(\liminf_m p_m = y\). Although \(p\) may attain the value \(+\infty\), this can easily be overcome by proceeding to \(\bar{p}_m := p_m\) for \(p_m \neq +\infty\) and \(\bar{p}_m := \max\{0, \bar{p}_0, \ldots, \bar{p}_{m-1}\}\) for \(p_m = +\infty\) because this transformation \(p \rightarrow \bar{p}\) on sequences obviously does not affect their \(\liminf < \infty\). This yields a \(\rho_<\)-name for \(y\) which can finally be converted to the desired \(\rho_<\)-name due to the easy part of Lemma \(4.14\)).

In order to obtain a similar uniform claim yielding Theorem \(4.30\)), recall that every \((\rho_\to \rho_<)\)-computable function \(f : \mathbb{R} \to \mathbb{R}\) is necessarily both nondecreasing and lower semi-continuous (Fact \(4.25+b+c\)). This suggests

**Definition 4.34.** Let \(\text{MLSC}(\mathbb{R})\) denote the class of all nondecreasing, lower semi-continuous functions \(f : \mathbb{R} \to \mathbb{R}\). A \([\rho_\to \rho_<]\)-name for \(f \in \text{MLSC}(\mathbb{R})\) is an enumeration of the set \(\{(a,c) \in \mathbb{Q}^2 : c < f(a)\}\).

**Lemma 4.35.**

a) **Distinct** \(f, g \in \text{MLSC}(\mathbb{R})\) **have different sets** \(\{(a,c) : \ldots\}\) **according to Definition 4.34**; that is, \([\rho_\to \rho_<]\) constitutes a well-defined representation.

b) **A function** \(f \in \text{MLSC}(\mathbb{R})\) is \((\rho_\to \rho_<)\)-computable iff it has a computable \([\rho_\to \rho_<]\)-name.

c) **Let** \(f \in \text{MLSC}(\mathbb{R})\), \((a_k, c_k)_k\) with \(\{(a,c) \in \mathbb{Q}^2 : c < f(a)\} = \{(a_k, c_k) : k \in \mathbb{N}\}\), \(x \in \mathbb{R}\), and \(q = (q_n) \subseteq \mathbb{Q}\) with \(x = \liminf q_n\). **Then**, the rational sequence \(p\) defined by

\[
P_{(k,\ell,n)} := \begin{cases} 
\max_{m \geq +\infty} \{c_m : m \leq k \wedge a_m \geq a_k\} & \text{if } a_k < q_n < a_k + 2^{-\ell} \\
\text{otherwise}
\end{cases}
\]

satisfies \(\liminf p = f(x) =: y\).
d) Therefore, the apply operator \( \text{MLSC}(\mathbb{R}) \times \mathbb{R} \ni (f, x) \mapsto f(x) \) is \( ([\rho_\ast \rightarrow \rho_\ast] \times \rho'_\ast \rightarrow \rho'_\ast) \)-computable.

Proof. a) Let \( f, g \in \text{MLSC}(\mathbb{R}) \) with \( f \neq g \), that is, w.l.o.g. \( f(x_0) < g(x_0) \) for some \( x_0 \in \mathbb{R} \). There exists some \( c_0 \in \mathbb{Q} \) with \( f(x_0) < c_0 < g(x_0) \). Because they are nondecreasing and lower semi-continuous, their pre-images \( f^{-1}([c_0, \infty)) \) \( \not\ni x_0 \) and \( g^{-1}([c_0, \infty)) \ni x_0 \) on open half-interval \( (c_0, \infty) \) are again open half-intervals \( (x_f, \infty) \) and \( (x_g, \infty) \), respectively. As \( x_0 \) belongs to the second but not to the first, we have \( x_g < x_0 < x_f \) and therefore \( x_g < a_0 < x_f \) for some \( a_0 \in \mathbb{Q} \). Then \( a_0 \in (x_g, \infty) = g^{-1}([c_0, \infty)) \) yields \( c_0 < g(a_0) \) whereas \( a_0 \not\in (x_f, \infty) = f^{-1}([c_0, \infty)) \) asserts \( c_0 \not\in f(a_0) \).

b) Let \( M \) denote a Type-2 Machine \( \rho_\ast \rightarrow \rho_\ast \)-computing \( f \in \text{MLSC}(\mathbb{R}) \). Evaluating \( f \) at \( a \in \mathbb{Q} \) by simulating \( M \) on the \( \rho_\ast \)-name \( (a, a, a, \ldots) \) for \( a \) thus yields a \( \rho_\ast \)-name for \( f(a) \) which is (equivalent to) a list of all \( c \in \mathbb{Q} \) with \( c < f(a) \) [Weih00] Lemma 4.1.8. So dove-tailing this simulation for all \( a \in \mathbb{Q} \) yields the desired \( \rho_\ast \)-name for \( f \).

Conversely, knowing a \( \rho_\ast \)-name \( (a_k, c_k)_k \) for \( f \in \text{MLSC}(\mathbb{R}) \) and given an increasing sequence \( (q_n) \subseteq \mathbb{Q} \) with \( x = \sup_n q_n \), let

\[
p_n := c_n \text{ if } a_n \leq q_n, \quad p_n := -\infty \text{ otherwise.}
\]

Then, in the first case, \( p_n = c_n < f(a_n) \leq f(q_n) \leq f(x) =: y \) by monotonicity, and in the second \( p_n = -\infty \leq y \); hence \( \sup_n p_n \leq y \). To see \( \sup_n p_n \geq y \), fix arbitrary \( \epsilon > 0 \) and consider the open half-interval \( f^{-1}([y-\epsilon, \infty)) = (x_\epsilon, \infty) \) containing \( x \) and thus also some rational \( a = a_k \in (x_\epsilon, \infty) \). As \( a \in \mathbb{Q} \) with \( c_0 \not\in f(a_0) \) and therefore \( a_0 \in (a_k, x) \) for all \( n \geq N \). And finally there exists \( M \geq N \) with \( a_M = a_k \) and \( c_M \geq f(a_M) - \epsilon \). Together this ensures \( q_M > a_k = a_M \) because \( M \geq N \) and thus \( p_M = c_M \geq f(a_k) - \epsilon > y - 2\epsilon \) due to \( a_k \in f^{-1}([y-\epsilon, \infty)) \).

c) Take arbitrary \( \epsilon > 0 \). As \( f \) is increasing and lower semi-continuous, the pre-image \( f^{-1}([y-\epsilon, \infty)) \) is an open half-interval \( (x, \infty) \) containing \( x \). Therefore there exist \( K, L \in \mathbb{N} \) such that \( x < a_K + 2^{-L} < x \); furthermore, since the sequence \( (a_k, c_k)_k \) contains all rational pairs \( (a, c) \) with \( c < f(a) \), there is \( M \geq K \) such that \( a_M = a_K \) and \( c_M \geq f(a_M) - \epsilon \); and finally, since \( \liminf_n q_n = x > a_M + 2^{-L} \), it holds \( q_n > a_M + 2^{-L} \) for all \( n \geq N \) with an appropriate \( N \in \mathbb{N} \). Observe that \( q > a_M + 2^{-L} \) and \( a < q < a + 2^{-L} \) implies \( a \geq a_M \); so together we have for all \( n \geq N, \ell \geq L \), and \( k \geq M \) that \( p_{(k, n, \ell)} \) is either \( +\infty \) or \( \geq c_M \geq f(a_K) - \epsilon \geq f(x_\ell) - \epsilon > y - 2\epsilon \) due to the monotonicity of \( f \) and by definition of \( x_\ell < a_K \). This proves \( \liminf p_n \geq y \) because \( \epsilon \) was arbitrary.

To see the reverse inequality “\( \liminf p_n \leq y \)”, take arbitrary \( \ell \in \mathbb{N} \). There exists \( k \in \mathbb{N} \) with \( a_k < x < a_k + 2^{-\ell} \) and, because \( \liminf_n q_n = x \), also \( n \in \mathbb{N} \) with \( a_k < q_n < a_k + 2^{-\ell} \).

We therefore have infinitely many triples \( (n, k, \ell) \) for which \( p_{(n, k, \ell)} \) agrees with a certain \( c_m < f(a_m) \leq f(a_k) \leq f(x) = y \).

d) Given a \( \rho_\ast \)-name for \( x \), one can obtain a sequence \( (q_n) \subseteq \mathbb{Q} \) with \( x = \liminf_n q_n \) by virtue of Lemma 4.13(b). From this, the sequence \( p \subseteq \mathbb{Q} \) with \( \liminf n p_m = f(x) \) according to c) is obviously computable and yields, again by Lemma 4.14(b), a \( \rho'_\ast \)-name for \( y = f(x) \). \( \square \)

### 4.3.3 Relative Weierstrass Computability versus Weak Evaluation

Section 4.3.2 established the sequence \( \rho, \rho', \rho'', \ldots \) of increasingly weaker representations for \( \mathbb{R} \) to yield the strict hierarchy of \( (\rho \rightarrow \rho) \)-computable, \( (\rho' \rightarrow \rho') \)-computable, and \( (\rho'' \rightarrow \rho'') \)-computable functions \( f : [0, 1] \rightarrow \mathbb{R} \). We now compare these classes with those induced by the other kind of real hypercomputation suggested at the beginning of Section 4.3 relative to the Halting Problem \( H = \emptyset \) and its iterated jumps \( \emptyset', \ldots \).

Such a comparison makes sense because both weakly and oracle-computable real functions are necessarily continuous according to Fact 4.25(d)/Theorem 4.26(c) and Corollary 4.21.
The classical Weierstraß Approximation Theorem (Fact 4.36) establishes any continuous real function \( f : [0,1] \to \mathbb{R} \) to be the uniform limit \( f = \lim_{n \to \infty} P_n \) of a sequence of rational polynomials \( (P_n) \subseteq \mathbb{Q}[X] \). Here we write ‘ulim’ to denote uniform convergence of continuous functions on \([0,1]: \|f - P_n\| \to 0\), compare Equation (2.3). The effective Weierstraß Theorem (compare Fact 4.45) relates effectively evaluable real functions to effectively approximable ones:

**Fact 4.36.** Fix \( f : [0,1] \to \mathbb{R} \).

a) The function \( f \) is \((\rho \to \rho)\)-computable if and only if it holds

\[
[\rho \to \rho] : \text{There exists a computable sequence of (degrees and coefficients of) rational polynomials } (P_n) \subseteq \mathbb{Q}[X] \text{ such that } \|f - P_n\| \leq 2^{-n}.
\]

b) For \( f \) there exists a \( \emptyset' \)-computable sequence of polynomials \( (P_n) \) with \( \|f - P_n\| \leq 2^{-n} \) if and only if it holds

\[
[\rho \to \rho'] : \text{There exists a computable sequence } (Q_m) \subseteq \mathbb{Q}[X] \text{ satisfying } \|f - P_n\| \to 0 \text{ (i.e. such that } f = \lim_{m \to \infty} Q_m).\]

The notion “\( \rho \to \rho' \)” in Claim a) is justified by Fact 2.60d). Claim b) merely repeats Proposition 4.8. It straightforwardly generalizes to

**Lemma 4.37.** For a real function \( f : [0,1] \to \mathbb{R} \), there exists a \( \emptyset'' \)-computable sequence of polynomials \( (P_n) \) satisfying \( \|f - P_n\| \leq 2^{-n} \) if and only if it holds

\[
[\rho \to \rho''] : \text{There exists a computable sequence } (Q_m) \subseteq \mathbb{Q}[X] \text{ such that } f = \lim_{i,j} Q_{(i,j)}.
\]

**Proof.** If \( \emptyset'' \)-computable \( (P_n) \subseteq \mathbb{Q}[X] \) satisfies \( \|f - P_n\| \leq 2^{-n} \), then by virtue of the relativization of Fact 4.36b there exists some \( \emptyset' \)-computable \( (P_n) \subseteq \mathbb{Q}[X] \) converging to the same \( f \) uniformly on \([0,1]\). Again by Fact 4.36b), \( P_n = \lim_{m \to \infty} Q_{n,m} \) for some computable sequence \((Q_{n,m}) \subseteq \mathbb{Q}[X]\).

Conversely, if \( f = \lim_{m \to \infty} \tilde{P}_n \) with \( \tilde{P}_n := \lim_{m \to \infty} Q_{n,m} \) for a computable \((Q_{n,m})\), then let \( P_n := Q_{n,m} \) where

\[
m_n := \min \{ m \in \mathbb{N} : \forall k, \ell \geq m : \|Q_{n,k} - Q_{n,\ell}\| \leq 2^{-n} \}.
\]

This sequence \((m_n)\) is well-defined and yields \( \|P_n - \tilde{P}_n\| \leq 2^{-n} \), so \( f = \lim_{m \to \infty} \tilde{P}_n = \lim_{m \to \infty} P_n \). Moreover, the minimum in Equation 4.9 is taken over a co-r.e. set; namely \( r := \|Q_{n,k} - Q_{n,\ell}\| / 2^n \) being \( \rho \)-computable by virtue of [Weih00, COROLLARY 6.2.5] and the complementary condition “\( r > 1 \)” being \( \rho \)-r.e. open and hence recursive in \( \emptyset' \). Similar to Equation 4.9, this \( \emptyset' \)-computable sequence \((P_n) \subseteq \mathbb{Q}[X]\) converging uniformly (though merely ultimately) to \( f \) can be turned into a \( \emptyset'' \)-computable, fast convergent one. \( \square \)

We thus have two hierarchies of hypercomputable continuous real functions:

- \([\rho \to \rho], \ [\rho \to \rho'] , \ [\rho \to \rho''], \ldots \)
- \((\rho \to \rho), \ (\rho' \to \rho'), \ (\rho'' \to \rho''), \ldots \)

By Fact 4.36b) their respective ground-levels coincide. Our next result compares their respective higher levels. They turn out to lie skew to each other (Claim c).

**Theorem 4.38.** a) Let \( f : [0,1] \to \mathbb{R} \) be \([\rho \to \rho] \)-computable (in the sense of Fact 4.36b). Then, \( f \) is \((\rho' \to \rho')\)-computable.

b) Let \( f : [0,1] \to \mathbb{R} \) be \((\rho' \to \rho')\)-computable. Then, \( f \) is \([\rho \to \rho']\)-computable (in the sense of Lemma 4.37).

c) There is a \((\rho' \to \rho')\)-computable but not \([\rho \to \rho] \)-computable \( f : [0,1] \to \mathbb{R} \).

Underlying c) is the idea that every \( [\rho \to \rho] \)-computable \( f : [0,1] \to \mathbb{R} \) has a modulus of uniform continuity recursive in \( \emptyset' \); whereas a \((\rho' \to \rho')\)-computable \( f \), although also uniformly continuous, in general does not.

Before proceeding to the proof, we first provide some generalizations of Observation 2.48.
Lemma 4.39.  

a) Let \( f : \mathbb{R} \to \mathbb{R} \) be \((\rho \to \rho \_ \_ \_ )\)-computable. Then the set
\[
\{(a,b,c) \in \mathbb{Q}^3 \mid \exists x \in [a,b] : f(x) > c \}
\]
(i.e. the question of whether \( f \) on \([a,b]\) exceeds \( c \)) is semi-decidable.

b) Let \( f : \mathbb{R} \to \mathbb{R} \) be \((\rho' \to \rho' \_ \_ \_ )\)-computable. Then the set
\[
\{(a,b,c) \in \mathbb{Q}^3 \mid \exists x \in [a,b] : f(x) > c \}
\]
(i.e. the question of whether \( f \) on \([a,b]\) exceeds \( c \)) is semi-decidable relative to \( \Psi' \).

c) Let \( f : \mathbb{R} \to \mathbb{R} \) be \((\rho' \to \rho')\)-computable. Then the set
\[
\{(a,b,c,m) \in \mathbb{Q}^3 \times \mathbb{N} \mid \forall x \in [a,b] : c - 2^{-m} \leq f(x) \leq c + 2^{-m} \}
\]
is decidable relative to \( \Psi'' \).

Proof. a) is standard; c) follows from b), which is established as follows:
Lower semi-continuity of \( f \) due to Theorem 4.26a) implies that, if \( f \) exceeds \( c \) on the compact interval \([a,b]\), then it does so on some rational \( x \). Feeding, for any such \( x \in [a,b] \cap \mathbb{Q} \), the \( \rho' \)-name \((x,x,x,x,\ldots)\) for \( x \) into the Type-2 Machine computing \( f \) reveals the mapping \( \mathbb{Q} \ni x \mapsto f(x) \) to be \((\nu_Q \to \rho_\_ \_ \_ )\)-computable. With \( \Psi' \)-oracle, it thus becomes \((\nu_Q \to \rho_\_ \_ )\)-computable by virtue of \( [ZHWe01 \text{ \textbf{Lemma 4.2}}] \). Since \( \{(y,c) : y > c \} \) is \((\rho_\_ \_ \_ \_ × \nu_Q)\)-semi-decidable, the claim follows. 

\[
\begin{array}{c}
\text{Figure 4.9: A piecewise linear and a smooth unit pulse, and a non-overlapping superposition by scaled shifts used in the Proof of Theorem 4.38.}
\end{array}
\]

Proof (Theorem 4.38).  

a) Let \((P_n) \subseteq \mathbb{Q}[X]\) denote a computable sequence converging uniformly (yet not necessarily fast) to \( f \). Let \( x \in [0,1] \) be given as the limit of a sequence \((q_n) \subseteq \mathbb{Q} \). Then, \( p_n := P_n(q_n) \) eventually converges to \( f(x) \).

b) Let \( x \in [0,1] \) be given by (an equivalent to) its \( \rho \)-name in the form of two rational sequences \((a_n)\) and \((b_n)\) with \( \{x\} = \bigcap_n [a_n,b_n] \). There exists a rational sequence \((c_m)\) forming a \( \rho \)-name for \( f(x) \), that is, satisfying \( c - 2^{-m} \leq f(x) \leq c + 2^{-m} \) for all \( m \); and by virtue of Lemma 4.39c), such a sequence can be found with the help of a \( \Psi'' \)-oracle. This reveals that \( f \) is \( \Psi'' \)-recursive in the sense of [Ho99 \text{ \textbf{SECTION 4}}] and thus, similarly to [Ho99 \text{ \textbf{COROLLARY 17}}], \((\rho \to \rho)''\)-computable.

c) Let \( h : \mathbb{N} \to \mathbb{N} \) denote a \( \Psi' \)-computable injective total enumeration of some subset \( H = h[\mathbb{N}] \subseteq \Sigma_2 \setminus \Delta_2 \). Observe that \( a_m := 2^{-h(m)} \) is a \( \rho' \)-computable real sequence converging to 0 with modulus of convergence [Weih00 \text{ \textbf{DEFINITION 4.2.2}}] lacking \( \Psi' \)-recursivity; compare [Weih00 \text{ \textbf{EXERCISE 4.2.4c}}]. Let \( \varphi : \mathbb{R} \to \mathbb{R} \) denote some \((\rho \to \rho)\)-computable unit pulse, that is, vanishing outside \([0,1]\) and having height \( \max f(x) = \varphi(\frac{1}{2}) = 1 \); a piecewise
linear ‘hat’ function for instance will do fine but we can even choose \( \varphi \) as in \[ \text{PER89} \] Theorem 1.1.1.1 to obtain the counter-example

\[
f(x) := \lim_{M \to \infty} f_M, \quad f_M(x) := \sum_{m=1}^{M} a_m \cdot \varphi(2^m x - 1), \quad (4.10)
\]

(that is, a non-overlapping superposition of scaled shifts of such pulses) to be \( C^\infty \); compare Figure 4.3. By Theorem 4.3.1, \( x \mapsto a_m \cdot \varphi(2^m x - 1) \) is \( (\rho' \to \rho') \)-computable; in fact even uniformly in \( m \): Given \( (q_n)_n \subseteq \mathbb{Q} \) with \( x = \lim_n q_n \) and \( M \in \mathbb{N} \), one can obtain a sequence \( (p_k,M)_k \subseteq \mathbb{Q} \) with \( \lim_k p_k,M = f_M \). The functions \( f_M \) converge uniformly (though not effectively) to \( f \) because of the disjoint supports of the terms \( \varphi(2^m x - 1) \) in Equation (4.10).

Therefore \( \lim_{M \to \infty} p_{M,M} = f(x) \), thus establishing the \( (\rho' \to \rho') \)-computability of \( f \).

Suppose \( f \) were \( \rho \in (\rho \mapsto \rho) \)-computable. Then, by virtue of \[ \text{Ho99} \text{ Lemma 15}, \] it has a \( \emptyset' \)-recursive modulus of uniform continuity; cf. \[ \text{WeZh00} \text{ Definition 6.2.6.2}. \] In particular given \( n \in \mathbb{N} \), one can \( \emptyset' \)-compute \( m \in \mathbb{N} \) such that \( x := 2^{-m} \) and \( y := \frac{3}{2}x \) satisfy \( 2^{-n} \geq |f(x) - f(y)| = |0 - a_m| \) contradicting the hypothesis that \( (a_m) \) has no \( \emptyset' \)-recursive modulus of continuity.

### 4.3.4 Weak Function Evaluation and Borel’s Hierarchy

According to Fact 2.45 and for r.e. open \( X \subseteq \mathbb{R}^k \), \( (\rho \to \rho) \)-computability of a real function \( f : X \to \mathbb{R} \) amounts to effective continuity: its pre-images \( f^{-1}((q, \infty)) \) and \( f^{-1}((\infty, p]) \) are open again (that is in \( \Sigma_1(X) \)) and \( \delta_{\Sigma_1} \)-computably uniformly in \( q,p \in \mathbb{Q} \). More precisely, \( (\rho \to \rho) \)-name of \( f \) is equivalent to a \( \delta_{\Sigma_1} \)-name of the two families of open sets \( (f^{-1}((q, \infty)))_{q \in \mathbb{Q}} \) and \( (f^{-1}((\infty, p]))_{p \in \mathbb{Q}} \) \[ \text{Weih00} \text{ Lemma 6.1.7}. \]

**Observation 4.40.** Indeed it follows that \( f^{-1}(\bigcup_p (q_n, p_n)) = \bigcup_q (f^{-1}((q_n, \infty)) \cap f^{-1}((\infty, p_n])) \) is \( \delta_{\Sigma_1} \)-computable according to \[ \text{Weih00} \text{ Corollary 5.1.18.1 and Example 5.1.19.1}. \]

A weakening of the above notion is \( (\rho \to \rho_\perp) \)-computability; equivalently \[ \text{WeZh00} \text{ Theorem 4.5.1(1) and Corollary 5.1.2(2)}: \] pre-images \( f^{-1}((q, \infty)) \) (but not necessarily \( f^{-1}((\infty, q]) \)) are in \( \Sigma_1(X) \) and \( \delta_{\Sigma_1} \)-computable uniformly in \( q \in \mathbb{Q} \). It is natural to call this condition effective lower semi-continuity, and it has led to the representation of the class \( \text{LSC}(X) \) of lower semi-continuous functions on \( X \) we already mentioned in Lemma 4.33 encode \( f \) as the join of a \( (\delta_{\Sigma_1})^2 \)-name of the family of open sets \( (f^{-1}((q, \infty)))_{q \in \mathbb{Q}} \). We shall write \( \rho \to \rho_\perp \) for this representation because \[ \text{WeZh00} \text{ actually establishes it as equivalent to the one from Fact 2.60a} \) when \( A = \mathbb{R} \) and \( B = \mathbb{R} \) are chosen, the latter equipped with topology \( \{(q, \infty) : q \in \mathbb{Q} \} \).

A different weakening due to Brattka considers, equally naturally, functions \( f : X \to \mathbb{R} \) for which both pre-images \( f^{-1}((q, \infty)) \) and \( f^{-1}((\infty, p]) \) belong to the Borel class \( \Sigma_d(X) \) and are \( \delta_{\Sigma_d} \)-computable uniformly in \( p,q \in \mathbb{Q} \). He terms this condition effective \( \Sigma_d \)-measurability because it effectivizes the well-known notion in classical measure theory which calls a function \( f \in \Sigma_d \)-measurable if \( f^{-1}(V) \) belongs to \( \Sigma_d(X) \) for every open \( V \). (The comprehensive paper \[ Brat05 \] studies this notion and its consequences thoroughly. It is as general as to also include partial and multi-valued functions on arbitrary computable metric spaces, but in that respect goes beyond our purpose.) \[ Brat05 \] also introduces a natural representation \( \delta_{\Sigma_d(X \to \mathbb{R}_\perp)} \) for the class of \( \Sigma_d(X) \)-measurable functions \( f : X \to \mathbb{R} \) as a \( (\delta_{\Sigma_d})^2 \)-name of the sequences \( (f^{-1}((q, \infty)))_{q \in \mathbb{Q}} \)

\[ \text{and } (f^{-1}((\infty, p]))_{p \in \mathbb{Q}} \text{.} \]

It is a simple matter to unify these approaches:

**Definition 4.41.** Call \( f : X \to \mathbb{R} \) \( \Sigma_d \)-lower semi-measurable if \( f^{-1}((q, \infty]) \in \Sigma_d(X) \) for all \( q \in \mathbb{R} \). This function is effectively \( \Sigma_d \) lower semi-measurable if the sequence \( (f^{-1}((q, \infty]))_{q \in \mathbb{Q}} \) is \( (\delta_{\Sigma_d})^2 \)-computable. The representation \( \delta_{\Sigma_d(X \to \mathbb{R}_\perp)} \) of all \( \Sigma_d \) lower semi-measurable functions is defined to encode \( f : X \to \mathbb{R} \) as a \( \delta_{\Sigma_d} \)-name of this sequence.
Effective lower semi-continuity thus amounts to effective $\Sigma_1$ lower semi-measurability; and Observation \[4.40\] of course generalizes: [Brat05] Proposition 3.2(4) and Lemma \[4.14b\] imply $\delta_{\Sigma_d}(X \to \mathbb{R}) \equiv \delta_{\Sigma_d}(X \to \mathbb{R}_\leq) \wedge \delta_{\Sigma_d}(X \to \mathbb{R}_\geq)$; where the latter representation of all $\Sigma_d$ upper semi-measurable functions encodes $f : X \to \mathbb{R}$ as a $\delta_{\Sigma_d}(X \to \mathbb{R}_\leq)$–name of the $\Sigma_d$ lower semi-measurable function $f$.

The main result of the present section connects these notions and representations to weak function evaluation ($\rho \to \rho^{(d)}$) and ($\rho \to \rho^{(d)}$) in the sense of Section \[4.3.1\]

**Theorem 4.42.** Fix r.e. open $X \subseteq \mathbb{R}^k$ and $d \in \mathbb{N}$.

a) The uniformly characteristic function

$$\chi : \Sigma_d(X) \times X \to \{0,1\}, \quad (S, \bar{x}) \mapsto \chi_S(\bar{x}) := 1 \text{ if } \bar{x} \in S \text{ and } \chi_S(\bar{x}) := 0 \text{ if } \bar{x} \notin S$$

is $(\delta_{\Sigma_d} \times \rho \to \rho^{(d-1)})$–computable.

b) The apply operator $(f, x) \mapsto f(x)$ on $\Sigma_d$ lower semi-measurable functions on $X$ is $(\delta_{\Sigma_d}(X \to \mathbb{R}_\leq) \times \rho \to \rho^{(d-1)})$–computable.

c) Every $(\rho \to \rho^{(d-1)})$–continuous function $f : X \to \mathbb{R}$ is $\Sigma_d$ lower semi-measurable; every $(\rho \to \rho^{(d-1)})$–computable one is effectively $\Sigma_d$ lower semi-measurable, uniformly in $f$ given by an encoding of a realization.

d) It holds $\delta_{\Sigma_d}(X \to \mathbb{R}_\leq) \equiv [\rho \to \rho^{(d-1)}]$ and $\delta_{\Sigma_d}(X \to \mathbb{R}) \equiv [\rho \to \rho^{(d-1)}]$ with representations of function spaces according to Fact \[2.60\].

In particular, the topological complexity of feasible functions strictly grows as the encoding of the real output weakens:

**Corollary 4.43.** A function $f : X \to \mathbb{R}$ is $(\rho \to \rho^{(d)})$–computable iff it is effectively $\Sigma_{d+1}$–measurable.

It is natural to ask for a generalization that also treats weakened input encodings. Corollary \[4.61\] below will, extending Fact \[4.25\] and Theorem \[4.26\], topologically characterize $(\rho^{(k)} \to \rho^{(d)})$–continuous functions; concerning $(\rho^{(d)} \to \rho^{(d)})$–computable ones, however, we ask

**Question 4.44.** Can one similarly characterize, for $k,d \in \mathbb{N}$, $(\rho^{(k)} \to \rho^{(d)})$–computable functions in terms of effective measurability?

Here, even the case $d = 2 = k$ seems awkward: By (respective minor modifications considering open domains of) the relativization of Fact \[2.45\] and by Theorem \[4.38\] every effectively $\Sigma_1$–measurable function relative to $\emptyset'$ is $(\rho' \to \rho')$–computable but a $(\rho' \to \rho')$–computable $f$ is in general effectively $\Sigma_1$–measurable only relative to $\emptyset''$.

**Proof.** Proof (Theorem \[4.42\]).

a) By induction on $d$, starting with $d = 1$: Given a $\rho$–name of $\bar{x} \in X$ and a $\theta_{<}$–name of an open $U \subseteq U$, membership “$\bar{x} \in U”$ is semi-decidable; so output 0s while uncertain and start writing 1s as soon as membership has been established: this yields a $\rho_{<}$–name of $\chi_U(\bar{x})$.

Now let $\Sigma_{d+1} \ni S = \bigcup_n (X \setminus S_n)$ be given by $\delta_{\Sigma_d}$–names of $S_n \in \Sigma_d(X)$, $n \in \mathbb{N}$. By virtue of the induction hypothesis, $\rho^{(d-1)}$–compute the respective values $y_n := \chi_{S_n}(\bar{x})$. Since $\bar{x} \in S \iff \exists n : \bar{x} \notin S_n$, we have $\chi_S(\bar{x}) \equiv \sup_n (1 - y_n)$.

b) Given (a $\rho$–name of) $\bar{x} \in X$, consider for each $q \in \mathbb{Q}$ a $\delta_{\Sigma_q}$–name of $S_q := f^{-1}((q, \infty])$. Claim a) yields from that a $\rho^{(d-1)}$–name of $y_q := \chi_{S_q}(\bar{x})$, that is $y_q = 1$ in case $\bar{x} \in S_q$ and $y_q = 0$ in case $\bar{x} \notin S_q$. Easy scaling converts that to $z_q := q$ in case $f(\bar{x}) > q$ and to $z_q := -\infty$ in case $f(\bar{x}) \leq q$. We finally obtain a $\rho^{(d-1)}$–name of $\sup_q z_q = f(\bar{x})$ because $\mathbb{R}^\mathbb{N} \ni (z_n)_n \mapsto \sup_n z_n \in \mathbb{R}$ is obviously $((\rho^{(d)})^n \to \rho^{(d)})$–computable.
4.3. HYPERCOMPUTING REAL FUNCTIONS

\textbf{d = 0:} Evaluate \( f \) simultaneously on all \( \bar{x} \in X \) to obtain rational sequences \( p_{\bar{x},n} \) with \( f(\bar{x}) = \sup_n p_{\bar{x},n} \). More precisely, using feasible countable (as opposed to infeasible uncountable) dovetailing, simulate the machine evaluating \( f \) on all initial parts of \( \rho \)-names of \( \bar{x} \in X \), that is on all finite rational sequences \( \bar{q} = (q_1, q_2, \ldots, q_N) \) in \( \mathbb{Q}^k \) with \( N \in \mathbb{N} \) and \( |q_n - q_k| \leq 2^{-n} \forall n \leq k \leq N \). For each \( \bar{q} \), we obtain as output a finite rational sequence \( (p_{\bar{q},m})_{m \leq N} \). Observe that \( \bar{q} \) is the initial segment of a \( \rho \)-name to any \( x \in B_{\bar{q}} := \bigcup_{n=1}^{\infty}(q_n, 2^{-n}) \), \( B_{\bar{q}} \) having a non-empty interior. Hence

\[
\exists m : p_{\bar{q},m} > a \iff \forall \bar{x} \in B_{\bar{q}} : f(\bar{x}) > a \iff \exists \bar{x} \in B_{\bar{q}} : f(\bar{x}) > a
\]

which implies

\[
f^{-1}[(a, \infty)] = \bigcup_{\bar{q}, m \in \Pi_1} \left\{ \prod B_{\bar{q}} : p_{\bar{q},m} > a \right\} = \bigcup_{\bar{q}, m \in \Pi_2} \left\{ \prod B_{\bar{q}} : p_{\bar{q},m} > a \right\}
\]

and immediately yields \( \delta_{\Sigma_1} \)-computability of \( f^{-1}[(a, \infty)] \) for a given \( a \in \mathbb{Q} \).

\textbf{d = 1:} Similarly, evaluate \( f \) on all \( \bar{x} \in X \) to obtain sequences \( p_{\bar{x},n,m} \) with \( f(\bar{x}) = \sup_n p_{\bar{x},n,m} \). More precisely countable dovetailing yields, for each finite \( \rho \)-initial segment \( \bar{q} \), a finite sequence \( (p_{\bar{q},m,n})_{m,n} \) in \( \mathbb{Q}^k \) with

\[
\exists m \forall n : p_{\bar{q},m,n} > a \iff \forall \bar{x} \in B_{\bar{q}} : f(\bar{x}) > a \iff \exists \bar{x} \in B_{\bar{q}} : f(\bar{x}) > a
\]

which, because of

\[
f^{-1}[(a, \infty)] = \bigcup_{\bar{q}, m,n \in \Pi_1} \left\{ \prod B_{\bar{q}} : p_{\bar{q},m,n} > a \right\} = \bigcup_{\bar{q}, m,n \in \Pi_2} \left\{ \prod B_{\bar{q}} : p_{\bar{q},m,n} > a \right\}
\]

constitutes a \( \delta_{\Sigma_2} \)-name of \( f^{-1}[(a, \infty)] \) as the \( \delta_{\Sigma_2} \)-names of all open \( X \setminus A_{\bar{q},n} \).

\textbf{d = 2:} Compute finite rational sequences \( (p_{\bar{q},m,n,k})_{m,n,k} \) with

\[
\exists m \forall n \exists k : p_{\bar{q},m,n,k} > a \iff \forall \bar{x} \in B_{\bar{q}} : f(\bar{x}) > a \iff \exists \bar{x} \in B_{\bar{q}} : f(\bar{x}) > a
\]

\[
f^{-1}[(a, \infty)] = \bigcup_{\bar{q}, m,n \in \Pi_1} \left\{ \prod \hat{B}_{\bar{q}} : p_{\bar{q},m,n,k} > a \right\} = \bigcup_{\bar{q}, m,n \in \Pi_2} \left\{ \prod \hat{B}_{\bar{q}} : p_{\bar{q},m,n,k} > a \right\}
\]

\textbf{d \geq 3:} analogously.

d) \( \delta_{\Sigma_d(X \rightarrow \mathbb{R}_c)} \equiv [\rho \rightarrow \rho^{(d-1)}] \) follows from Claims a+b) and \([\text{Wei}h00, \text{Lem}a 3.3.14] \). The second claim is a consequence of \( \delta_{\Sigma_d(X \rightarrow \mathbb{R}_c)} \equiv \delta_{\Sigma_d(X \rightarrow \mathbb{R}_c)} \land \delta_{\Sigma_d(X \rightarrow \mathbb{R}_c)} \).
4.3.5 Are Discontinuous Real Functions Hypercomputable?

Let us return to Question 4.22 (which is not to be confused with the question of physical realizability we discussed in Section 3.4.1).

It is easy to define a reasonable representation (and thus also a notion of computability) on a subclass of real functions including discontinuous ones. For instance, do so for the space of generalized functions in order to study the effective solvability of partial differential equations. But for these objects $f$, evaluation $x \mapsto f(x)$ on real points $x$ does not even make sense mathematically.

Let us therefore be more specific about Question 4.22 and understand it in the sense of (hyper-)effective function evaluability. Here, ordinary oracle access does not help (Corollary 4.21); but hypercomputation in the form of weakened real number representations does: Heaviside's function, for instance, has turned out to be $(\rho \rightarrow \rho')$–computable.

Note that this example relies on a weaker encoding $\rho'$ for the output value $y = f(x)$ than that of the input argument $x$, namely $\rho$. However, a notion of computability involving weaker output encoding than input obviously lacks closure under composition:

**Example 4.45.** Let $f : \mathbb{R} \rightarrow \mathbb{R}$, $f(0) := 0$ and $f(x) := 1$ for $x \neq 0$. Let $g(x) := -x$. Then both $f$ and $g$ are $(\rho \rightarrow \rho')$–computable but their composition $g \circ f : 0 \mapsto 0, 0 \neq x \mapsto -1$ fails lower semi-continuity.

On the other hand if input and output encoding are required to coincide, nothing is gained topologically by weakening them both: compare Fact 4.25 and Theorem 4.26.

Under these reasonable conditions, the answer to Question 4.22 thus still seems to be negative. Only in Section 4.5 shall we see that non-deterministic computation does allow for the evaluation of certain discontinuous functions while maintaining closure under composition. In the meantime, let us study

4.3.6 Markov with Oracles

In Section 2.6.3 we considered the Markov computability of real functions and saw that (at least every total) such $f$ is necessarily continuous. Observe how the proof of Fact 2.67 proceeded by reducing the evaluation of Heaviside's function at a point of discontinuity to the task of deciding “$e \in H$?”; it is thus justified for Chee Yap to call the real zero test “$x = 0?$” the numerical Halting problem. On the other hand, when oracle access to $H$ is granted, this test becomes easily decidable in the Markov setting (as opposed to relativized evaluability, recall Corollary 4.21):

**Example 4.46.** Given a Gödel index $e$ of some machine $M_e$ computing $x$, modify $M_e$ slightly to detect, and to abort (only) in the case that $x \neq 0$. Feed this new machine’s index $\tilde{e}$ into the Halting oracle. A negative answer is equivalent to $x = 0$.

Observe that this example shows only a certain partial relativization of Fact 2.67 to fail: the operations on Gödel indices are permitted oracle access, whereas the indices input and output themselves still refer to oracle-free machines.

It seems useful to generalize Tseitin’s Theorem to Markov hypercomputability by answering the following

**Question 4.47.** Fix a total function $f : \mathbb{R} \rightarrow \mathbb{R}$.

a) If $f$ is Markov–computable relative to $\emptyset'$, is it then $\Sigma_2$–measurable?

b) Consider naive Markov hypercomputation of $f$ in the following sense: the input consists of a Gödel index $e$ of some Turing machine $M_e$ generating $(q_n) \subseteq \mathbb{Q}$ with $x = \lim_n q_n$, and the output is a similar integer $e'$ such that $M_{e'}$ produces $(p_m) \subseteq \mathbb{Q}$ with $f(x) = \lim_m p_m$. If the transition $e \mapsto e'$ is effective, does this require $f$ to be continuous, again?

c) Characterize the class of total functions that are Markov–computable relative to $\emptyset'$.
d) How about higher degrees?

Item b) amounts to a full relativization of Fact 2.67.

4.4 Revising Type-2 Computation

Definition 4.12 has turned the two real representations \( \rho \leq \rho \) into a countably infinite hierarchy of weakened representations \( \rho^{(d)} \leq \rho^{(d)} \leq \rho^{(d+1)} \) related to the classical Arithmetical Hierarchy in the sense that a real number \( x \) is \( \rho^{(d)} \)-computable (\( \rho^{(d)} \)-computable) iff it is \( \rho \)-computable (\( \rho \)-computable) relative to \( \emptyset^{(d)} \). The construction of \( \rho' \) from \( \rho \) (\( \rho' \) from \( \rho \)) was ad-hoc and based on Theorems 4.9 and 4.10. Similarly, Fact 4.36a) and Lemma 4.37 provided the justifications for the representations \( [\rho \rightarrow \rho'] \) and \( [\rho \rightarrow \rho''] \) for \( C([0,1]) \) implicitly defined in them.

The present section proposes as a natural generalization a generic way of turning a given representation \( \alpha : \{0,1\}^\omega \rightarrow A \) into a weakened one \( \alpha' : \{0,1\}^\omega \rightarrow A \) such that it holds: \( a \in A \) is \( \alpha' \)-computable iff \( a \) is \( \alpha \)-computable relative to \( \emptyset \).

4.4.1 Motivation

An important point in the definition of a Type-2 machine is that its output tape must be one-way: compare e.g. [Weih00, top of p.15]. This condition models and reflects the practitioner’s reasonable requirement an (infinite) real number computation be aborted as soon as the desired precision is reached, knowing that this preliminary approximation will not be reverted. It also enters crucially in the Main Theorem (e.g. in the form of Proposition 2.44 or Fact 2.57).

In the Type-1 setting with finite results appearing after finite time, every machine with rewritable output tape can be simulated by one with one-way output tape. However, when finite results appearing after possibly finite time are considered, we enter the realm of revising computations; recall Section 3.6. Its Definition 3.6 for finite strings leads to two natural ones for infinite ones (recall Requirement 3.8):

- either require sequence \( (\bar{\tau}_m)_m \) in \( \{0,1\}^\omega \) to converge to \( \bar{\sigma} \) in Cantor topology, i.e. symbol-wise;
- or demand that \( (\bar{\tau}_m)_m \) ‘stabilizes’ to \( \bar{\sigma} \) in the following sense: \( \exists M : \forall m \geq M : \bar{\tau}_m = \bar{\sigma} \).

Section 4.4.2 will focus on the notion of revising Type-2 computation induced by the first condition while consideration of the second is postponed to Section 6.4. Both are based on the observation that preliminary information in the sense of strings with embedded ‘editing’ commands as in Example 3.7 occur not only as output from but also as input to programs:

Data Stream Algorithms

Many practical applications are desired to run ‘forever’: a scheduler, a router, a monitor are all not supposed to terminate but to continue processing the stream of data presented to them. This has led to the prospering field of Data Stream Algorithms\(^{16}\). It distinguishes various ways in which the input can be presented to the program [Muth05, Section 4.1]:

- In the Time Series Model, all data items (binary digits, say) are enumerated in order; in particular, they must not later be reverted.

This corresponds to classical, i.e. non-revising input.

- The Turnstile Model on the other hand permits (finitely many) later updates to previously enumerated items.

This corresponds to revising input (subject to Requirement 3.8).

\(^{16}\)Which usually focuses on the (space) complexity of randomized approximations of discrete problems, however
4.4.2 The Jump of a Representation

We start with Cantor space \( \{0,1\}^\omega \) which is usually and canonically represented by the identity \( \iota \) [Weih00 Definition 3.1.2.1].

**Definition 4.48.** Let the revising representation \( \iota' : \subseteq \{0,1\}^\omega \rightarrow \{0,1\}^\omega \) encode (via some computable standard pairing \( \langle \cdot \rangle \) : \( \{0,1\}^{\omega \times \omega} \rightarrow \{0,1\}^\omega \)) an infinite string \( \sigma \in \{0,1\}^\omega \) as a sequence of infinite strings ultimately converging to \( \sigma \).

This amounts to the naive Cauchy representation of the effective metric Cantor space [BrHe02 Section 6]. An \( \iota' \)-name for \( (\sigma_n)_n \) is thus (an \( \iota \)-name for) some \( ((\tau_{(n,m)})_m)_m \in \{0,1\}^\omega \) such that, for each \( n \in \mathbb{N} \), \( \sigma_n = \lim_{m \to \infty} \tau_{(n,m)} \). We state for the records and for later application the following easy

**Claim 4.49.** For each \( n \in \mathbb{N} \) let \( (\bar{\tau}_{(n,m)})_m \) be a sequence in \( \{0,1\}^\omega \) and \( \bar{\sigma}_n \in \{0,1\}^\omega \). Then
\[
\forall n : \bar{\tau}_{(n,m)} \xrightarrow{m \to \infty} \bar{\sigma}_n \iff ((\bar{\tau}_{(n,m)})_m)_{m \to \infty} = (\bar{\sigma}_n)_n .
\]

The name \( \iota' \), reminiscent of the recursion-theoretic *jump*, is justified because Shoenfield’s Limit Lemma 3.5 immediately yields

**Remark 4.50.** Let \( O \) denote an arbitrary-theoretic oracle. An infinite string is (\( \iota \)-) computable relative to \( O' \) if and only if it is \( \iota' \)-computable relative to \( O \).

**Lemma 4.51.** a) Every (\( (\iota \to \iota) \)-computable string function \( F : \subseteq \{0,1\}^\omega \rightarrow \{0,1\}^\omega \) is also (\( \iota' \to \iota' \)-computable.

b) More precisely, the apply operator \( (F, \bar{\sigma}) \mapsto F(\bar{\sigma}) \) is \( (\eta^{\omega \times \omega} \times \iota' \to \iota') \)-computable on
\[
\{(F, \bar{\sigma}) \mid F : \subseteq \{0,1\}^\omega \rightarrow \{0,1\}^\omega \text{ continuous, } \bar{\sigma} \in \text{dom}(F) \} .
\]

c) Every (\( \iota' \to \iota' \)-continuous string function \( F : \subseteq \{0,1\}^\omega \rightarrow \{0,1\}^\omega \) is (Cantor-)continuous.

In b), \( \eta^{\omega \times \omega} \) denotes a natural representation for continuous string functions [Weih00 Section 2.3].

**Proof.** a) follows from b) and Fact 2.57

b) Let \( \bar{\tau}_m := F(\bar{\tau}_m) \) where \( F : \subseteq \{0,1\}^\omega \rightarrow \{0,1\}^\omega \) is continuous. Then \( \lim_{m \to \infty} \bar{\tau}_m = F(\lim_{m \to \infty} \bar{\tau}_m). \)

c) Similar to Fact 4.25); see also [BrHe02 Section 6] which applies to any effective metric space.

We now extend Lemma 4.51) in five steps to multivalued functions:

**Lemma 4.52.** Fix \( X \subseteq \{0,1\}^\omega \) and write \( X' := \bigcup_{\bar{x} \in X} \bar{x}' \) where \( \bar{x}' := \{(\bar{\sigma}_m)_m \subseteq \{0,1\}^\omega : \lim_{m \to \infty} \bar{\sigma}_m = \bar{x}\} \). For each \( n \in \mathbb{N} \), let \( G_n : X' \rightarrow \{0,1\}^\omega \) be continuous (i.e. locally constant). Furthermore suppose that, for every \( (\bar{\sigma}_m)_m \in X' \), \( \lim_n G_n((\bar{\sigma}_m)_m) \) exists.

a) To \( \bar{x} \in X \) there exist \( b \in \{0,1\}, M, K \in \mathbb{N} \), and \( \bar{\sigma}_1, \ldots, \bar{\sigma}_M \in \{0,1\}^\omega \) such that every \( (\bar{\sigma}_m)_m \in \bar{x}' \) with \( \bar{\sigma}_{m,K} = x_k \) (for all \( k \leq K \) and all \( M > k \)) satisfies \( \lim_n G_n((\bar{\sigma}_m)_m) = b \).

b) For \( \bar{x} \in X \) and \( b \in \{0,1\} \) as in a), there exists \( L \in \mathbb{N} \) such that, to every \( \bar{y} \in X \) with \( x_k = y_k \) for \( k \leq L \), there is some \( (\bar{\tau}_m)_m \in \bar{y}' \) with \( \lim_n G_n((\bar{\tau}_m)_m) = b \).

c) Let \( (U_i)_{i \in I} \) denote a family of open balls in Cantor space \( \{0,1\}^\omega \), i.e. \( U_i = \bar{u}_i \circ \{0,1\}^\omega \) with \( \bar{u}_i \in \{0,1\}^\prime \). Then there exists a subfamily \( (U_{i_n})_n \) which is pairwise disjoint such that \( \bigcup_i U_i = \bigcup_n U_{i_n} \).

d) Let \( g \subseteq \{0,1\}^\omega \Rightarrow \{0,1\} \) denote a multivalued mapping. If \( g \) is (\( \iota' \to \iota' \)-continuous, then it admits a locally constant selection.
4.4. REVISING TYPE-2 COMPUTATION

e) Every \((i' \to i')\)-continuous multivalued mapping \(f : \subseteq \{0,1\}^\omega \Rightarrow \{0,1\}^\omega\) admits a (Cantor-)continuous selection.

Proof. a) W.l.o.g. let \(\bar{x} = \bar{0}\) and \(b := \lim G_n(\bar{x}, \bar{x}, \ldots) = 0\). Start off with \(M, K := 0\): if every sequence \((\bar{\sigma}_m)\) converging to \(0\) satisfies \(\lim G_n((\bar{\sigma}_m)) = 0\), then we are done.

Otherwise \(\lim G_n((\bar{\sigma}_m^{(1)})) = 1 =: b_1\) for some \((\bar{\sigma}_m^{(1)})\) with \(\lim_m \bar{\sigma}_m = \bar{0}\). In particular there is some \(n_1, M_1 \in \mathbb{N}\) such that \(G_{n_1}((\bar{\sigma}_m^{(1)})) = 1\) and \(\sigma_m^{(1)} = 0\) for all \(m > M_1\). By continuity of \(G_{n_1}\), there exists some \(M_1 < \bar{M}_1\) such that \(G_{n_1}((\bar{\sigma}_m)) = 1\) holds whenever \((\bar{\sigma}_m)\) satisfies \(\bar{\sigma}_m = \sigma_m^{(1)}\) for all \(m \leq M_1\). Now if even \(\lim_n G_n((\bar{\sigma}_m)) = 1\) holds for every \((\bar{\sigma}_m) \in \bar{0}'\) as above with \(\bar{\sigma}_{m,1} = 0\), then we can choose \((K, M, b) := (1, M_1, b_1)\) and are done again.

Otherwise \(\lim G_n((\bar{\sigma}_m^{(2)})) = 0 =: b_2\) for some \((\bar{\sigma}_m^{(2)})\) with \(\lim_m \bar{\sigma}_m^{(2)} = \bar{0}\) and \(\sigma_m^{(2)} = 0\) for all \(m > M_1\). Similarly as above, there are \(n_2 \geq n_1\) and \(M_2 > M_1\) such that \(G_{n_2}((\bar{\sigma}_m)) = 0\) holds whenever \((\bar{\sigma}_m)\) satisfies \(\bar{\sigma}_m = \sigma_m^{(2)}\) for all \(m \leq M_2\); and either we are done with \((K, M, b) := (2, M_2, b_2)\), or we proceed to \(K := 3\) and \(b_3 := 1 - b_2\) and \(M_3 \geq M_2\); and so on.

Now the crucial observation asserts this iterative procedure to indeed terminate for some \(K\): because otherwise it would yield in the limit sequences \((\bar{\sigma}^{\omega}_m) \in \bar{0}'\) and \((n_K)_K\) such that \(G_{n_K}((\bar{\sigma}^{\omega}_m)) = 0\) for \(K\) even and \(1 = 1\) for \(K\) odd: contradicting that \(\lim_n G_n((\bar{\sigma}_m))\) was supposed to exist for every \((\bar{\sigma}_m) \in \bar{0}'\).

b) Again presume w.l.o.g. \(\bar{x} = \bar{0}\) and \(b = 0\). Take \(K\) and \(\bar{\sigma}_1, \ldots, \bar{\sigma}_M\) from a) and argue indirectly:

Failure of \(L = K\) means there is some \(\bar{y} \in X\) starting with \(K\) 0s such that the sequence \((\bar{\tau}_m^{(K)})_m := (\bar{\sigma}_1, \ldots, \bar{\sigma}_M, \bar{y}, \bar{y}, \ldots) \in \bar{y}'\) has \(\lim_n G_n((\bar{\tau}_m^{(K)})) = 1\); in particular \(G_{n_K}((\bar{\tau}_m)) = 1\) for some \(n_K, M_K \in \mathbb{N}\) and any \((\bar{\tau}_m)\) satisfying \(\bar{\tau}_m = \bar{\tau}_m^{(K)}\) for all \(m \leq M_K\).

Similarly, failure of \(L = K + 1\) means that, for some \(\bar{y} \in X\) starting with \(K + 1\) 0s, the sequence \((\bar{\tau}_m^{(K+1)})_m\), defined as \(\bar{\tau}_m^{(K+1)} = \bar{\tau}_m^{(K)}\) for \(m \leq M_K\) and \(\bar{\tau}_m^{(K+1)} = \bar{y}\) for \(m > M_K\), has \(\lim_n G_n((\bar{\tau}_m^{(K+1)})) = 1\) and \(G_{n_{K+1}}((\bar{\tau}_m^{(K+1)})) = 1\) for some \(n_{K+1}, M_{K+1} \in \mathbb{N}\) and any \((\bar{\tau}_m)\) satisfying \(\bar{\tau}_m = \bar{\tau}_m^{(K+1)}\) for all \(m \leq M_{K+1}\).

Again we argue that, for some finite \(L\), this iteration of successive failures terminates and hence leads to success: otherwise it would yield sequences \((\bar{\tau}_m^{(\omega)}), (n_L)_L, (M_L)_L\) such that it holds

- \(\bar{\tau}_m^{(\omega)} = \sigma_m\) for \(m \leq M\)
- \(\bar{\tau}_m, \ell = 0\) for all \(m \geq M_L\) and all \(\ell \geq L\)
- (in particular with \((\bar{\tau}_m^{(\omega)}) \in \bar{0}'\))
- \(\lim_n G_n((\bar{\tau}_m^{(\omega)})) = 1\)

contradicting a) that \(\lim_n G_n((\bar{\tau}_m^{(\omega)})) = 0\).

c) Notice that any two open balls \(U_i, U_j\) are either disjoint \(U_i \cap U_j = \emptyset\), subsets \(U_i \subseteq U_j\) or \(U_j \subseteq U_i\), or equal \(U_i = U_j\). So take\footnote{using the axiom of countable choice} some maximal set \(J \subseteq I\) of indices satisfying

\[\forall j, k \in J, j \neq k : U_j \nsubseteq U_k .\]

Then \((U_j)_{j \in J}\) has the desired properties.
d) Let $X := \text{dom}(g)$ and $G = \bigoplus_n G_n : X' \to \bigoplus_n \{0, 1\}^\omega$ denote a continuous $(i' \to i')$-realizer of $g$; that is such that, for every $(\bar{\tau}_m)_m \in X'$, \( \lim_n G_n((\bar{\tau}_m)_m) \) exists and belongs to $G(\lim_m \bar{\tau}_m)$. Now Claim b) means that there exists, to every $\bar{x} \in X$, some $L_x \in \mathbb{N}$ such that $g_2$, restricted to the open ball $U_\varepsilon := (x_1, \ldots, x_{L_x}) \circ \{0, 1\}^\omega$, admits a constant selection. According to c), these open restrictions $U_\varepsilon$ can be attained as pairwise disjoint, i.e. yield a locally constant selection of $g$.

e) Let $X := \text{dom}(f)$ and consider the mapping $g : X \to \{0, 1\}$ defined by $g(\bar{x}) \equiv 0$ if there is some $\bar{b} \in \{0, 1\}^\omega$ such that $0 \circ \bar{b} \in f(\bar{x})$ and $g(\bar{x}) \equiv 1$ if $1 \circ \bar{b} \in f(\bar{x})$ for some $\bar{b} \in \{0, 1\}^\omega$. Performing this operation on the continuous $(i' \to i')$-realization of $f$ yields a continuous $(i' \to i')$-realization of $g$. So apply d) to get a locally constant selection $h_1 : X \to \{0, 1\}$ of $g$; that is such that, for every $\bar{x} \in X$, $h_1(\bar{x}) \in \{0, 1\}$ has an extension to an infinite string in $f(\bar{x})$.

Let $U := h_1^{-1}(0) \subseteq X$ and $V := h_1^{-1}(1) \subseteq X$. Now consider the restriction $f_0 := f_h \mid_U$, where $sh : \{0, 1\}^\omega \to \{0, 1\}^\omega$, $(\sigma_1, \sigma_2, \sigma_3, \ldots) \rightsquigarrow (\sigma_2, \sigma_3, \ldots)$ denotes the left-shift operator. Repeating the above argument, separately for $f_0$ and for $f_1 := f_h \mid_V$, instead of $f$, yields a joint, locally constant function $h_2 : U \uplus V = X \to \{00, 01, 10, 11\}$ which

- refines $h_1$ in the sense that, for every $\bar{x} \in X$, $h_1(\bar{x})$ is a prefix of $h_2(\bar{x})$
- which in turn admits an extension to an infinite string in $f(\bar{x})$, that is to a (not necessarily continuous) selection of $f$.

Iteration yields for every $k$ a locally constant refinement $h_k : X \to \{0, 1\}^k$ of $h_{k-1}$ (and thus also of $h_\ell$ for each $\ell < k$) which admits an extension to a selection of $f(\bar{x})$. This process yields a function $h_\omega : X \to \{0, 1\}^\omega$ which

- refines all locally constant $h_k$, and thus is continuous;
- constitutes a selection of $f$.

Note that $\alpha \circ i' := \{0, 1\}^\omega \to A$ is a representation for $A$ (called the jump of $\alpha$) whenever $\alpha : \{0, 1\}^\omega \to A$ is. This new representation need not be admissible even if $\alpha$ is. SCHRÖDER has extended the Fact 2.62 to a generalized notion of admissibility covering certain topological spaces lacking a countable base [Schr06]. However, it turns out that we can do without his contributions in proving the following

**Lemma 4.53.** a) Let $\alpha$ and $\beta$ denote representations for $A$ and $B$, respectively. Every $(\alpha \circ i' \to \beta \circ i')$-continuous function $g : \subseteq A \to B$ is $(\alpha \to \beta)$-continuous.

b) $\alpha \preceq \beta$ implies $\alpha \circ i' \preceq \beta \circ i'$.

c) For $I = \mathbb{N}$, or for $I$ finite, let $\alpha_i$ be a representation for $A_i$, $i \in I$. Then the representations $(\prod_{i \in I} \alpha_i) \circ i' \text{ and } \prod_{i \in I} (\alpha_i \circ i')$ for $\prod_{i \in I} A_i$ coincide.

**Proof.** b) Let $F$ denote a computable string function converting $\alpha$-names to $\beta$-names. By Lemma 4.51b), $F$ has a computable $(i' \to i')$-realization $G : \subseteq \{0, 1\}^\omega \to \{0, 1\}^\omega$. This $G$ converts $(\alpha \circ i')$-names to $(\beta \circ i')$-names.

c) It suffices to treat the case $A_i = \{0, 1\}^\omega$ and $\alpha_i = i$ and $I = \mathbb{N}$. For $(\prod_{i \in I} i') \preceq (\prod_{i \in I} \alpha_i) \circ i''$ let $\bar{\sigma}_m \in \{0, 1\}^\omega$ be a sequence of infinite strings with respective $i'$-names $\bar{\tau}_{n,m}$, i.e. $\bar{\sigma}_m = \lim_n \bar{\tau}_{n,m}$; then apply Claim 4.49. Proceed similarly for the converse reduction.

a) Let $G : \subseteq \{0, 1\}^\omega \to \{0, 1\}^\omega$ be a continuous $(\alpha \circ i' \to \beta \circ i')$-realization of $g$; that is $G$ maps (encodings as infinite strings of) sequences $(\bar{\tau}_m)_m$ of infinite strings to (encodings as infinite strings of) sequences $(G_n((\bar{\tau}_m)_m))_n$ of infinite strings. By continuity of $G$ (and due to Claim 4.49 above),

$$F := \subseteq \{0, 1\}^\omega \to \{0, 1\}^\omega, \quad F(\lim_m \bar{\tau}_m) := \lim_n G_n((\bar{\tau}_m)_m)$$


is well-defined multivalued and \((i' \to i')\)-continuous \((G \text{ being a realization})\). Hence by Lemma 4.52, \(F\) admits a continuous selection. This constitutes an \((\alpha \to \beta)\)-realization of \(g\). \(\Box \)

Remark 4.54. Notice how Lemma 4.52e) entered in the proof to Lemma 4.53a); conversely, Lemma 4.53a) implies Lemma 4.52e) as follows:

Let multivalued \(f : \subseteq \{0,1\}^\omega \equiv \{0,1\}^* \) be \((i' \to i')\)-continuous. Equip \(B := \text{dom}(f)\) with the following artificial representation \(\beta\): A pair \((\bar{x}, \bar{y})\) is a \(\beta\)-name of \(\bar{x}\) iff \(\bar{x} \in \text{dom}(f)\) \& \(\bar{y} \in f(\bar{x})\) holds. Then \(g : \bar{x} \mapsto \beta(\bar{x}, f(\bar{x}))\) is single-valued (!), namely the identity on \(B\); and, by \((i' \to i')\)-continuity of \(f, g\) is \((i' \to \beta \circ i')\)-continuous. So take an \((i \to \beta)\)-continuous realization \(h\) according to Lemma 4.53(a). This \(h\) thus has the form \(h : B \ni \bar{x} \mapsto (\bar{x}, h_2(\bar{x}))\) with continuous single-valued \(h_2 : B \to \{0,1\}^\omega\) satisfying \(h_2(\bar{x}) \in f(\bar{x})\): the desired selection.

For concrete representations, the next remark may provide better intuition about their respective jumps in terms of ‘stabilizing’ double sequences and sequences with ‘revocations’.

Remark 4.55. Let representation \(\alpha : \subseteq \{0,1\}^\omega \to A\) arise\(^1\) from a notation \(\nu : \subseteq \{0,1\}^* \to X\) in that every \(\alpha\)-name is \((\nu^\omega\text{-name of})\) some sequence \((x_n)_n\) in \(X\). Also suppose that the \(\nu\)-equivalence problem \(\{(u, v) \mid u, v \in \text{dom}(\nu) \& \nu(u) = \nu(v)\}\) is recursive.

a) An \(\alpha \circ i'\)-name for \(a \in A\) is uniformly equivalent to a \(\prod^\omega \nu\)-name of a double sequence \((x_{n,m})\) in \(X\) satisfying

\[
\begin{align*}
\forall n & \exists M \quad x_{n,M} = x_{n,M+1} = \ldots =: x_{n,\infty} \quad ((x_{n,m})_m \text{ stabilizes for every } n) \\
\exists & \in \mathbb{N} \quad x_{n,\infty} = x_n \text{ for each } n \in \mathbb{N}.
\end{align*}
\]

b) Let \(\tilde{\nu}\) denote an extension of \(\nu\) to \(\tilde{X} := X \cup \mathbb{N}\). Define \(\tilde{\alpha} : \subseteq \{0,1\}^\omega \to A\) to encode \(a \in A\) as (any \(\tilde{\nu}\)-name of) some sequence \((\tilde{x}_n)_n\) in \(\tilde{X}\) where \(\tilde{x}_n = m \in \mathbb{N}\) means (in informal terms):

\text{Revoke member } \tilde{x}_n \text{ of this representation of } A \text{ which also uniformly equivalent to } \alpha \circ i'.

Proof. a) We begin with the reduction to \(\alpha \circ i'\). For \(m \in \mathbb{N}\), let \(\bar{x}_m\) denote a \(\nu^\omega\)-name of \((x_{n,m})_n\). Since \(\nu\)-equivalence is decidable, one can effectively obtain an encoding of this sequence in which identical elements \(x\) have identical \(\nu\)-names. Property i) then implies that \((\bar{x}_m)_m\) converges; and by ii), its limit is an \(\alpha\)-name of \(a\).

Concerning the converse reduction from \(\alpha \circ i'\), suppose that \(\lim_m \bar{x}_m =: \bar{s}\) is a \(\nu^\omega\)-name of \((x_n)_n\). Since the decidability of \(\nu\)-equivalence implies the decidability of \(\text{dom}(\nu)\), one can, for each \(m \in \mathbb{N}\), start decoding \(\bar{x}_m\) (which is in general not a valid \(\nu^\omega\)-name) into a finite sequence \((x_{1,m}, \ldots, x_{N_m,N_m})\) in \(x\) up to the maximal \(N_m \leq m\), for which the corresponding initial segment of \(\bar{x}_m\) is a valid \(\nu^{N_m}\)-name. Since \(m \mapsto N_m\) is computable, we may furthermore effectively extend and output arbitrary values for \(x_{n,m}, n > N_m\).

For \(n \in \mathbb{N}\), let \(s_n \leq t_n \in \mathbb{N}\) denote the respective starting and ending positions of the substring of \(\bar{s}\) \(\nu\)-encoding sequence member \(x_n\). From \(\bar{s} = \lim_m \bar{x}_m\) it follows that, for every \(n\), there is some \(M \in \mathbb{N}\) such that the initial segment of \(\bar{x}_m\) up to \(t_n\) coincides with that of \(\bar{s}\) for all \(m \geq M\). In particular, this initial segment of \(\bar{x}_m\) constitutes a valid \(\nu^M\)-name for \((x_{1,\ldots,n})\); hence, according to the above algorithm, it holds \(x_{n,m} = x_n\) for \(m \geq M\), i.e., the output satisfies both i) and ii).

\(^{18}\)I am grateful to Vasco Brattka for pointing out that gap in a previous version of this proof
\(^{19}\)This is the case for example for many standard representations of a computable topological space \(\text{Web0}\)
\(^{20}\)DEFINITIONS 3.2.1 and 3.2.2. Strictly speaking, every representation \(\alpha\) arises in the above way from some—more or less artificial—notation \(\nu\)...
\(^{21}\)Without this condition, that is when revocations of revocations are permitted, we arrive at representations equivalent to iterated jumps; cf. Section 4.4.5.
b) We first show “\( \hat{\alpha} \preceq \alpha \circ i’ \)”: For \( m \in \mathbb{N} \), consider (and \( \nu^* \)-compute) the finite subsequence \((x_{m,n})_{n \leq N_m}\) of ‘tentatively surviving elements’, i.e. where all revoke-commands have been applied and removed. Notice that the length \( N_m \) of this subsequence is computable. This permits us to effectively extend it (in fact arbitrarily) to an infinite sequence \((x_{m,n})_{n \leq m}\). We claim that this satisfies i) and ii) in a) and is therefore computably convertible to an \( \alpha \circ i'' \)-name of the same element. Indeed, by the above condition on survival, it holds \( N_m \to \infty \) for \( m \to \infty \) because the ultimately prevailing elements must enumerate an infinite subset of \( X \). Moreover, precisely every surviving element eventually becomes and remains tentatively surviving as soon as all its (ultimately revoked) predecessors have been removed, that is beyond some finite \( m \).

For “\( \alpha \circ i' \preceq \hat{\alpha} \)”, again invoke a) to obtain (a \( \nu^* \)-name of) a double sequence \((x_{n,m})\) in \( X \) satisfying i) and ii). For each \( m \), let \( N_{m+1} := \min \{ n : x_{n,m} \neq x_{n,m+1} \} \) denote the first position \( n \) where \((x_{n,m})\) differs from \((x_{n,m+1})\); \( N_1 := 0 \), see also Figure 4.10. The integer sequence \((N_m)\) is computable because we required the \( \nu \)-equivalence problem to be decidable. Now apply the following algorithm:

For \( m = 1, 2, \ldots \)

I) in case \( N_{m+1} > N_m \), append (\( \nu \)-names of) \( x_{N_m,m+1}, \ldots, x_{N_{m+1}+1,m+1} \) to the output;

II) in case \( N_{m+1} < N_m \), revoke (the indices in the output stream of) \( x_{N_m+1,m+1}, \ldots, x_{N_{m-1},m} \).

Since every \( x_{n,\infty} \) stabilizes beyond some \( M(n) \), \((N_m)\) is unbounded; hence \( x_{n,\infty} \) eventually gets output in I) at \( m = M(n) \) and not revoked by II). Conversely, to every \( x_{n,m} \neq x_{n,\infty} \), if output at I), there corresponds some \( M \) at which it changes and is therefore revoked in II).

4.4.3 Examples of Known Jumps

We now reveal several known representations to be of the form \( \alpha \circ i' \) for some (also well-known) \( \alpha \).

**Proposition 4.56.** \( \rho \circ i' \equiv \rho' \).

In combination with Remark 4.50 this implies Proposition 4.8.

**Proof.** By Remark 4.55 a), a \( (\rho \circ i') \)-name for \( x \in \mathbb{R} \) can be regarded as a sequence of rational sequences stabilizing (elementwise) to a fast converging Cauchy sequence \((q_{(n,\infty)})\); that is a double
sequence \((q_{(n,m)})\) in \(Q\) such that
\[\forall n \exists m_0 \forall m \geq m_0 : q_{(n,m)} = q_{(n,m_0)} \land |x - q_{(n,m_0)}| \leq 2^{-n}.\]

\[\preceq\]: For each \(m\), let \((q_{(1,m)}, q_{(2,m)}, \ldots, q_{(N_m,m)})\) denote the longest initial part of \((q_{(1,m)}, \ldots, q_{(m,m)})\) satisfying
\[|q_{(n,m)} - q_{(n',m')}| \leq 2^{1-n} \forall 1 \leq n \leq n' \leq N_m.\] (4.11)

Since \((q_{(n,\infty)})_n\) is a \(p\)-name and due to the eventual stabilization, \(N_m \to \infty\) as \(m \to \infty\). Also, the sequence \((N_m)_m\) is computable from the above input. Consider the following algorithm, starting with empty output tape:

For each \(m = 1, 2, \ldots\), test whether the initial parts of \(q_{(1,m)}\) and \(q_{(2,m-1)}\) up to \(N_m\) coincide: \((q_{(1,m)}, \ldots, q_{(N,m,m)}) = (q_{(1,m)}, \ldots, q_{(N,m,m-1)}).\) (For notational convenience, set \(q_{n,0} := \infty\) and \(N_0 := 0\).) If so, then obviously \(N_m \geq N_m-1\); so append (the possibly empty sequence) \((q_{(N_{m-1},m)}, \ldots, q_{(N_{m},m)})\) to the output. Otherwise let \(n_m\) be maximal with \((q_{(1,m)}, \ldots, q_{(n_m,m)}) = (q_{(1,m)}, \ldots, q_{(n_m,m-1)})\); obviously \(n_m < N_m\), so append \((q_{(n_m,m)}, \ldots, q_{(N_{m},m)})\) to the output in this case.

It remains to be shown that this yields a valid \(p\)-name for \(x\). Let \(\epsilon = 2^{1-n}\). Then \(|q_{(n,\infty)} - q_{(n',\infty)}| \leq \epsilon\) for all \(n' \geq n\) because \(q_{(n,\infty)}\) constitutes a \(p\)-name. Moreover, due to stabilization, there exists some maximal \(m\) with \(q_{(n,m)} \neq q_{(n,m-1)}\). During phase no. \(m\) corresponding to that last change, the above algorithm will detect \(n_m < N_m\) and thus output (a finite sequence beginning with) \(q_{(n,m)}\). Moreover, as \(q_{(n,\cdot)}\) afterwards no longer changes, all elements \(q_{(n',m')}\) appended subsequently will have \(n' \geq n\) and \(m' \geq m\); in fact \(N_{m'} \geq n_{m'} \geq n\), hence \(|q_{(n',m')} - q_{(n,m)}| \leq \epsilon\) because \(q_{(n,m')} = q_{(n,m)}\) and due to Equation (4.11). Therefore the output constitutes a (naive) Cauchy sequence converging to \(x\).

\[\succeq\]: Let \((q_n)_n\) be a sequence in \(Q\) ultimately converging to \(x\). There exists an increasing sequence \((n_m)_m\) in \(N\) such that
\[\forall k \geq n_m : |q_{n_m} - q_k| \leq 2^{-m-1}.\] (4.12)

The subsequence \((q_{n_m})_m\) constitutes a \(p\)-name for \(x\). For each single \(m\), Condition (4.12) can be falsified (formally: is co-r.e. in the input). A Turing machine is therefore able to iteratively try for \(n_m\) all integer values from \(n_m\) on and fail only finitely often for each \(m\).

Trial no. \(\ell\) thus yields a sequence \((n_{\ell,m})_{m \leq \ell}\) of length \(\ell\) such that, for each \(m\), \(n_{\ell,m}\) eventually stabilizes to \(n_{m}\) satisfying (4.12). By artificially extending each finite sequence to an infinite one, we obtain a \(p \circ \ell\)-name for \(x\).  

\textbf{Proposition 4.57.} \(\rho < \circ \ell \equiv \rho^\ell <\).

\textbf{Proof.} A \((\rho < \circ \ell)\)-name for \(x \in R\) amounts according to Remark 4.55 to a sequence of rational sequences eventually stabilizing (elementwise) to a sequence approaching \(x\) from below, that is a double sequence \((q_{(n,m)})\) in \(Q\) such that
\[\forall n \exists m_0 = m_0(n) \forall m \geq m_0 : q_{(n,m)} = q_{(n,m_0)} \land x = \sup_{n} q_{(n,m_0(n))}.\]

\[\preceq\]: Since the limit (which exists) coincides with the least accumulation point, we have
\[x = \sup_{n} \lim_{m} q_{(n,m)} = \sup_{n} \sup_{m} \inf_{j} q_{(n,m)} = \sup_{m} \inf_{j} \left\{ q_{(n,m)} : m \geq j \right\} = \inf_{m} \sup_{j} q_{(n,m)} = \inf_{m} \sup_{j} q_{(n,m)} \]

and thus deduced a \(\rho^\ell\)-name for \(x\).

\[\succeq\]: Let \((q_{(n,m)})\) be the given double sequence in \(Q\) with \(x = \sup_{n} \inf_{m} q_{(n,m)}\). We may suppose that all single sequences \(q_{(n,\cdot)}, n \in N\), are monotonically nonincreasing; and that the single sequence \(\{ \inf_{m} q_{(n,m)} \}_n\) is nondecreasing: by proceeding (in either order) from \(q_{(n,m)}\) to
min\(k\leq m\) \(q_{(n,k)}\) and to \(\max_{\ell\leq n} q_{(\ell,m)}\). Moreover, one can ensure each single sequence \(q_{(n,\cdot)}\) to eventually stabilize, thus yielding a \(\rho_\ll\circ\ell'\)–name of \(x\): Consider for \(m\in\mathbb{N}\) the function \([\cdot]_m: \mathbb{Q}\rightarrow\mathbb{Q}\) mapping every rational to the next lower dyadic rational having denominator \(2^{-m}\); formally: \(a/b\mapsto [[2^m a/b]]\) where \([\cdot]=[[\cdot]]\) denotes the usual floor function on integers. Then proceeding from \(q_{(n,m)}\) to \([q_{(n,m)}]_m\) satisfies this requirement without affecting \(x=\sup_{n} m \inf_{m} q_{(n,m)}\).

**Proof.** This result includes \(\text{Theorem 16}\). The proof proceeds similarly to that of Proposition \(4.56\) because Equations \((4.11)\) and \((4.12)\) are still decidable and co-r.e. when replacing rational numbers \(q\) with rational polynomials \(Q=\sum_{i=0}^{d}q_i x^i\) and absolute value \(|q|\) with maximum norm \(\|Q\|\):

**Lemma 4.59.** The following set of integers is (classically) decidable:

\[
\{ \langle n, d, q(0,\ldots,0),\ldots q(d,\ldots,d), b \rangle : n,d\in\mathbb{N}, q_\mathbb{R}, b\in\mathbb{Q}, \forall 0\leq x_1,\ldots, x_n \leq 1 : \left| \sum_{i=0}^{d} q_\mathbb{Q} x_1^i \cdots x_n^i \right| \leq b \} =: \Phi(q_\mathbb{Q},\ldots,q_\mathbb{R},b)
\]

**Proof.** Abbreviate \(\Phi(x) := \sum_{i=0}^{d} q_i x^i\). By separately treating +\(Q\) and −\(Q\), we may drop the absolute value. Moreover, strict inequalities \(\|Q\|<b^r\) and \(\|Q\|>b^r\) are semi-decidable: by continuity it suffices to evaluate \(Q\) on a countable dense subset like the rationals. It remains to be decided whether \(Q-b\) admits a real root in \([0,1]^n\). In the univariate case \(n=1\), this can be performed by means of root bounds \(\text{BFMS00}\), recall Section 2.7. In the multivariate case, we content ourselves with a less constructive argument: Quantifier Elimination \(\text{Fact 2.13}\) yields a quantifier-free formula \(\Psi\) in \((q_\mathbb{Q},\ldots,q_\mathbb{R},b)\) with constants from \(F:=\mathbb{Q}\) equivalent to \(\Phi(q_\mathbb{Q},\ldots,q_\mathbb{R},b)\).

### 4.4.4 Jump of Set Representations \(\theta_\ll\) and \(\psi_\gg\)

We now establish a nice characterization of the jump \(\theta_\ll\circ\ell'\) of the inner representation of open sets \(\text{Definition:Represenation}\). For reasons of notational convenience we turn to complements and consider closed real subsets with the representation \(\psi_\gg\) \(\text{Section 4.1.2}\).

**Definition 4.60.** Let \(\psi_\gg\) denote the representation of closed real sets encoding \(A\subseteq\mathbb{R}^k\) as a sequence \((\tilde{q}_n)\) of rational vectors whose set of accumulation points coincides with \(A\).

Let \(\psi_\gg\) encode \(A\subseteq\mathbb{R}^k\) as an enumeration of a set \(Q\) of rational vectors whose accumulation points are precisely the elements of \(A\).

Recall that a sequence may repeat elements in order to establish an accumulation point but a set cannot.

**Theorem 4.61.** It holds \(\psi_\gg\circ\ell' = \psi_\gg\equiv \psi_\gg\).

**Proof.** In view of Remark \(4.55\), a \(\psi_\gg\circ\ell'\)–name for \(A\subseteq\mathbb{R}^d\) is (equivalent to) a double sequence of open rational balls \(B_{n,m}\) such that, for every \(n\) and some \(M=M(n)\), \(B_{n,M} = B_{n,M+1} = \ldots = B_{n,\infty}\) and \(A = \mathbb{R}\setminus \bigcup_{n} B_{n,\infty}\).

\(\psi_\gg\circ\ell' < \psi_\gg\): Given \((B_{n,m})\) as above, calculate \(\hat{N}(m) := \max\{N\leq m : B_{n,m} = B_{n,m+1}\ \forall n \leq N\}\) and let \((\tilde{q}_{(m,k)})_k\) enumerate (without repetition) the set \(\mathbb{Q}^d \setminus \bigcup_{n=1}^{\hat{N}(m)} B_{n,m}\) for each \(m \in \mathbb{N}\). Then this sequence \((\tilde{q}_i)_{i}\) is a \(\psi_\gg\)–name for \(A\):

- For \(\bar{x} \notin A\), there exists \(N\) such that \(\bar{x} \in B_{N,\infty}\). By finite stabilization, \(\hat{N}(m) \geq N\) for \(m \geq M(N)\). Therefore \(\tilde{q}_{(m,k)} \in B_{N,\infty}\) can hold only for \(m < M(N)\), i.e., finitely often; hence \(\bar{x}\) is not an accumulation point of \((\tilde{q}_i)\).
4.4. REVISING TYPE-2 COMPUTATION

- Suppose \( \bar{x} \in A \), i.e. \( \bar{x} \not\in \bigcup_n B_{n,\infty} \supseteq \bigcup_{n=1}^N B_{n,m} \) for every \( N \) and \( m \geq M(N) \). In particular for \( m \geq M(N) \), it holds \( x \not\in \bigcup_{n=1}^{N(m)} B_{n,m} \) and by construction plus Claim 4.62 there is some \( k_m \) with \( |\tilde{q}_{(m,k_m)} - \bar{x}| \leq 2^{-m} \). So \( \bar{x} \) is an accumulation point of \( (\tilde{q}_n) \).

\( \psi_k \leq \psi \): We are given a rational sequence \( (\tilde{q}_n) \). Starting with \( Q = \emptyset \), add, inductively for each \( \ell \in \mathbb{N} \), a rational vector not yet in \( Q \) and closer to \( \tilde{q}_\ell \) than \( 2^{-n} \). Indeed, the finiteness of \( Q \) at each step ensures: \( \exists \tilde{p} \in Q^d \cap ((\tilde{q}_\ell - 2^{-n}, \tilde{q}_\ell + 2^{-n}) \setminus Q) \).

\( \psi_k \leq \psi \circ \psi' \): Let \( (B_{k,0})_n \) denote an effective enumeration of all open rational balls. Given \( (\tilde{q}_\ell) \), calculate inductively for \( m \in \mathbb{N} \) the subsequence \( (B_{n,m+1})_n \) of \( (B_{n,m})_n \) containing those balls disjoint to \( \{\tilde{q}_1, \ldots, \tilde{q}_m\} \).

- For \( \bar{x} \) accumulation point of \( (\tilde{q}_n) \) and \( B_{k,0} \) an arbitrary open rational ball containing \( \bar{x} \), there is some \( \tilde{q}_M \in B_{k,0} \). By construction, this \( B_{k,0} \) will not occur in \( (B_{n,m+1})_n \) for \( m \geq M \). Hence \( \bar{x} \not\in \mathbb{R}^d \setminus \bigcup_n B_{n,\infty} = A \).

- If \( \bar{x} \) is contained in some ball \( B_{k,0} \) which ‘prevails’ as \( B_{n,\infty} \), it cannot contain any \( \tilde{q}_m \) by construction. Therefore \( \bar{x} \) is not an accumulation point.

Claim 4.62. Let \( N \in \mathbb{N} \), \( \bar{u}_n, \bar{v}_n \in Q^d \), and \( \bar{x} \in \mathbb{R}^d \) with \( \bar{x} \not\in \bigcup_{n=1}^N (\bar{u}_n, \bar{v}_n) \). Then, for every \( \epsilon > 0 \), there is some \( \tilde{q} \in Q^d \setminus \bigcup_{n=1}^N (\bar{u}_n, \bar{v}_n) \) such that \( ||\bar{x} - \tilde{q}|| \leq \epsilon \).

Proof. If \( \bar{x} \in Q^d \) then let \( \tilde{q} := \bar{x} \). Otherwise \( \bar{x} \) belongs to the open set \( \mathbb{R}^d \setminus \bigcup_{n=1}^N (\bar{u}_n, \bar{v}_n) \) in which rational numbers lie dense.

Corollary 4.63. A subset of \( X = \mathbb{R}^k \) belongs to the relativized class \( \Pi_1[0'] \) iff it is the set of accumulation points of a recursive sequence in \( \mathbb{Q}^k \).

When the representation \( \psi' \) from Definition 4.60 is restricted to subsets of \([0,1]^k\) (rather than of \(\mathbb{R}^k\)), one may in general not require the sequence \( (\tilde{q}_n) \) of rational vectors to remain in \([0,1]^k\); otherwise the empty set has no encoding. (Technically speaking, in our proof of “\(\psi \circ \psi' \leq \psi'_n\)”, the set \([0,1]^d \cap Q^d \setminus \bigcup_{n=1}^{N(m)} B_{n,m} \) may be empty...)

For non-empty closed subsets of \([0,1]^k\), on the other hand, this additional requirement can be achieved effectively:

Lemma 4.64. Given a sequence \( (\tilde{q}_n) \) in \( Q^d \) with at least one accumulation point in \([0,1]^d\), one can effectively obtain a sequence \( (\tilde{p}_n) \) in \( Q^d \cap [0,1]^d \) whose accumulation points coincide with those of \( (\tilde{q}_n) \) contained in \([0,1]^d\).

Proof. Simply dropping from \( (\tilde{q}_n) \) all elements not in the cube \([0,1]^k\) maintains all accumulation points in its interior but may miss some on its boundary. We show how to effectively intersect both the sequence \( (\tilde{q}_n) \) and its accumulation points with the halfspace \([0,\infty) \times \mathbb{R}^{d-1}\). Further similar application to \((-\infty,1] \times \mathbb{R}^{d-1}, \mathbb{R} \times [0,\infty) \times \mathbb{R}^{d-2}, \mathbb{R} \times (-\infty,1] \times \mathbb{R}^{d-2}, \ldots, \mathbb{R}^{d-1} \times (0,\infty), \mathbb{R}^{d-1} \times (-\infty,1] \) yields the claim.

Start (phase \#0) by passing through from input to output all \( q_n \geq 0 \) and dropping all \( q_n < -1 \) until encountering some \(-1 \leq q_n < 0\); in this case, instead of this element, append 0 to the output and enter phase \#1: again, all \( q_n \geq 0 \) are simply transmitted, but now \( q_n < \frac{1}{2} \) are dropped until encountering some \(-\frac{1}{2} \leq q_n < 0 \) which is transformed to 0 and initiates phase \#2. Generally phase \#j passes on \( q_n \geq 0 \), skips \( q_n < -2^{-j} \), and waits for some \(-2^{-j} \leq q_n < 0 \).
Straightforward inductive application of Remark 4.50 shows that \( \iota^{(d)} \)-computability is equivalent to \( \iota \)-computability relative to \( \emptyset^{(d)} \). If \( F \) and \( G \) are partial \( (\iota \rightarrow \iota') \)-computable string functions, then their composition \( G \circ F \) is \( (\iota \rightarrow \iota') \)-computable by Lemma 4.51\( a \).\( b \).

Theorem 4.66. For each \( d \in \mathbb{N} \), it holds \( \rho^{(d)} \equiv \rho \circ \iota^{(d)} \) and \( \rho^{(d)}_{<} \equiv \rho_{<} \circ \iota^{(d)} \).

Proof. The induction start \( d = 1 \) has been treated in Propositions 4.56 and 4.57. Since a \( \rho^{(d+1)} \)–name of \( x \in \mathbb{R} \) is the join of \( \rho^{(d)} \)–names of elements \( x_{n} \) with \( x = \lim_{n} x_{n} \), Proposition 4.56 together with Lemma 4.53\( a \) also provides the induction step; similarly for \( \rho^{(d+1)}_{<} \).

As a consequence we obtain by inductive application of Lemmas 4.51\( b+c \) and Lemma 4.53\( a \) the following extension (and simplification of the proofs) of Theorems 4.26 and 4.30 which also includes Corollary 4.49.

Corollary 4.67. Fix \( f : X \rightarrow \mathbb{R} \).

- a) If \( f \) is \( (\rho^{(d)} \rightarrow \rho^{(d)}) \)-continuous, then it is continuous.
  - If \( f \) is \( (\rho^{(d)} \rightarrow \rho^{(d)}) \)-computable, then it is also \( (\rho^{(d+1)} \rightarrow \rho^{(d+1)}) \)-computable.
  - If \( f \) is \( (\rho^{(k)} \rightarrow \rho^{(k+d)}) \)-continuous, then it is \( \Sigma_{d+1} \)-measurable.

- b) If \( f \) is \( (\rho^{(d)} \rightarrow \rho^{(d)}) \)-continuous, then it is lower semi-continuous.
  - If \( f \) is \( (\rho^{(d)} \rightarrow \rho^{(d)}) \)-computable, then it is also \( (\rho^{(d+1)} \rightarrow \rho^{(d+1)}) \)-computable.
  - If \( f \) is \( (\rho^{(k)} \rightarrow \rho^{(k+d)}) \)-continuous, then it is \( \Sigma_{d+1} \) lower semi-measurable.

- c) If \( f \) is \( (\rho^{(d)} \rightarrow \rho^{(d)}) \)-continuous, then it is nondecreasing.
  - If \( f \) is \( (\rho^{(d)} \rightarrow \rho^{(d)}) \)-computable, then it is also \( (\rho^{(d+1)} \rightarrow \rho^{(d+1)}) \)-computable.

We finally turn to uniform variants of the claims considered in Theorem 4.4.

Theorem 4.68. For \( X = \mathbb{R}^{k} \) it holds

- a) \( \theta^{k} \circ \iota' \preceq \delta_{\Sigma_{1}} \Sigma^{k} \)
- b) \( \theta^{k} \circ \iota' \not\preceq \delta_{\Pi_{2}} \Sigma^{k} \)
- c) \( \delta_{\Pi_{2}} \Sigma^{k} \preceq \theta^{k} \circ \iota'' \)

Recall that \( \Sigma^{k} \) denotes the class of open subsets of \( X \) for which \( \theta^{k} \equiv \delta_{\Sigma^{k}} \) is a representation. Of course the restriction of \( \delta_{\Sigma} \) and \( \delta_{\Pi_{2}} \) is thus necessary for the reductions to make sense.

Proof. a) Non-reducibility from \( \delta_{\Sigma_{2}} \Sigma^{k} \) to \( \theta^{k} \circ \iota' \) follows from Proposition 4.7, so let us turn to the reverse positive reducibility claim. A \( (\theta \circ \iota') \)-name for \( U \in \mathcal{O}(X) \) consists of two rational double sequences \( (r_{(m,n)}) \) and \( (r_{(m,n)})' \) such that each single sequence \( c_{(m,n)} \) and \( r_{(m,n)} \), \( n \in \mathbb{N} \), eventually stabilizes to some \( c_{(m,n)} \) and \( r_{(m,n)} \) where \( U = \bigcup_{n \in \mathbb{N}} B(c_{(n,\infty)}, r_{(n,\infty)}) \) and \( B(c, r) \) denotes the open ball with center \( c \) and radius \( r \).

Both representations admit of effective countable unions; it therefore suffices to show the reduction on open rational balls, that is, we may assume w.l.o.g. \( U = B(c_{m}, r_{m}) \) for all \( m \geq m_{0} \). So let

\[
A_{n} := \begin{cases} \overline{B}(c_{m}, r_{m} \cdot (1 - 2^{-n})) & (c_{k}, r_{k}) = (c_{k+1}, r_{k+1}) \forall k \geq m \\ \emptyset & \text{otherwise} \end{cases},
\]

hence \( U = \bigcup_{n} A_{n} \). Moreover the closed set \( A_{n} \) can be \( \psi \)-computed, uniformly in \( n \) and the given sequences \( (c_{m}) \) and \( (r_{m}) \), start generating \( \overline{B}(\ldots) \); if the co-r.e. condition \( \psi_{k} \geq m^{n} \) eventually turns out to fail, the machine may still revert to a \( \psi \)-name for \( \emptyset \) by adding further negative information to the output. We thus obtain a \( \delta_{\Sigma_{2}} \)–name for \( U \).
b) Suppose a Type-2 Machine $\mathcal{M}$ performs the reduction described above. Feed into it the constant double-sequence $B_{n,m} := (0, 1)$, obviously a valid $\theta_\prec \circ t \prec$-name for $U_1 = \bigcup_n B_{n,\infty} = (0, 1)$. Due to the presumption, the double sequence sequence $(B'_{n,m})$ of open rational intervals output satisfies $A := \left[\frac{1}{2}, \frac{2}{3}\right] \subseteq U_1 = \bigcap_n \bigcup_m B_{n,m}$. In particular, by compactness, $A \subseteq \bigcup_{m=1}^{M_1} B'_{1,m}$ for some $M_1 \in \mathbb{N}$. Up to this output, $\mathcal{M}$ has read only finitely many $B_{n,m}$; up to $(n_1, m_1)$, say.

Now modify (some of the unread part of) the input to $B_{n,m} := \emptyset$ for $n \leq n_1$ and $m > m_1$. Observe that, for each $n$, the sequence $(B_{n,m})_m$ is still finitely stabilizing and hence a valid $\theta_\prec \circ t \prec$-name for $(0, 1) \supseteq A$. By hypothesis, it leads $\mathcal{M}$ to extend its output $(B'_{n,m})$ such that additionally $A \subseteq \bigcup_{m=2}^{M_2} B'_{2,m}$ for some $M_2 \in \mathbb{N}$.

Continuing iteratively, we ultimately obtain a $(B_{n,m})$ with $B_{n,m} = \emptyset$ for some $m_n \in \mathbb{N}$ (i.e. a $\theta_\prec \circ t \prec$-name for $\emptyset$) whose input leads $\mathcal{M}$ to output $(B'_{n,m})$ with $A \subseteq \bigcup_{m=1}^{M_3} B'_{n,m}$ for each $n$ and some $M_3$; in particular, $\emptyset \neq A \subseteq \bigcap_n \bigcup_m B'_{n,m}$, that is the output cannot be a $\delta T_1 \prec$-name for $\emptyset$: contradiction.

c) Given a double-sequence $(B_{n,m})$ of open rational balls with $U = \bigcap_n \bigcup_m B_{n,m}$ open. Let $(B_K)$ denote an enumeration of all (non-empty) open rational balls. Define the triple-sequence

$$B'_{K,N,M} := \begin{cases} B_K : \forall n \leq N : B_K \subseteq \bigcup_{m=1}^{M} B_{n,m} \\ \emptyset : \text{otherwise} \end{cases}$$

Firstly, this is easy to compute from the input. Secondly, we claim that this is a valid $(\theta_\prec \circ t' \circ t')$-name: For fixed $(K, N)$ either there exists some $M$ such that $\overline{B}_K \subseteq \bigcup_{m=1}^{M} B_{n,m}$; then the sequence $(B'_{K,N,M})_m$ stabilizes to $B_K$ beyond $M$; or there is no such $M$, in which case the sequence is constantly $\emptyset$. Let $B'_{K,\infty}$ denote this (finitely attained) limit and remember that it equals either $\emptyset$ or, independently of $N$, $B_K$. Then, similarly, the sequence $(B'_{K,N,\infty})_N$ is either constantly $B_K$ (namely in case that, for every $N$, there exists some $M$ such that $\overline{B}_K \subseteq \bigcup_{m=1}^{M} B_{n,m}$): or eventually stabilizes to $\emptyset$.

Thirdly, this $(\theta_\prec \circ t' \circ t')$-name encodes (i.e. $\bigcup_K B'_{K,\infty}$ coincides with) the set $U$: If $B'_{K,\infty} \neq \emptyset$ then, by construction, it equals $B_K$ where for all $N$ there exists some $\overline{B}_K \subseteq \bigcup_{m=1}^{M} B_{n,m}$; i.e. $B_K \subseteq \bigcap_N \bigcup_m B_{n,m} = U$. Conversely, if $x \in U$, then $x \in B_K \subseteq \overline{B}_K \subseteq U$ for some $K$ because $U$ is open; moreover, as $\overline{B}_K$ is compact, $\overline{B}_K \subseteq U \subseteq \bigcup_M B_{N,M}$ implies: $\forall N \exists M : \overline{B}_K \subseteq \bigcup_{m=1}^{M} B_{n,m}$. Therefore $x \in B_K = B'_{K,\infty}$ by construction. \hfill \Box

### 4.5 Type-2 Nondeterminism

Our failure so far to hypercompute discontinuous functions as discussed in Section 4.3.5 hinges on Fact 4.25 and Theorem 4.26. Closer examination of their proofs in turn reveals that crucially rely on the underlying Turing Machines to behave deterministically. It is therefore an obvious step to consider nondeterminism as a putative means of effective evaluation of discontinuous real functions like Heaviside’s.

In the discrete (i.e., Type-1) setting, where any computation is required to terminate, the finitely many possible choices of a nondeterministic machine can of course be simulated by a deterministic one—though even here, this is subject to the important condition that all paths of the nondeterministic computation indeed terminate, cf. [BST89]. In contrast, a Type-2 computation realizes a transformation from/to infinite strings and is therefore a generally non-terminating process. Therefore, nondeterminism here involves an infinite number of guesses which it turns out cannot be simulated by a deterministic Type-2 machine.

We also point out that nondeterminism was already revealed to be not only a useful but indeed the most natural concept of computation on $\Sigma^\omega$. More precisely, Büchi extended automaton theory from finite to infinite strings and proved that here, as opposed to deterministic ones,
non-deterministic finite automata are closed under complementation and thus the more appropriate model of computation. Since automata and Turing Machines constitute the top and bottom

<table>
<thead>
<tr>
<th>Chomsky Level</th>
<th>(\Sigma^*)</th>
<th>(\Sigma^\omega)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3: regular</td>
<td>Finite Automata</td>
<td>Büchi Automata (nondeterministic)</td>
</tr>
<tr>
<td>2: context-free</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: context-sensitive</td>
<td>(Type-1) Turing Machines</td>
<td>non-deterministic Type-2 Machines</td>
</tr>
<tr>
<td>0: unrestricted</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.11: Models of Computation in Chomsky’s Hierarchies over finite/infinite strings

levels, respectively, of Chomsky’s Hierarchy of classical languages \(L \subseteq \Sigma^*\) (Type-1 setting), we suggest that over infinite strings \(\Sigma^\omega\) (Type-2 setting) both their respective counterparts, that is Büchi Automata and Type-2 Machines be considered nondeterministically; compare Figure 4.11.

The concept of nondeterministic computation of a function \(f : \subseteq \Sigma^* \rightarrow \Sigma^*\) (as opposed to a decision problem) is taken from the famous Immerman–Szelepcsényi Theorem in computational complexity; cf. for instance the paragraph preceding Theorem 7.6 in [Papa94]. For \(\bar{x} \in \text{dom}(f)\), some computing paths of the machine \(M\) may fail by leading to rejecting states, as long as

1) there is an accepting computation of \(M\) on \(\bar{x}\) and

2) every accepting computation of \(M\) on \(\bar{x}\) yields the correct output \(f(\bar{x})\).

This notion extends straightforwardly from Type-1 to the Type-2 setting:

**Definition 4.69.** Let \(A\) and \(B\) be sets with respective representations \(\alpha : \subseteq \Sigma^\omega \rightarrow A\) and \(\beta : \subseteq \Sigma^\omega \rightarrow B\). A function \(f : \subseteq A \rightarrow B\) is called nondeterministically \((\alpha \rightarrow \beta)\)-computable if some nondeterministic one-way Turing Machine \(M\),

- upon input of any \(\alpha\)-name \(\bar{\sigma} \in \Sigma^\omega\) for some \(a \in \text{dom}(f)\),

- has a computation which outputs a \(\beta\)-name for \(b = f(a)\) and

- every infinite computation of \(M\) on \(\bar{\sigma}\) outputs a \(\beta\)-name for \(b = f(a)\).

This definition is sensible insofar as it leads to closure under composition:

**Observation 4.70.** Let \(f : \subseteq A \rightarrow B\) be nondeterministically \((\alpha \rightarrow \beta)\)-computable and \(g : \subseteq B \rightarrow C\) be nondeterministically \((\beta \rightarrow \gamma)\)-computable. Then, \(g \circ f : \subseteq A \rightarrow C\) is nondeterministically \((\alpha \rightarrow \gamma)\)-computable.

While admittedly even less realistic than a classical \(\mathcal{NP}\)-machine, Type-2 nondeterminism turns out in Section 4.5.2 to benefit from particular structural elegance (in addition to closure under composition).

**Remark 4.71.** [20, Definition 14] defined nondeterministic computability deviating from Definition 4.69 in the third condition: there we required any infinite output of \(M\) on \(\bar{\sigma}\) to constitute a \(\beta\)-name for \(b = f(a)\). Since any infinite output requires infinite computation but not vice versa, this may seem to lead to a different notion. However, both do coincide: \(M\) may additionally guess and verify a function \(F : \mathbb{N} \rightarrow \mathbb{N}\) such that the \(n\)-th symbol is output after \(F(n)\) steps. If \(F\) has been guessed incorrectly (and in particular if, for the given input \(\bar{\sigma}\), no such \(F\) exists at all), then this can be detected within finite time and the computation then be aborted, thus complying with the (only seemingly stronger) Definition 4.69.
4.5. TYPE-2 NONDETERMINISM

4.5.1 Computational Power on Finite Inputs

A subtle point in Definition 4.69: the nondeterministic machine may ‘withdraw’ from a guess as long as it does so within finite time. This gives it the surprising power of ‘deciding’ all hyperarithmetical sets:

**Theorem 4.72.** For \( L \subseteq \mathbb{N} \), the ‘padded’ characteristic function \( \chi_L : \mathbb{N} \to \{0, 1\} \times \omega \) is nondeterministically computable if and only if \( L \in \Delta^1_1(2^{\mathbb{N}}) \).

In particular, the power of Type-2 nondeterminism goes strictly beyond the finite Effective Borel Hierarchy (Section 4.2.2); see also Corollary 4.75 below.

The following notion turns out to be both natural and useful in the proof of Theorem 4.72:

**Definition 4.73.** A set \( L \subseteq \mathbb{N} \) is nondeterministically semi-decidable if there exists a nondeterministic Turing machine \( M \) which, upon input of \( x \in \mathbb{N} \),
- has a computational path which outputs an infinite string in case \( x \in L \);
- in case \( x \notin L \) aborts, on each computational path, after finite time.

\( L \) is nondeterministically enumerable if a nondeterministic Turing machine \( M \) without input
- has a computational path which outputs a list \( (x_n) \) of integers with \( L = \{x_n : n \in \mathbb{N}\} \);
- every infinite computation of \( M \) prints a list \( (x_n) \) of integers with \( L = \{x_n : n \in \mathbb{N}\} \).

Nondeterministic enumerability thus amounts to nondeterministic computability of an En–name, cf. [Weih00, Definition 3.1.2.5]. Surprisingly, it turns out to be equivalent not to nondeterministic semi-decidability but to nondeterministic decidability:

**Proposition 4.74.** With respect to Type-2 nondeterminism and nondeterministic reducibility “\( \preceq^n \)”, it holds:

a) \( \text{En} \equiv^n \text{Cf} \), where the latter refers to the representation of the powerset of \( \mathbb{N} \) enumerating a set’s members in order [Weih00, Definition 3.1.2.6].

b) \( L \subseteq \mathbb{N} \) is nondeterministically decidable if and only if it has a computable Cf–name; equivalently: both \( L \) and its complement are nondeterministically semi-decidable.

c) \( L \subseteq \mathbb{N} \) is nondeterministically semi-decidable if and only if \( L \in \Sigma^1_1(2^{\mathbb{N}}) \).

**Proof.** a) “Cf \( \preceq \) En” holds even deterministically. For the converse we are given a list \( (x_n) \) of integers enumerating \( L \). Let the machine guess a function \( F : \mathbb{N} \to \mathbb{N} \) with \( x_n \geq m \forall n \geq F(m) \): Such obviously \( F \) exists; and an incorrect guess can be detected within finite time. Knowing \( F \), we can determine and sort all restrictions \( L \cap [1, m] \), \( m \in \mathbb{N} \).

b) Immediate.

c) Let \( L \in \Sigma^1_1 \). By Equation (3.8), a nondeterministic Type-2 machine \( M \) may, given \( x \), iteratively guess entries of \( b \) and check “\( \langle x, n; b|\leq_n \rangle \in R \)” for each \( n \in \mathbb{N} \): if this condition is violated, abort (within finite time); otherwise output a dummy symbol and continue with \( n + 1 \). This yields nondeterministic semi-decision of \( L \).

Conversely let \( L \) be semi-decided by \( M \). Then \( x \in \mathbb{N} \) belongs to \( L \) if and only if there exists a sequence \( (b_n) \) of guesses \( b_n \in \{0, 1\}^\omega \) such that \( M \), on input \( x \) and \( b \), “performs at last \( n \) steps”. The latter predicate on \( \langle x, n; b_1, \ldots, b_n \rangle \) being decidable, \( L \) is of the form \( \Sigma^1_1 \). \( \square \)

Claims b) and c) together yield Theorem 4.72.
Corollary 4.75. There is nondeterministically computable real \( c \in \mathbb{R} \) which does not belong to (any finite level of) the Arithmetical Hierarchy of real numbers \( \Delta_d(\mathbb{R}) \) from Section 4.2 (The number \( c \) will, however, belong to some transfinite \( \Delta_\omega(\mathbb{R}) \), recall Section 4.2.2.)

Proof. Take some hyperarithmetical but not finitely arithmetical \( L \subseteq \mathbb{N} \), that is, \( L \in \Delta^1_1 \setminus \Sigma^0_1 \); see e.g. 

\[ \text{[Roge87] \text{THEOREM } \S 16.1.XI} \text{ or } \text{[Odif89] \text{COROLLARY } IV.2.23} \]. Since \( L \) is nondeterministically decidable, it leads to a nondeterministically \( \rho_{b,2} \)–computable real \( c := \sum_{n \in L} 2^{-n} \in \mathbb{R} \). Were \( c\rho^{(d)} \)–computable for some \( d \in \mathbb{N} \), its (unique) binary expansion would be decidable relative to \( \emptyset^{(d)} \) (Theorem 4.9), i.e. belong to \( \Delta_{d+1} \): contradiction. \( \square \)

4.5.2 Computational Power on Infinite Inputs

Both Theorem 4.72 and Corollary 4.75 employed nondeterministic computation involving a finite (or no) input. Type-2 computation generally involves infinite strings for both input and output; e.g. when converting uniformly between different equivalent representations. This capability of a nondeterministic Type-2 machine is exploited in the next

Theorem 4.76. For each \( d = 0, 1, 2, \ldots \), the identity \( \mathbb{R} \ni x \mapsto x \) is nondeterministically \( (\rho^{(d+1)} \mapsto \rho^{(d)}) \)–computable. It is furthermore nondeterministically \( (\rho \mapsto \rho_{b,2}) \)–computable.

Proof. Consider first the case \( d = 0 \). Let \( x \in \mathbb{R} \) be given by a sequence \( (q_n) \subseteq \mathbb{Q} \) eventually converging to \( x \). Then there exists a fast convergent Cauchy subsequence \( (q_{n_k})_k \), that is, satisfying

\[
\forall k \geq \ell : \quad |q_{n_k} - q_n| \leq 2^{-\ell - 1} \quad (4.13)
\]

and thus forming a \( \rho \)–name for \( x \). To find this subsequence, guess, iteratively for each \( k \in \mathbb{N} \), some \( n_k > n_{k-1} \) and check whether it complies with Inequality (4.13) for the (finitely many) \( \ell \leq k \); if it does not, we may abort this computation in accordance with Definition 4.69.

For \( d = 1 \), let \( x = \lim_{n} x_n \) with \( x_n = \lim_{m} q_{n,m} \). Then apply the case \( d = 0 \) to convert for each \( n \) the \( \rho' \)–name \( (q_{n,m})_m \) of \( x_n \in \mathbb{R} \) into an according \( \rho \)–name, that is, a sequence \( p_{n,m} \) satisfying \( |x_n - p_{n,m}| \leq 2^{-m} \). Its diagonal \( (p_{n,n})_n \) then has \( |x - p_{n,n}| \leq |x - x_n| + 2^{-n} \rightarrow 0 \) and is thus a \( \rho' \)–name for \( x \). Higher levels \( d \) can be treated similarly by induction.

For \( (\rho \mapsto \rho_{b,2}) \)–computability, let \( x \in (0, 2) \) be given by a fast convergent sequence \( (q_n) \subseteq \mathbb{Q} \). We guess the leading digit \( b \in \{0, 1\} \) for \( x \)'s binary expansion \( b.*; \) in case \( b = 0 \), check whether \( x > 1 \)–a \( \rho \)–semi decidable property—and if so, abort; similarly in case \( b = 1 \), abort if it turns out that \( x < 1 \). Otherwise (that is, proceeding while simultaneously continuing the above semi-decision process via dove-tailing) replace \( x \) by \( 2(x - b) \) and repeat guessing the next bit. \( \square \)

It is instructive to observe how, in the case of non-uniform binary expansion (i.e., for dyadic \( x \), nondeterminism in the above \( (\rho \mapsto \rho_{b,2}) \)–computation generates, in accordance with the third requirement of Definition 4.69, both possible expansions.

Theorem 4.76 implies that the nondeterministic computability of real functions is largely independent of the representation under consideration—in striking contrast to the deterministic case (Observation 4.13), where the effectivity subtleties arising from different encodings had already confused Turing himself [Tur37].

Corollary 4.77. a) With respect to nondeterministic reduction \( \equiv^n_{\Sigma} \), it holds that \( \rho_{b,2} \equiv^n \rho \equiv^n \rho' \equiv^n \rho'_{<} \equiv^n \rho'' \equiv^n \ldots \).

b) The entire Real Arithmetical Hierarchy from Section 4.2 is nondeterministically computable.

Proof. a) follows from Theorem 4.76

b) Let \( x \in \Delta_{d+1} \) for some \( d \in \mathbb{N} \). Then \( x \in \mathbb{R} \) is \( \rho^{(d)} \)–computable, hence nondeterministically also \( \rho_{b,2} \)–computable by a). Alternatively or for \( d \geq \omega \) combine Theorem 4.72 with Theorem 4.9. \( \square \)
In particular, this kind of hypercomputation allows for nondeterministic \((\rho \rightarrow \rho)\)-evaluation of Heaviside’s function by appending to the \((\rho \rightarrow \rho_-)\)-computation in Example 4.23 a conversion from \(\rho_- \preceq^n \rho'\) back to \(\rho\).

Section 6.4 establishes many more real functions, both continuous and discontinuous, to be nondeterministically computable, too.
Chapter 5

BCSS Hypercomputation

Unlike the Turing machine, the BCSS machine over \((\mathbb{R}, +, -, \times, \div, <)\) suggests (at least) two ways of extending its power: by means of oracles or by enhancing the set of operational primitives. We first turn to the latter.

5.1 Machines with Additional Operations

One source of criticism of the BCSS model is its inability to compute the square root or simple transcendental functions like exponential or sine [Brat00]. Adding the first as a primitive trivially renders the square root function computable (compare Example 2.18d) but does not increase the machine’s power regarding decidability (as opposed to function computability):

**Observation 5.1.** Every problem (semi-)decidable by a BCSS machine \(M\) over \((\mathbb{R}, +, -, \times, \div, <, \sqrt{}\) is also (semi-)decidable by one without square root operation involving no additional real constants.

**Proof.** As in Section 2.3.2, unroll a computation of \(M\) into a ternary tree, now with each node \(u\) assigned a function \(f_u : \mathbb{R}^n \to \mathbb{R}\) composed from \(+, -, \times, \div\) and square roots; and with outgoing edges corresponding to the three possible signs of \(f_u(\vec{x})\). Recursively substituting each (sub)expression \(\sqrt{y_i}\) of \(f_u\) with a new variable \(y_i\) and introducing additional conditions \(y_i^2 = g_i \land y_i \geq 0\) reveals that \(\text{sgn} f_u(\vec{x}) = \sigma\) is equivalent to the solvability w.r.t. \(\vec{y}\) of a finite system \(G_u(\vec{x}, \vec{y})\) of (in)equalities. Hence termination of \(M\) on \(\vec{x}\) amounts to the existence of some leaf \(v\) such that \(G_v(\vec{x}, \vec{y})\) is satisfiable w.r.t. \(\vec{y}\). This in turn can be decided by Fact 2.13; even uniformly in \(u\), compare Fact 2.32.

Further, for a machine with square root, the third root function \(\mathbb{R} \ni x \mapsto \sqrt[3]{x}\) remains uncomputable (although its graph is of course easily decidable). This raises

**Question 5.2.** Is there a countable collection of operations \(o_a\) such that

- Every real language decidable by a BCSS machine with these operations is also decidable by a standard one.
- If \(f\) is a real function with decidable graph(\(f\)), then \(f\) is computable by a machine with these operations.

Including exponentiation \(x \mapsto e^x\) positively affects function computability and the decidability of graph(exp), recall Example 2.18.
5.1.1 ... and Nondeterminism

Recall that nondeterminism does not increase a standard BCSS machine’s power (Corollary 2.31b). This is desirable for its similarity to the discrete case as it ensures a real complexity class like $NP_R$ to be (if not equal to $P_R$) at least decidable. It remains true when the square root operation is included; but regarding exponentiation we are confronted with the following open

**Question 5.3.** *Is the first-order existential theory $(\mathbb{R}, +, -, \times, \div, <, \exp)$ BCSS-decidable?*

On the one hand, quantifier elimination provably fails there [Drie84]. On the other hand, [Maci91] has observed that nondeterministic machines with exponentiation might admit of a simulation by a deterministic one by means of strengthening the Lindemann–Weierstraß Theorem (Fact 2.5c) known as Schanuel’s

**Conjecture 5.4 (Schanuel).** Let $x_1, \ldots, x_n \in \mathbb{R}$ be linearly independent over $\mathbb{Q}$. Then the field extension $\mathbb{Q}(x_1, \ldots, x_n, e^{x_1}, \ldots, e^{x_n})$ has transcendence degree at least $n$ over $\mathbb{Q}$.

For a full account of Question 5.3 refer to [MaWi96, Mark96]. However, regarding the sine function (with its close relation to complex exponentiation) as an additional operation, this has been observed in [Meer93] to not allow simulation of nondeterministic machine by deterministic ones:

**Example 5.5.**

- a) The set $\mathbb{Z}$ of integers is decidable in constant time (compare Example 2.15c) to a BCSS machine over $(\mathbb{R}; +, -, \times, \div, <, \sin)$.

- b) To such an enhanced machine, the set $\mathbb{Q}$ of rational numbers remains undecidable (compare Example 2.18a).

- c) $\mathbb{Q}$ becomes decidable if the above machine may in addition make (and verify) a real guess.

**Proof.**

a) For $x \in \mathbb{R}$, $x \in \mathbb{Z} \Leftrightarrow \sin(x \cdot \pi) = 0$.

b) By (the first proof of) Lemma 2.19, a putative algorithm unrolled into a tree contains a non-empty open leaf set $X_v$, but this open set will contain both rational and irrational numbers, which are thus not separated by the computation: contradiction. □

c) For $x \in \mathbb{R}$, $x \in \mathbb{Q} \Leftrightarrow \exists y \in \mathbb{R} : \sin(\pi \cdot x \cdot y) = 0 \land y \in \mathbb{Z}$.

5.2 Machines with Oracles

An immediate extension of Definition 2.14 equips the BCSS machine with oracle access [BCSS98 Definition 21.3.5]:

**Definition 5.6.** A BCSS oracle machine using an oracle set $O \subseteq \mathbb{R}^*$ is a BCSS machine with an additional type of node called an oracle node. By entering such a node, the machine asks the oracle whether a previously computed element $\vec{y} \in \mathbb{R}^*$ belongs to $O$ and can proceed according to the answer (yes or no).

A real function $f : \subseteq \mathbb{R}^* \rightarrow \mathbb{R}^*$ is BCSS–computable relative to $O$ iff some BCSS machine with oracle $O$ can compute $f$. A real language $L \subseteq \mathbb{R}^*$ is semi-decidable relative to $O$ if it coincides with the domain of a function BCSS–computable relative to $O$; $L$ is r.e. relative to $O$ iff it coincides with the range of a real function BCSS–computable relative to $O$. We call $L$ BCSS reducible to $O$ (and write “$L \leq O$”) if $L$ is decidable relative to $O$ (compare Definition 3.2).

The major technique for many uncomputability proofs in Section 2.3.2 would unroll a putative computation of $M$ into a ternary algebraic computation tree corresponding to the three possible outcomes of a sign test “$g(\vec{x}) : 0$”. Adding oracle power induces further branching possibilities of $M^O$ based on the outcome of a query “$g(\vec{x}) \in O$”. Here, $g$ denotes a multi-variate quolynomial (vector) over the field extension $\mathbb{Q}(c_1, \ldots, c_D)$ generated by the constants of $M^2$; and each tree node $u$ has such a $g = g_u$ associated with it.
5.2.1 Relativized BCSS Recursion Theory

Most results in discrete computability relativize straightforwardly, that is they and their proofs immediately carry over to the setting of computation with a (fixed, common, but arbitrary) oracle. This is not necessarily the case in BCSS theory with its heavy use of quantifier elimination. For instance, one direction of Fact (2.17) fails for BCSS oracle computation:

Example 5.7. Let

$$\emptyset := \{e^{\sqrt{2}}, e^{\sqrt{3}}, e^{\sqrt{5}}, e^{\sqrt{7}}, \ldots, e^{\sqrt{p_n}}, \ldots\} \quad \text{and} \quad \mathbb{L} := \{e^{2\sqrt{2}}, e^{2\sqrt{3}}, e^{2\sqrt{5}}, e^{2\sqrt{7}}, \ldots, e^{2\sqrt{p_n}}, \ldots\}$$

where \(p_n \in \mathbb{N}\) denotes the \(n\)-th prime number. Then \(\mathbb{L}\) is r.e. relative to \(\emptyset\) but not semi-decidable relative to \(\emptyset\).

Intuitively speaking, the enumerability of \(\mathbb{L}\) relative to \(\emptyset\) proceeds by computing squares, whereas semi-decision requires taking square roots (recall Example 2.18d). Our proof applies the following

Lemma 5.8. Let \(\{x_1, x_2, \ldots, x_n, \ldots\} \subseteq E\) be algebraically independent over some field \(F\) and \(y_1, y_2, \ldots, y_m \in E\). Then there exists some \(N \in \mathbb{N}\) such that \(\{x_N, x_{N+1}, \ldots\}\) is algebraically independent over \(F(y_1, y_2, \ldots, y_m, x_1, \ldots, x_{N-1})\).

We refrain from putting this (far more combinatorial than algebraic) claim into the general context of infinite matroids [Rado49] but give a (semi-)tailored

Proof. Let

$$f : \mathbb{N} \rightarrow \mathbb{N}, \quad f(n) := \text{tr.\,deg}_F(y_1, \ldots, y_m)(x_1, \ldots, x_n)$$

Then, according to Fact (2.5d),

$$n \geq f(n) = \text{tr.\,deg}_F(y_1, \ldots, y_m, x_1, \ldots, x_n) - \text{tr.\,deg}_F(y_1, \ldots, y_m) \geq n - m$$

because \(\text{tr.\,deg}_F(y_1, \ldots, x_n) = n\) by hypothesis. On the other hand, \(f(n) \leq f(n + 1) \leq f(n) + 1\) requires that \(f(n + 1) = f(n)\) holds for at most \(m\) values of \(n\). Let \(N\) denote the largest of them. Then \(f(N + n) = f(N) + n\) for all \(n \in \mathbb{N}\), that is

$$\text{tr.\,deg}_F(y_1, \ldots, y_m, x_1, \ldots, x_{N-1})(x_N, x_{N+1}, \ldots, x_{N+n}) = f(N + n) - f(N) = n \quad \text{again by Fact (2.5d).} \quad \square$$

Proof (Example 5.7). Let \(f : \mathbb{R}^* \rightarrow \mathbb{R}^*\) be the function with \(f(x) := x^2\) for \(x \in \emptyset\) and \(f(\bar{x}) := \exp(\sqrt{2})\) otherwise: It is computable relative to \(\emptyset\) (by a BCSS-machine with one real constant), total, and has range \(\mathbb{L}\).

Now suppose that some BCSS oracle machine \(M^\emptyset\) with constants \(c_1, \ldots, c_D\) semi-decides \(\mathbb{L}\). As indicated above, this leads to a 6-ary algebraic computation tree whose leaf sets \(X_v\) partition \(\mathbb{L}\), each \(X_v\) a finite intersection of sets \(g_u^{-1}(\{0\})\), \(g_u^{-1}([0, \infty))\), \(g_u^{-1}((0, \infty])\), \(g_u^{-1}([0, \infty))\), and \(g_u^{-1}([\infty, \infty))\); where \(g_u \in F(X)\) denotes some quolynomial with coefficients from \(F := \mathbb{Q}(c_1, \ldots, c_D)\).

Since \(\emptyset\) is algebraically independent over \(\mathbb{Q}\) (compare Example 2.29), according to Lemma 5.8 below there exists some \(N \in \mathbb{N}\) such that \(\{x_N, x_{N+1}, \ldots\}\) is algebraically independent over \(E := F(x_1, \ldots, x_{N-1})\) where \(x_n := \exp(\sqrt{p_n})\). Consider the leaf \(v_0\) which input \(x_N^2 \in \mathbb{L}\) ends up in via nodes \(u_1, \ldots, u_m = v\). We claim that

a) \(g_{u_m}(x_N^2) \neq 0\) (because if \(x_N^2\) were algebraic over \(E\), then so would \(x_N\) be)

b) and \(g_{u_m}(x_N^2) \notin \emptyset\).

Indeed \(g(x_N^2) = x_n\) with \(g \in F[X]\) implies that \(\{x_N, x_n\}\) are algebraically dependent over \(E\), leading to three possible cases:

\(\mathbf{n < N}\) contradicts the above choice of \(N\) such that \(x_N\) is transcendental over \(E \supseteq F(x_n)\);
\[ n > N \] also contradicts the choice of \( N \) such that \( \{ x_N, x_n \} \) is algebraically independent even over \( E \supseteq F \);

\[ n = N \] implies that \( x_N \) is a root of \( g_u(X^2) - X \in F[X] \setminus \{ 0 \} \) contradicting, again, the fact that \( x_N \) is transcendental over \( F \).

Now \( g_u \) being a (w.l.o.g. non-constant) quolynomial, its pre-image \( g_u^{-1}(y) \) is finite for each \( y \); hence \( \bigcup_{n=1}^{m} g_u^{-1}(y) \cup L \subseteq R \) is at most countable and the complement thus dense in \( R \). This means that there exists some \( z \in R \setminus L \) with \( \text{sgn} (g_u(z)) = \text{sgn} (g_u(x_N^2)) \) —an open condition according to Property a) above—such that \( g_u(z) \not\in O \) for all \( i = 1, \ldots, m \) i.e. also satisfying Property b). This \( z \) will thus follow the same path \( u_1, \ldots, u_m \) through the above computation tree to end up in leaf \( v \) like input \( x_N^2 \in L \); in particular, the putative machine \( M^O \) will also terminate on input \( z \) although \( z \not\in L \); contradiction.

The other direction of (the BCSS analogue of) Fact 2.17 does hold, though:

**Observation 5.9.** If \( L \neq \{ \} \) is semi-decidable relative to \( O \), then it is also r.e. relative to \( O \).

The proof is identical to that described at the beginning of Section 2.3.3

Regarding real constants, Scholium 2.33 and Example 2.34 have pointed out that it makes a difference (of at most one) whether \( L \) is the range of a partial or of a total real function. Proceeding from the former to the latter may, in the case of oracle computation, even require an unbounded number of additional real constants:

**Example 5.10.** For \( n \in \mathbb{N} \) let
\[
\vec{z}_n := (e^{\sqrt{2}}, e^{\sqrt{3}}, e^{\sqrt{5}}, e^{\sqrt{7}}, e^{\sqrt{11}}, \ldots, e^{\sqrt{p_n}}) \in \mathbb{R}^n
\]
where \( p_n \in \mathbb{N} \) denotes the \( n \)-th prime number. Then \( L_n := \{ \vec{z}_n \} \) is trivially decidable relative to \( O := \{ \vec{z}_n : n \in \mathbb{N} \} \) by a constant-free BCSS-machine; but any BCSS-machine \( M_n \) computing, relative to \( O \), a total real function \( f_n \) with image \( L_n \) must contain at least \( n \) real constants.

**Proof.** Since \( L_n = \{ \vec{z}_n \} \) is a singleton, the value \( f_n(0) \) must coincide with tuple \( \vec{z}_n \) of \( n \) algebraically independent real numbers, compare Example 2.29. According to the above argument of unrolling the computation of \( M_n^O \), \( f_n \) on the other hand is a vector of quynomials over \( F = \mathbb{Q}(c_1, \ldots, c_D) \) where \( c_1, \ldots, c_D \) denote the real constants of \( M_n \). In particular, the components of \( f_n(0) \) belong to \( F \) and thus have transcendence degree at most \( D \); whereas those of \( \vec{z} \) have transcendence degree \( n \).

### 5.2.2 Relative Function Computability

Undecidable oracles of course strictly increase a BCSS machine’s power: Trivially, the characteristic function \( X_H : \mathbb{R}^* \to \{ 0, 1 \} \) of \( H \) is uncomputable but computable relative to \( H \). In contrast, oracles do not help in computing many practical functions; specifically, we have the following strengthening of Example 2.18d):

**Theorem 5.11.** Square root and exponentiation are both BCSS uncomputable relative to any oracle \( O \subseteq \mathbb{R}^* \).

We thus cannot avoid he considerations of Section 5.1.

**Proof.** Suppose BCSS machine \( M \) with constants \( c_1, \ldots, c_D \in \mathbb{R} \) and oracle \( O \) computes some real function \( f \). Unrolling the computation leads to an at most countable tree whose leaves \( v \) induce a partition of \( \text{dom}(f) = \bigcup \mathcal{X}_v \), and, as in Lemma 2.26b), each restriction \( f|_{\mathcal{X}_v} \) coincides with a quolynomial over \( F := \mathbb{Q}(c_1, \ldots, c_D) \). In particular, for each \( \vec{x} \in \text{dom}(f) \), \( f(\vec{x}) \) belongs to the field \( \mathbb{Q}(\vec{x}; c_1, \ldots, c_D) \).
5.3. *POST’S PROBLEM OVER THE REALS*

For the case \( f(x) = \sqrt{x} \) and letting \( x \) run through the sequence \( (n_1, n_2, \ldots) \) of all squarefree integers, this implies \( \{\sqrt{n_1}, \sqrt{n_2}, \ldots\} \subseteq \mathbb{Q}(c_1, \ldots, c_D) \); hence

\[
1 = \left[ \mathbb{Q}(c_1, \ldots, c_D; \sqrt{n_1}, \sqrt{n_2}, \ldots) : \mathbb{Q}(c_1, \ldots, c_D) \right] = \frac{\left[ \mathbb{Q}(c_1, \ldots, c_D; \sqrt{m_1}, \sqrt{m_2}, \ldots) : \mathbb{Q} \right]}{\left[ \mathbb{Q}(c_1, \ldots, c_D) : \mathbb{Q} \right]} \geq \frac{\left[ \mathbb{Q}(\sqrt{m_1}, \sqrt{m_2}, \ldots) : \mathbb{Q} \right]}{\left[ \mathbb{Q}(c_1, \ldots, c_D) : \mathbb{Q} \right]}
\]

by Fact 2.2a); but, according to Lemma 2.3b), the right hand side equals infinity. The argument for the case \( f(x) = e^x \) similarly yields a contradiction to Fact 2.5a).

In view of Corollary 2.20 we ask

**Question 5.12.** Is there a relatively computable real pairing function \( \mathbb{R}^2 \rightarrow \mathbb{R} \)?

### 5.3 Post’s Problem over the Reals

Let us recall Question 3.13 on (semi-decidable yet) undecidable problems that are *strictly* easier than the Halting problem: Its answer turned out to be positive but to rely basically on diagonalization and hence yield merely the existence of such an intermediate problem \( P \subseteq \mathbb{N} \) described in terms of a Turing machine enumerating \( P \). On the other hand, within the realm of real computation, we were able to establish an explicit (and far less artificial) problem that is BCSS (semi-decidable and) undecidable yet *strictly* easier than \( \mathbb{H} \): the set \( \mathbb{Q} \) of rational numbers. This parallels results in complexity where *non*-complete problems in \( \mathcal{N}\mathcal{P} \setminus \mathcal{P} \) (under the hypothesis that the latter is non-empty of course) could be ‘constructed’ only by diagonalization in the discrete case (recall Section 3.1.4) but explicitly over the reals [Buer00 Section 5.5].

**Theorem 5.13.**

a) For the set \( \mathbb{A} \) of algebraic reals, it holds \( \mathbb{Q} < \mathbb{A} < \mathbb{H} \).

b) But the set \( \mathbb{T} = \mathbb{R} \setminus \mathbb{A} \) of transcendental reals is not BCSS semi-decidable relative to \( \mathbb{Q} \).

In particular \( \mathbb{A} \not< \mathbb{Q} \), hence \( \mathbb{H} < \mathbb{Q} \): both \( \mathbb{A} \) and \( \mathbb{H} \) remain undecidable even when oracle access to \( \mathbb{Q} \) is granted.

**Proof.** Regarding a), use the \( \mathbb{A} \)-oracle to test whether input \( x \in \mathbb{R} \) is algebraic: if not, then certainly \( x \not\in \mathbb{Q} \), so reject; otherwise \( (x \in \mathbb{A}) \) invoke Lemma 2.23 to test whether \( \deg(x) \leq 1 \). This establishes \( \mathbb{Q} < \mathbb{A} \); \( \mathbb{A} < \mathbb{H} \) follows from Example 2.21 The bottom claim follows from b) since Fact 2.17a) relativizes.

Regarding the major Claim b), our proof below makes use of three lemmas. The first one ensures that algebraic reals remain dense in \( \mathbb{R} \) even when restricted to arbitrary high degree:

**Lemma 5.14.** Let \( x \in \mathbb{R} \), \( \varepsilon > 0 \), and \( N \in \mathbb{N} \).

There then exists an algebraic real \( a \) of degree \( N \) with \( |x - a| < \varepsilon \).

**Proof.** Take some arbitrary algebraic real \( b \) of degree \( N \), such as \( b := 2^{1/N} \) (Section 2.2.1). Since \( \mathbb{Q} \) is dense in \( \mathbb{R} \ni y := x - b \), there exists some rational number \( r \) with \( |r - y| < \varepsilon \). Then \( a := r + b \) has the desired property.

The undecidability of \( \mathbb{A} \) without further oracle assistance was already established in Example 2.18a). Its proof relied on every semi-decidable set consisting of at most countably many connected components. This argument fails for oracle computation: \( \mathbb{R} \setminus \mathbb{Q} \) has uncountably many components (namely singletons) but is decidable relative to \( \mathbb{Q} \). Similarly, a putative algorithm might try distinguishing algebraic from transcendental reals relative to \( \mathbb{Q} \) by mapping a given \( x \) through some rational function \( f \in \mathbb{R}(X) \), then querying the oracle whether the value \( f(x) \) is rational or not, and proceeding adaptively according to the answer. The following observation basically says that in any sensible approach of this kind, for transcendental \( x \), \( f(x) \) will be irrational rather than rational.
Lemma 5.15. Let \( f : \text{dom}(f) \subseteq \mathbb{R} \rightarrow \mathbb{R} \) be analytic and non-constant, \( T \subseteq \text{dom}(f) \) uncountable. Then \( f \) maps some \( x \in T \) to a transcendental value, that is, \( f(x) \not\in \mathbb{A} \).

Proof. Consider an arbitrary \( y \in \mathbb{A} \); by uniqueness of analytic functions \([\text{Rud66, Theorem 10.18}]\), \( f \) can map at most countably many different \( x \in \text{dom}(f) \) to that single value \( y \). Hence, if \( f(x) \in \mathbb{A} \) for all \( x \in T \), \( f^{-1}(\mathbb{A}) = \bigcup_{y \in \mathbb{A}} f^{-1}\{y\} \) is a countable union of countable sets and thus also countable—contradicting the prerequisite that \( T \subseteq f^{-1}(\mathbb{A}) \) is uncountable. \( \square \)

Thus, there remains the case of an algorithm trying to map algebraic \( x \) to rationals \( f(x) \) and transcendental \( x \) to irrational \( f(x) \). The final ingredient formalizes the intuition that this approach cannot distinguish transcendentals from algebraic numbers of sufficiently high degree:

Lemma 5.16. Let \( f \in \mathbb{R}(X) \) be non-constant such that \( f = p/q \) with polynomials \( p, q \) of \( \deg(p) < n \), \( \deg(q) < m \). Let \( a_1, \ldots, a_{n+m} \in \text{dom}(f) \) be distinct real algebraic numbers with \( f(a_1), \ldots, f(a_{n+m}) \in \mathbb{Q} \).

a) There are co-prime polynomials \( \tilde{p}, \tilde{q} \) of \( \deg(\tilde{p}) < n \), \( \deg(\tilde{q}) < m \) with coefficients in the algebraic field extension \( \mathbb{Q}(a_1, \ldots, a_{n+m}) \) such that, for all \( x \in \text{dom}(f) = \{ x : q(x) \neq 0 \} \subseteq \mathbb{R} \), it holds \( f(x) = f(x) := \tilde{p}(x)/\tilde{q}(x) \).

b) Let \( d := \max, \deg(a_i) \). Then \( f(x) \not\in \mathbb{Q} \) for all transcendental \( x \in \text{dom}(f) \) as well as for all \( x \in \mathbb{A} \) of \( \deg(x) > D := d^{n+m} \cdot \max\{n-1, m-1\} \).

Notice that \( p \) and \( q \) themselves in general do not satisfy claim a); e.g. \( p = \pi \cdot \tilde{p} \) and \( q = \pi \cdot \tilde{q} \).

Proof. a) Without loss of generality, take \( p \) and \( q \) to be co-prime. Let \( y_i := f(a_i) \). The idea is to solve the rational interpolation problem for \( (a_i, y_i) \). Already knowing that it has a solution (namely \( p, q \)) avoids many of the difficulties discussed in \([\text{MaDr62}]\).

More precisely, observe that the coefficients \( p_0, \ldots, p_{n-1}, q_0, \ldots, q_{m-1} \in \mathbb{R} \) of \( p \) and \( q \) satisfy the homogeneous \( (n+m) \times (n+m) \)-size system of linear equations

\[
\begin{pmatrix}
1 & a_1 & a_1^2 & \cdots & a_1^{n-1} & -y_1 & -y_1 a_1 & \cdots & -y_1 a_1^{m-1} \\
1 & a_2 & a_2^2 & \cdots & a_2^{n-1} & -y_2 & -y_2 a_2 & \cdots & -y_2 a_2^{m-1} \\
1 & a_3 & a_3^2 & \cdots & a_3^{n-1} & -y_3 & -y_3 a_3 & \cdots & -y_3 a_3^{m-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
1 & a_{n+m} & a_{n+m}^2 & \cdots & a_{n+m}^{n-1} & -y_{n+m} & -y_{n+m} a_{n+m} & \cdots & -y_{n+m} a_{n+m}^{m-1}
\end{pmatrix}
\begin{pmatrix}
p_0 \\
p_1 \\
p_2 \\
p_{n-1} \\
q_0 \\
q_1 \\
q_2 \\
q_{m-1}
\end{pmatrix}
= 0.
\]

In particular, this system has \((p_0, \ldots, q_{m-1}) \in \mathbb{R}^{n+m}\) as a non-zero solution. The coefficients of the matrix live in \( \mathbb{Q}(a_1, \ldots, a_{n+m}) \). Therefore, Gaussian Elimination yields a (possibly different) non-zero solution \((\tilde{p}_0, \ldots, \tilde{q}_{m-1})\), also with entries in \( \mathbb{Q}(a_1, \ldots, a_{n+m}) \). Now apply the Euclidean Algorithm to the polynomials \( \tilde{p}, \tilde{q} \) obtained in this way and calculate their greatest common divisor \( \tilde{h} \) which, again, has coefficients in \( \mathbb{Q}(a_1, \ldots, a_{n+m}) \).

Thus, \( \tilde{p} := \tilde{p}/\tilde{h} \) and \( \tilde{q} := \tilde{q}/\tilde{h} \) are co-prime polynomials over \( \mathbb{Q}(a_1, \ldots, a_{n+m}) \) of \( \deg(\tilde{p}) < n \) and \( \deg(\tilde{q}) < m \) such that \( \tilde{p} \cdot \tilde{q} \) coincides with \( p \cdot q \) on arguments \( a_1, \ldots, a_{n+m} \). This implies that the latter polynomials of degree less than \( n + m \) are identical: \( \tilde{p} \cdot \tilde{q} = p \cdot q \).

It follows that \( q \) divides both sides; and coprimality of \( (p, q) \) in the factorial ring \( \mathbb{R}[X] \) requires that \( q \) divides \( \tilde{q} \). Similarly, \( \tilde{q} \) divides \( q \), yielding \( \tilde{q} = \lambda q \) for some \( \lambda \in \mathbb{R} \). Analogously, \( \tilde{p} = \lambda p \) for the same \( \lambda \).

b) Consider \( x \in \mathbb{R} \) with \( y := f(x) \in \mathbb{Q} \) and suppose \( x \) is algebraic of \( \deg(x) > d^{n+m} \cdot \max\{n-1, m-1\} \) or transcendental. Since \( f \) is non-constant, the polynomial \( p - y \cdot q \) is not identically zero. Being, by virtue of a), a zero of this polynomial with coefficients from \( \mathbb{Q}(a_1, \ldots, a_n) \), \( x \) lies in an algebraic extension of the latter field, hence ruling out the possibility that it is
transcendental. More precisely, the degree of $x$ over $\mathbb{Q}(a_1, \ldots, a_n)$ is bounded by $\deg(p-y \cdot q)$; and $\deg(x)$, its degree over $\mathbb{Q}$, is at most $\deg(p-y \cdot q) \cdot \deg(a_1) \cdots \deg(a_{n+m}) \leq \max\{n-1, m-1\} \cdot n^m$ by Fact 2.2a)—contradiction.

**Proof (Theorem 5.13b).** Consider the 6-ary tree obtained by unrolling a putative BCSS algorithm semi-deciding $\mathbb{R} \setminus \mathbb{A}$ relative to $\mathbb{Q}$. Since this tree has at most countably many leaves $v$ but the set $\mathbb{R} \setminus \mathbb{A}$ of terminating inputs $x$ is uncountable, some $X_v$ must be uncountable as well. Consider the path $u_1, \ldots, u_m = v$ of nodes to that leaf: Each such $u$ is associated some quolynomial $g_u \in \mathbb{R}(X)$ with $X_u \subseteq \text{dom}(g_u)$ and gives rise to a branch based on $\text{sgn}(g_u(x))$ and/or on oracle query “$g_u(x) \in \mathbb{Q}$?.” W.l.o.g. no branch tests “$0 = g_u(x)$” or (positively answered) queries “$g_u(x) \in \mathbb{Q}$” occur on $u_1, \ldots, u_m$; for, otherwise, the uncountable set $X_v$ of transcendentals $x$ passing through this branch would require $g_u$ to be constant (Lemma 5.15) and node $u$ be thus dispensable.

Now take some $t \in X_v \subseteq \mathbb{R} \setminus \mathbb{A}$. Due to the continuity of rational functions, there exists $\varepsilon > 0$ such that $\text{sgn}(g_u(t)) = \text{sgn}(g_u(x)) \neq 0$ for all $i = 1, \ldots, m$ and all $x \in \mathbb{R}$ satisfying $|x-t| < \varepsilon$.

In particular, $g_u(a)$ and $g_u(t)$ have the same sign for algebraic numbers $a$ of unbounded degree according to Lemma 5.14. From the presumption, we conclude that none of them completes the (terminating) computational path to leaf $v$: they must branch off somewhere, that is, satisfy $g_u(a) \in \mathbb{Q}$ for some $i = 1, \ldots, m$. However by Lemma 5.16), each single $g_u$ can sort out only algebraics of degree up to some finite $D = D(v)$—contradiction.

### 5.3. POST’S PROBLEM OVER THE REALS

A further achievement of the works of Friedberg and Muchnik was the existence of incomparable semi-decidable degrees below the Halting problem. We now extend Theorem 5.13 to establish the same for the real case—again explicitly.

Consider the following type of algebraic extensions:

**Definition 5.17.** For fields $\mathbb{Q} \subseteq F \subseteq \mathbb{R}$ and $0 < r \in \mathbb{Q}$, let

$$F(\sqrt[r]{\mathbb{Q}}) := F\{\{r^\frac{n}{n} : n \in \mathbb{N}\}\}$$

where the fractional powers are understood as positive real numbers.

$\mathbb{Q}(\sqrt{2})$ for instance results from $\mathbb{Q}$ by field adjunction of all $n$-th roots of 2, $n \in \mathbb{N}$. The ancient proof of $\sqrt{2}$'s irrationality immediately generalizes to reveal that $|\mathbb{Q}(\sqrt{2} : \mathbb{Q})|$ is indeed infinite.

By Lemma 5.18a) below this extends to show, for example, $|\mathbb{Q}(\sqrt{2}, \sqrt{3} : \mathbb{Q}(\sqrt{2}))|$. In combination with Lemma 5.14 it generalizes Lemma 5.18a).

**Lemma 5.18.** a) For distinct prime numbers $p_1, p_2, \ldots, p_d, p_{d+1}$ and $n \in \mathbb{N}$, it holds

$$|\mathbb{Q}(\sqrt[p_1]{\mathbb{Q}}, \sqrt[p_2]{\mathbb{Q}}, \ldots, \sqrt[p_d]{\mathbb{Q}}, \sqrt[p_{d+1}]{\mathbb{Q}}) : \mathbb{Q}(\sqrt[p_1]{\mathbb{Q}}, \sqrt[p_2]{\mathbb{Q}}, \ldots, \sqrt[p_d]{\mathbb{Q}})| = \infty.$$  

b) For any $n \in \mathbb{N}$, $\epsilon > 0$, and $x \in \mathbb{R}$, there exists $y \in \mathbb{Q}(\sqrt{2})$ of degree at least $n$ over $\mathbb{Q}(\sqrt{3})$ such that $|x-y| < \epsilon$.

**Proof.** a) Combine Lemma 2.3a+c) with Fact 2.2a).

b) By a), $b := 2^{1/n}$ has degree $n$ over $\mathbb{Q}(\sqrt{3})$; and so has $y := b + r$ for any $r \in \mathbb{Q}$. $\mathbb{Q}$ being dense, take $r$ close to $x-b$.

**Theorem 5.19.** Both sets $\mathbb{Q}(\sqrt{2})$ and $\mathbb{Q}(\sqrt{3})$ are BCSS semi-decidable—yet both are incomparable with respect to BCSS reducibility: It holds $\mathbb{Q}(\sqrt{2}) \ntriangleleft \mathbb{Q}(\sqrt{3})$ and $\mathbb{Q}(\sqrt{3}) \ntriangleleft \mathbb{Q}(\sqrt{2})$.

The proof is based on the following immediate generalization of Lemma 5.16

**Lemma 5.20.** Let $f \in \mathbb{R}(X), f = \frac{p}{q}$ with polynomials $p, q$ of degree less than $n$ and $m$, respectively. Let $a_1, \ldots, a_{n+m} \in \mathbb{Q}(\sqrt{2}) \cap \text{dom}(f)$ be distinct with $f(a_i) \in \mathbb{Q}(\sqrt{3})$. 

a) There are co-prime polynomials \( \bar{p}, \bar{q} \) of \( \deg(\bar{p}) < n, \deg(\bar{q}) < m \) with coefficients in the algebraic field extension \( \mathbb{Q}(\sqrt{3}, a_1, \ldots, a_{n+m}) \) such that, for all \( x \in \text{dom}(f) = \{ x : q(x) \neq 0 \} \subseteq \mathbb{R} \), it holds \( f(x) = \bar{f}(x) := \bar{p}(x)/\bar{q}(x) \).

b) Let \( d := \max \deg_{\mathbb{Q}(\sqrt{3})}(a_i) \). Then \( f(x) \notin \mathbb{Q}(\sqrt{3}) \) for all transcendental \( x \in \text{dom}(f) \) as well as for all \( x \in \mathbb{Q}(\sqrt{2}) \) of \( \deg_{\mathbb{Q}(\sqrt{3})}(x) > D := d^{n+m} \cdot \max\{ n-1, m-1 \} \).

Proof (Theorem 5.19). For semi-decidability observe that, by virtue of Lemma 2.3(a) and Fact 2.2(c),
\[
\mathbb{Q}(\sqrt{2}) = \bigcup_{n \in \mathbb{N}} \mathbb{Q}(\sqrt{2}) = \{ x \in \mathbb{Q} \mid \exists n \in \mathbb{N} \exists a_0, \ldots, a_{n-1} \in \mathbb{Q} \exists y \in \mathbb{R} : x = a_0 + ya_1 + \ldots + y^{n-1}a_{n-1} \land y^n = 2 \} .
\]

Now existential quantification with respect to \( y \) can be eliminated from \( \Phi \) (Fact 2.13); hence membership of \( x \) in \( \mathbb{Q}(\sqrt{2}) \) can be semi-decided by searching for \( n \in \mathbb{N} \) and \( a_0, \ldots, a_{n-1} \in \mathbb{Q} \).

Consider a putative machine semi-deciding \( \mathbb{R} \setminus \mathbb{Q}(\sqrt{3}) \) by means of an \( \mathbb{Q}(\sqrt{3}) \)-oracle. Follow the proof of Theorem 5.13 and apply Lemma 5.15 to obtain in just the same way a leaf \( v \) together with the related path set \( X_v \subseteq \mathbb{R} \setminus \mathbb{Q}(\sqrt{2}) \). Since \( X_v \) is uncountable, it contains a transcendental \( t \) and, in each neighborhood of \( t \) by virtue of Lemma 5.18(a), elements of \( \mathbb{Q}(\sqrt{2}) \) of arbitrarily high degree over the field \( \mathbb{Q}(\sqrt{3}) \). Thus, applying Lemma 5.20, there exist elements in \( \mathbb{Q}(\sqrt{2}) \) that are branched along \( v \), contradicting the assumption that the machine semi-decides \( \mathbb{R} \setminus \mathbb{Q}(\sqrt{2}) \).

The semi-decidability of \( \mathbb{R} \setminus \mathbb{Q}(\sqrt{3}) \) relative to \( \mathbb{Q}(\sqrt{2}) \) is ruled out similarly. \( \square \)

The numbers 2 and 3 in the above proof can obviously be replaced by any two distinct primes; that is, the sets \( \mathbb{Q}(\sqrt{p}) \) and \( \mathbb{Q}(\sqrt{q}) \) are incomparable for any two \( p, q \in \mathbb{P} = \{ 2, 3, 5, 7, 11, 13, 17, \ldots \} \). In particular, we have explicitly an infinite number of incomparable degrees. Moreover, the argument immediately extends to reveal that, for \( P, Q \subseteq \mathbb{P} \), \( Q(\{ \sqrt{p} : p \in P \}) < Q(\{ \sqrt{q} : q \in Q \}) \) iff \( P \) is contained in \( Q \). Since the collection of subsets with inclusion is the prototype of a poset, we have thus arrived at the following

**Scholium 5.21.** Every countable poset can be embedded in the semi-decidable BCSS-degrees.

The latter parallels classical results in discrete recursion theory; see for instance Exercise VIII.2.2(b) and Exercise VIII.4.10).

### 5.3.2 The Case of Linear BCSS Machines

We have so far considered the full BCSS model over the reals, that is the structure \( (\mathbb{R}, +, -, \times, \div, <) \). Over the last decade, its linearly restricted version \( (\mathbb{R}, +, -, 0, 1, <) \) has attracted increasing interest due to its relation with the classical (i.e., discrete) \( \mathcal{P} = \mathcal{NP} \) question. Only additions, subtractions and comparisons are allowed in this model but no multiplication \( \times \) or division \( \div \) or constants other than 0 and 1. Thus, all computed intermediate results on inputs \( x \in \mathbb{R} \) have the form \( ax + b \) for some \( a, b \in \mathbb{Z} \). Analogously to the full model, the Halting Problem for linear machines is undecidable to a linear machine; and Post’s problem makes sense in the linear version as well.

Section 2 of [26] gives an explicit solution to it by establishing

**Theorem 5.22.** \( \mathbb{Q} \) is semi-decidable by a linear machine; yet undecidable even by one equipped with the set \( Q : Q := \{ q^2 : q \in \mathbb{Q} \} \) of quadratic rationals as oracle.

Instead of repeating its proof, we report on a very recent and much stronger solution due to C. Gassner:

**Definition 5.23.** For \( d \in \mathbb{N} \) let
\[
L_d := \{(x_1, \ldots, x_d) \mid \exists q_0, \ldots, q_{d-1} \in \mathbb{Q} : q_0 + \sum_{i=1}^{d-1} q_ix_i = x_d \} .
\]
These sets thus capture tuples of reals whose last component is affinely independent (over $\mathbb{Q}$) from the others. According to [Gass88] they form a strict hierarchy with respect to linear oracle machines:

**Theorem 5.24.** For each $d \in \mathbb{N}$, $\mathbb{L}_d$ is semi-decidable by a linear BCSS machine and decidable by one with oracle $\mathbb{L}_{d+1}$ but not with oracle $\mathbb{L}_{d-1}$.

Indicating by superscript “$\ell$” the reference to linear machines, it thus holds

$$ \mathbb{Q} \cdot \mathbb{Q} <^{\ell} \mathbb{Q} = \mathbb{L}_1 <^{\ell} \mathbb{L}_2 <^{\ell} \ldots <^{\ell} \mathbb{L}_{d-1} <^{\ell} \mathbb{L}_d <^{\ell} \mathbb{H}^{\ell} \quad (5.1) $$

We are proud of our own (hopefully more transparent and sound)

**Proof.** The two positive claims are easy: A linear machine can enumerate all numerators $n_0, n_1, \ldots, n_{d-1} \in \mathbb{Z}$ and common denominators $m \in \mathbb{N}$ of putative $(q_0, \ldots, q_{d-1}) \in \mathbb{Q}^d$ and test whether $n_0 + \sum n_i x_i = mx$ holds. Secondly, $(x_1, \ldots, x_d)$ belong to $\mathbb{L}_d$ iff $(1, x_1, \ldots, x_d)$ belongs to $\mathbb{L}_{d+1}$.

Regarding the negative claim, suppose that $\mathbb{R}^d \setminus \mathbb{L}_d$ is semi-decidable relative to $\mathbb{L}_{d-1}$. Now observe that

i) Unrolling such a hypothetical algorithm leads to a 6-ary tree with nodes $u$ assigned (tuples of) affinely linear functions $g_u : \mathbb{R}^d \ni x = (x_1, \ldots, x_d) \mapsto n_0 + \sum n_i x_i$, with branches according to $\text{sgn} g_u(x)$ and to oracle queries “$g_u(x) \in \mathbb{L}_{d-1}$?”

ii) It holds $\pi := (\pi, \pi^2, \ldots, \pi^d) \notin \mathbb{L}_d$ (i.e. $\pi$ ends up in a leaf of the above tree). Moreover, $g_u(\pi) \notin \mathbb{Q} = \mathbb{L}_1$ for any $u$ (unless $g_u$ is constant).

Indeed, $\pi$ being transcendental (Fact 2.5) means that its integer powers are linearly independent over $\mathbb{Q}$. (Any odd $d$-tuple $\pi$ of reals affinely independent over $\mathbb{Q}$ will work as well.) In particular the set of all $x \in \mathbb{R}^d$ with $\text{sgn} g_u(x) = \text{sgn} g_u(\pi) \neq 0$ is open by continuity and ii).

Now in case $d = 2$ consider $\pi' := (\pi, C \cdot \pi)$ for $r \in \mathbb{Q}$ sufficiently close to $\pi$ such that $g_u$ has the same sign. If $\mathbb{Q} \ni q = g_u(\pi') = n_0 + (n_1 + n_2) r \pi$, then $n_1 + n_2 r = 0$ because $\pi \notin \mathbb{Q}$, in which case choose some other $r$ (recall that $g_u$ is w.l.o.g. non-constant by ii). This reveals that, on input of $\pi'$, the finitely many branches based on both comparisons and oracle answers agree with those performed on input of $\pi$ and thus also lead to termination: contradicting the fact that $\pi' \in \mathbb{L}_2$.

The cases $d > 2$ require some more sophistication, to which end we generalize ii) as follows:

iii) Either $g_u(x) \in \mathbb{L}_{d-1} \forall x \in \mathbb{R}^d$ or $g_u(\pi') \notin \mathbb{L}_{d-1}$.

iv) For finitely many $g_u$, $i = 1, \ldots, k$, the subset $\{ x \in \mathbb{L}_d : g_u(x) \notin \mathbb{L}_{d-1} \forall i = 1, \ldots, k \}$ of $\mathbb{L}_d$ is dense in $\mathbb{R}^d$.

Indeed, this follows from applying Lemma 5.25(b) below to the homogeneously linear $\tilde{f}(x_0, x) := (1, x_0 - 1) \tilde{g}_u(0) + \tilde{g}_u(x)$ and $\tilde{x} := (1, \pi, \pi^2, \ldots, \pi^d)$: linear dependence of the component functions of $\tilde{g}_u$ can (according to the first case of iii), without loss of generality be ruled out as redundant an oracle query. By i) and iv), there exist inputs $\pi' \in \mathbb{L}_d$ for which sign tests and oracle queries give the very same results as for input $\pi \notin \mathbb{L}_d$ and in particular lead to termination: contradiction.

**Lemma 5.25.** Fix linear functions $f_1, \ldots, f_n : \mathbb{Q}^m \to \mathbb{Q}$, $f_i(x_1, \ldots, x_m) = \sum a_{ij} x_j$ with $A = (a_{ij}) \in \mathbb{Q}^{m \times m}$.

a) If $f_1, \ldots, f_n$ (i.e. the rows $a_{i\bullet}$ of $A$, $i = 1, \ldots, n$) are linearly dependent over $\mathbb{Q}$, then so are the real (!) numbers $f_1(x), \ldots, f_n(x)$ for any choice of $x \in \mathbb{R}^m$.

b) If both $f_1, \ldots, f_n$ and $x_1, \ldots, x_m \in \mathbb{R}$ are linearly independent over $\mathbb{Q}$, then so are the real numbers $f_1(x), \ldots, f_n(x)$.

c) Suppose $\text{rank}_Q(A) = n < m$. Then the set of vectors $(x_1, \ldots, x_m) \in \mathbb{R}^m$ which are linearly dependent over $\mathbb{Q}$ but mapped to linearly independent numbers $f_1(x), \ldots, f_n(x)$ is dense.
d) The same holds for finitely many matrices $A_1, \ldots, A_k \in \mathbb{Q}^{n \times m}$: If they all have rank$_{\mathbb{Q}}(A) = n < m$, then the following set is dense in $\mathbb{R}^m$:

$$\{ \vec{x} \in \mathbb{R}^m \text{ linearly dependent over } \mathbb{Q} \land A_j \cdot \vec{x} \text{ linearly independent over } \mathbb{Q} \forall j = 1, \ldots, k \}$$

Of course $\mathbb{Q} \subseteq \mathbb{R}$ may be replaced by any other field extension $F \subseteq E$ in Claims a) and b). Claim c) trivially fails for rank$_{\mathbb{Q}}(A) < n$ or for $n = m$; and so does d) for the case of countably many matrices.

**Proof.**

a) By prerequisite there exists non-zero matrices.

b) Let $\vec{z} \in \mathbb{Q}^n$ be such that $\vec{z}^\dagger \cdot A = 0 \in \mathbb{R}^m$; hence

$$\sum_i z_i f_i(\vec{x}) = \vec{z}^\dagger \cdot (A \cdot \vec{x}) = 0 \in \mathbb{R}.$$  

b) Let $\vec{z} \in \mathbb{Q}^n$ such that $A \supseteq 0 = \sum_i z_i f_i(\vec{x}) = \vec{z}^\dagger \cdot A \cdot \vec{x}$. Then $\mathbb{Q}^m \ni \vec{z}^\dagger \cdot A = 0$ because the $x_j$ are linearly independent over $\mathbb{Q}$; hence $\vec{z} = 0$ by linear independence of $f_1, \ldots, f_n$.

c) Observe that $f_1(\vec{x}), \ldots, f_n(\vec{x})$ are linearly dependent over $\mathbb{Q}$ iff $\vec{x}$ is perpendicular ('⊥') to some $\vec{z} = A^\dagger \cdot \vec{y}$ with non-zero $\vec{y} \in \mathbb{Q}^n$. Since rank$(A^\dagger) = n$ by prerequisite, such $\vec{z}$ must itself be non-zero:

$$f_1(\vec{x}), \ldots, f_n(\vec{x}) \text{ linearly dependent over } \mathbb{Q} \iff \exists \vec{z} \in \text{range}_{\mathbb{Q}}(A^\dagger) \setminus 0 : \vec{x} \bot \vec{z}. \quad (5.2)$$

Also by prerequisite, range$_{\mathbb{Q}}(A^\dagger)$ is strictly a subspace of $\mathbb{Q}^m$; hence $\mathbb{Q}^m \setminus \text{range}_{\mathbb{Q}}(A^\dagger)$ is dense. Any $\vec{z} \in \mathbb{Q}^m \setminus \text{range}_{\mathbb{Q}}(A^\dagger)$ gives rise to a (real) hyperplane $\perp \vec{z} := \{ \vec{x} \in \mathbb{R}^m : \vec{x} \bot \vec{z} \}$ of vectors linearly dependent over $\mathbb{Q}$. By the choice of $\vec{z} \notin \text{range}_{\mathbb{Q}}(A^\dagger)$, $\perp \vec{z}$ is neither included in, nor coincides with, the real hyperplane $\perp \vec{y}$ for any $\vec{y} \in \text{range}_{\mathbb{Q}}(A^\dagger)$. (Observe that $\vec{z} \neq 0$!)

Thus, $(\perp \vec{z}) \cap (\perp \vec{y})$ constitutes a strict, nowhere dense subspace of $\perp \vec{z}$. Removing from $\perp \vec{z}$ the countably many subspaces $(\perp \vec{z}) \cap (\perp \vec{y})$ with $\vec{y} \in \text{range}_{\mathbb{Q}}(A^\dagger) \setminus 0$ leaves, by Baire’s Category Theorem, a dense subset of $\perp \vec{z}$: consisting according to Equation (5.2) only of vectors $\vec{x}$ linearly dependent over $\mathbb{Q}$ such that $f_1(\vec{x}), \ldots, f_n(\vec{x})$ are linearly independent over $\mathbb{Q}$. Since $\vec{z}$ was arbitrary from the dense set $\mathbb{Q}^m \setminus \text{range}_{\mathbb{Q}}(A^\dagger)$, this proves the claim.

d) follows similarly to c) by considering $\vec{z} \in \mathbb{Q}^m \setminus \bigcup_j \text{range}_{\mathbb{Q}}(A^\dagger_j)$.  

\[\square\]

### 5.4 Arithmetical Hierarchy

The diagonalization argument establishing the undecidability of the real Halting Problem $\mathbb{H} =: \mathcal{O}'$ (Fact 2.17d) relativizes and yields by iteration a strict hierarchy of problems

$$\mathcal{O}^{(d)} := \{ \langle M', \vec{x} \rangle \mid M^{\mathcal{O}^{(d-1)}} \text{ terminates on input } \vec{x} \} \subseteq \mathbb{R}^* \quad (5.3)$$

similar to Equation (3.2). In the discrete case, Theorem 3.9 established an equivalent characterization in syntactical terms. However, any problem $L \subseteq \mathbb{N}$ of the form (3.3) is BCSS-decidable by Example 2.20, and replacing $X = \mathbb{N}$ with $X = \mathbb{R}$ in (3.3) does not render it undecidable either because of Fact 2.13. Instead, Cucker in [Cuck92], motivated by Corollary 2.27, arrives at the following

**Definition 5.26.** For $d, k \in \mathbb{N}$ and a field $F \subseteq \mathbb{R}$, consider the class $\Sigma_{d,F}(2^k)$ of real languages of the form

$$\{ \vec{x} \in \mathbb{R}^k \mid \exists n_1 \in \mathbb{N} \forall n_2 \in \mathbb{N} \exists n_3 \in \mathbb{N} \ldots \theta_d n_d \in \mathbb{N} : \vec{x} \in B_{(n_1, \ldots, n_d)} \} \quad (5.4)$$

where $B_r \subseteq \mathbb{R}^k$ is a sequence of sets (not necessarily basic-) semi-algebraic over $F$; recall Definition 2.10. Furthermore let $\Sigma_{d,F}(2^k) := \bigcup_k \Sigma_{d,F}(2^k)$ and $\Sigma_d(2^\mathbb{R}) := \bigcup_F \Sigma_{d,F}(2^\mathbb{R})$ where the last union is understood to range over all finite real field extensions: $F = \mathbb{Q}(c_1, \ldots, c_j)$, $j \in \mathbb{N}$, $c_1, \ldots, c_j \in \mathbb{R}$. Similarly for classes $\Pi_{d,F}$ and $\Delta_d, \ldots$. 
Hence $\Sigma_1(2^\mathbb{R})$ are the BCSS semi-decidable problems. Even just this bottom class ($d = 1$) lies skew to $\Sigma_d$ and $\Pi_d$, as we illustrate by extending [Zion98, Section 3]:

**Lemma 5.27.** Every set $U \in \Sigma_1(2^\mathbb{R})$ is BCSS semi-decidable, but not every set in $\Pi_1(2^\mathbb{R})$ is. Conversely, every BCSS semi-decidable $L \subseteq \mathbb{R}^k$ belongs to $\Sigma_2(2^\mathbb{R}^k)$ but not necessarily to $\Sigma_2(2^\mathbb{R}^k)$ or to $\Pi_2(2^\mathbb{R}^k)$.

**Proof.** Every open set $U \subseteq \mathbb{R}^k$ is a countable union of certain open rational balls $U = \bigcup_n B(\vec{c}_n, r_n)$; (recall Section 2.2.2). By encoding the sequence of centers $\vec{c}_n$ and radii $r_n$ in a real constant, a BCSS machine may effectively search for some $n$ such that a given input belongs to $B(\vec{c}_n, r_n)$ and thus semi-decide $U$. The Cantor set belongs to $\Pi_1(2^\mathbb{R})$ (Example 2.18) but is not BCSS semi-decidable (Example 2.18).

Conversely, every basic semi-algebraic subset of $\mathbb{R}^k$ is by Definition 2.10 a finite intersection of pre-images $q_i^{-1}([0, \infty))$ and $p_i^{-1}([0])$ of open or closed sets, respectively, under (continuous) polynomials and thus (Section 2.2.2) belongs to $\Sigma_1(2^\mathbb{R}) \cup \Pi_1(2^\mathbb{R}) \subseteq \Sigma_2(2^\mathbb{R})$. According to Corollary 2.27(a), a semi-decidable set is a countable union of basic semi-algebraic ones and therefore still an element of $\Sigma_2(2^\mathbb{R})$.

The set $\mathbb{Q}$ of rational numbers is BCSS semi-decidable (Example 2.21) but not in $\Pi_1(2^\mathbb{R})$ (Example 2.50). The complement $\mathbb{R} \setminus \{c\}$ of every singleton is trivially BCSS-decidable but, for $c := \sum_{n \in \mathbb{Q}} 2^{-n} \in \Sigma_2(\mathbb{R}) \setminus \Pi_2(\mathbb{R})$, not in $\Sigma_2(2^\mathbb{R})$ according to Fact 4.15(b). \qed

In contrast, the following refinement of [Cuck92, Theorems 2.11 and 2.13] holds:

**Theorem 5.28.** For $L \subseteq \mathbb{R}^k$, $d \in \mathbb{N}$, and $c_1, \ldots, c_j$, the implication chain “$a \Rightarrow b \Rightarrow c \Rightarrow d$” holds where $a)$ to $d)$ refer to the following claims:

- **a)** $L$ is BCSS semi-decidable by a $\mathcal{O}^{(d-1)}$-oracle machine with constants $c_1, \ldots, c_j$.
- **b)** $L \in \Sigma_{d,F}(2^\mathbb{R})$ where $F = \mathbb{Q}(c_1, \ldots, c_j)$.
- **c)** There is some $c_{j+1} \in \mathbb{R}$ and a many-one reduction $f : \mathbb{R}^* \to \mathbb{R}^*$ from $L$ to $\mathcal{O}^{(d)}$, computable by a BCSS-machine with constants $c_1, \ldots, c_j, c_{j+1}$.
- **d)** $L$ is BCSS semi-decidable by a $\mathcal{O}^{(d-1)}$-oracle machine with constants $c_1, \ldots, c_j, c_{j+1}$ from $c$.

**Proof.** $c \Rightarrow d)$ $\mathcal{O}^{(d)}$ is semi-decidable relative to $\mathcal{O}^{(d-1)}$ by the universal BCSS-machine without constants.

$b \Rightarrow c)$ holds by induction: For $d = 1$, encode the rational coefficients of the descriptions of the (polynomials whose solutions constitute the) sequence of sets $(B_{n,k})$ as elements in $F = \mathbb{Q}(c_1, \ldots, c_j)$ in the real number $c_j$; compare Lemma 5.26. Then devise a BCSS machine $M$ which, using this encoded information, generates and searches the sequence $(B_{n,k})$ for a member to contain a given input $\vec{x} \in \mathbb{R}^k$ and which terminates when one is found. The mapping $\vec{x} \mapsto (M, \vec{x})$ is thus a reduction from $L$ to $\mathcal{O}'$ computable by a machine with constants $c_1, \ldots, c_j, c_{j+1}$.

The induction step now proceeds similarly to that of Theorem 3.9 with $K$ replaced by

$$K := \{(\vec{x},n_1) \mid n_1 \in \mathbb{N}, \forall n_2 \in \mathbb{N} \exists n_3 \in \mathbb{N} \ldots \exists n_d \in \mathbb{N} : \vec{x} \in B(n_1, \ldots, n_d, m)\}.$$

**a \Rightarrow b)** also proceeds inductively, the case $d = 1$ having been treated in Corollary 2.27(a). For the inductive step $d \geq 2$, unroll a computation of a $\mathcal{O}^{(d-1)}$-oracle machine on input $\vec{x} \in \mathbb{R}^k$, as described above. Let $g_u \in F[X_1, \ldots, X_k]$ denote the quolynomial (vector) associated with tree node $u$. By induction hypothesis, it holds $\mathcal{O}^{(d-1)} \in \Sigma_{d-1, Q}(2^\mathbb{R})$: a universal BCSS-machine without constants can semi-decide $\mathcal{O}^{(d-1)}$ relative to $\mathcal{O}^{(d-2)}$. According to Lemma 5.29(c) below, the inputs $\vec{x}$ leading to positive/negative answers to the single oracle
query \( g_u(\vec{x}) \in \emptyset^{(d-1)} \) thus belong to sets in \( \Sigma_{d-1,F} \) and \( \Pi_{d-1,F} \), respectively; that is, in \( \Sigma_{d,F} \). And the sign conditions \( g_u(\vec{x}) : 0 \) give rise to sets in \( \Delta_{1,F} \subseteq \Sigma_{d,F} \), too. A leaf set \( \mathbb{X}_v \) of the above \( \emptyset^{(d-1)} \)-oracle machine, thus being a finite intersection of sets in \( \Sigma_{d,F} \), therefore still lies in \( \Sigma_{d,F} \) by virtue of Lemma 5.29); and so does the set \( L \) accepted by the machine, that is the countable union over these \( \mathbb{X}_v \): Lemma 5.29(b).

We point out some interesting differences between (the last step of) the proof of Theorem 5.28 and that of its discrete counterpart: on the one hand the quantifiers over \( \mathbb{N} \) in Equation (5.4) would not merge (or commute) with one ranging over a finite collection \( \mathbb{Q}_k \) of putative real oracle queries; on the other hand, the (merely) decidable set \( R \) in Equation (5.3) is in (5.4) replaced by the much more explicit countably semi-algebraic sequence \( (\mathbb{B}_{n,m}) \). Thus, instead of Lemma 5.11 one now has the following

**Lemma 5.29 (Closure Properties).** Fix \( d \in \mathbb{N} \) and field \( F \subseteq \mathbb{R} \).

1. \( \Sigma_{d,F} (2^{2^d}) \subseteq \Pi_{d+1,F} (2^{2^d}) \).
   
   If \( E \subseteq \mathbb{R} \) is a field extension of \( F \), then \( \Sigma_{d,F} (2^{2^d}) \subseteq \Sigma_{d,E} (2^{2^d}) \).

2. For \( k \in \mathbb{N} \) and \( L \subseteq \mathbb{R}^k \), \( L \subseteq \Sigma_{d,F} (2^{2^d}) \iff \mathbb{R}^k \setminus L \in \Pi_{d,F} (2^{2^d}) \).

3. For \( f : \mathbb{R}^k \to \mathbb{R}^k \) a vector of polynomials over \( F \) and \( L \subseteq \Sigma_{d,F} (2^{2^d}) \), it holds \( f^{-1}[L] \subseteq \Sigma_{d,F} (2^{2^d}) \).

4. Let \( L_1, L_2 \subseteq \Sigma_{d,F} (2^{2^d}) \). Then \( L_1 \cap L_2 \subseteq \Sigma_{d,F} (2^{2^d}) \).

5. Let \( L_n \subseteq \Sigma_{d,F} (2^{2^d}) \) for \( n \in \mathbb{N} \) implies that also \( \bigcup_n L_n \subseteq \Sigma_{d,F} (2^{2^d}) \).

**Dual claims hold for \( \Pi \)-classes.**

### 5.4.1 Completeness and Other Applications of the Normal Form Theorem

Neglecting the exact choice and number of real constants, one arrives at a formulation even more similar to Post’s Theorem 3.9

**Corollary 5.30.** A language \( L \subseteq \mathbb{R}^* \) is semi-decidable by an \( \emptyset^{(d-1)} \)-oracle BCSS machine iff there exists a BCSS-decidable \( P \subseteq \mathbb{R}^* \) such that

\[
L = \left\{ \vec{x} \in \mathbb{R}^* \mid \exists n_1 \in \mathbb{N} \forall n_2 \in \mathbb{N} \exists n_3 \in \mathbb{N} \ldots \theta_d n_d \equiv \mathbb{N} : (\vec{x}; n_1, \ldots, n_d) \in P \right\} .
\]

**Proof.** In view of Theorem 5.28 let \( L \subseteq \Sigma_{d,F} \) and \( B_{m,k} \subseteq \mathbb{R}^k \) be basic semi-algebraic over \( F = \mathbb{Q}(c_1, \ldots, c_j) \subseteq \mathbb{R} \). Consider

\[
P := \bigcup_{n_1, \ldots, n_d} B_{(n_1, \ldots, n_d),k} \times \{(n_1, \ldots, n_d)\} \subseteq \mathbb{R}^* .
\]

This language is decidable by a BCSS machine \( M \) with constants \( c_1, \ldots, c_j \): Given \( (\vec{x}, n_1, \ldots, n_d) \in \mathbb{R}^k \times \mathbb{N}^d \) (for inputs not of this form, \( M \) may reject right away) simply check whether \( \vec{x} \in B_{(n_1, \ldots, n_d),k} \) holds. Moreover we have, for any choice of \( (n_1, \ldots, n_{d-1}) \),

\[
\theta_d n_d \equiv \mathbb{N} : \vec{x} \in B_{(n_1, \ldots, n_{d-1}),k} \iff \theta_d n_d \equiv \mathbb{N} : (\vec{x}; n_1, \ldots, n_{d-1}, n_d) \in P
\]

and thus the claimed form (5.5).

Conversely let \( L \) be of this form with \( P \subseteq \mathbb{R}^* \) decidable by a machine with constants \( c_1, \ldots, c_j \). Then, by Lemma 5.26 \( P = \bigcup_{m,k} B_{m,k} \) with \( B_{m,k} \subseteq \mathbb{R}^k \) basic semi-algebraic over \( F = \mathbb{Q}(c_1, \ldots, c_j) \). We shall suppose w.l.o.g. that \( d \) is odd, i.e. \( \theta_d = \exists \); otherwise apply the above representation to the complement of \( P \). Define

\[
B_{(n_1, \ldots, n_{d-1}, n_{d,m}),k} := \{ \vec{x} \in \mathbb{R}^k \mid (\vec{x}, n_1, \ldots, n_{d-1}, n_d) \in P \}
\]
In Theorem 5.28 the implication “\( b \Rightarrow c \)” means that \( \mathcal{O}^{(d)} \) is complete for \( \Sigma_d (2^{\mathbb{R}}) \); compare Example 3.12. Another complete problem is given in [Cuck92 Theorem 2.13], a uniform variant of Theorem 5.28.

Scholium 5.31. For \( F = \mathbb{Q}(c_1, \ldots, c_d) \), let \( \mathbb{B}_{n,k} \subseteq \mathbb{R}^k \) be a sequence of sets semi-algebraic over \( F \).

Encode this sequence as the real \((j + 1)\)-tuple consisting of \( c_1, \ldots, c_j \) plus \( c_j \in \mathbb{R} \) containing the sequence of the rational coefficients (over \( F \)) of the polynomials describing \( \mathbb{B}_{n,k} \); recall Example 2.17 and Fact 2.32. Write \( (\ell) := (c_1, \ldots, c_j, c_{j+1}) \) for this encoding of \( \ell = \bigcup_{n,k} \mathbb{B}_{n,k} \in \Sigma_d F (2^{\mathbb{R}}) \), compare Equation (5.4) in Definition 5.26. Then the language

\[
\mathcal{S}_d := \{ (\ell, x) \mid \ell \in \Sigma_d, x \in \ell \}
\]

is \( \Sigma_d \)-complete, that is it holds \( \mathcal{S}_d \in \Sigma_d (2^{\mathbb{R}}) \) and every \( \ell \in \Sigma_d (2^{\mathbb{R}}) \) satisfies \( \mathcal{L} \subset \mathcal{S}_d \).

A counterpart to Example 3.12(b), [Cuck92 Theorem 2.15] also establishes a real analogue of \( \text{FIN} \) to be \( \Sigma_2 \)-complete.

By combining Lemma 5.27 and Theorem 5.28 we conclude [Cuck92 Theorem 4.3]:

Corollary 5.32. For each \( d \in \mathbb{N} \), the class \( \Delta_d (2^{\mathbb{R}}) \) of real subsets BCSS semi-decidable relative to \( \mathcal{O}^{(d-1)} \) is strictly between \( \Sigma_d \) and \( \Delta_{d+1} \); that \( \Delta_d (2^{\mathbb{R}}) \) of sets BCSS decidable relative to \( \mathcal{O}^{(d-1)} \) is between \( \Delta_d \) and \( \Delta_{d+1} \).

In particular, the (finite) Borel Hierarchy of Euclidean subsets \( \bigcup_{d,k} \Sigma_d (\mathbb{R}^d) \) coincides with the (finite) BCSS Arithmetical Hierarchy \( \bigcup_{d} \Sigma_d (2^{\mathbb{R}}) \).

Proof. By induction on \( d \) with inductive start \( d = 1 \) treated in Lemma 5.27.

Observe that in Example 5.27 the relatively computable real function \( f \) with image \( \mathcal{L} \) lacks injectivity. This can be remedied by proceeding from a total to a partial \( f \). More seriously, the oracle \( \mathcal{O} \) under consideration belongs to but is not complete for \( \Pi_1 \subset \Pi_2 \) (Lemma 5.27). On the other hand, for injective functions computable relative to a \( \Sigma_d \)-complete oracle, the real counterpart to Fact 2.17(b) does hold:

Proposition 5.33. Let (possibly partial) \( f : \subseteq \mathbb{R}^* \rightarrow \mathbb{R}^* \) be injective and computable relative to \( \mathcal{O}^{(d)} \) by a machine with constants \( c_1, \ldots, c_D \). Then range(\( f \)) is semi-decidable relative to \( \mathcal{O}^{(d)} \) by a machine with constants \( c_1, \ldots, c_D \).

Proof. By “\( n \Rightarrow b \)” of Theorem 5.28 \( \text{dom}(f) \in \Sigma_{d+1,F} \) where \( F := \mathbb{Q}(c_1, \ldots, c_D) \); that is

\[
\text{dom}(f) = \bigcup_{k \in \mathbb{N}} \bigcup_{n_1 \in \mathbb{N}} \bigcap_{n_2 \in \mathbb{N}} \bigcup_{n_3 \in \mathbb{N}} \bigcup_{n_{d+1} \in \mathbb{N}} \mathbb{B}_{(n_1, \ldots, n_d),k}
\]

where \( \mathbb{B}_{n,k} \subseteq \mathbb{R}^k \) is semi-algebraic over \( F \) (and \( \bigcup \) denotes either union or intersection depending on \( d \)'s parity). Moreover \( f_{n,k} := f|_{\mathbb{B}_{n,k}} \) is a quynomial vector over \( F \); hence \( f|_{\mathbb{B}_{n,k}} \in \mathbb{R}^k \) is semi-algebraic over \( F \), too, by virtue of Quantifier Elimination (Fact 2.13). Therefore

\[
\text{range}(f) = f[\text{dom}(f)] = f \left[ \bigcup_{k \in \mathbb{N}} \bigcup_{n_1 \in \mathbb{N}} \bigcup_{n_2 \in \mathbb{N}} \bigcup_{n_3 \in \mathbb{N}} \bigcup_{n_{d+1} \in \mathbb{N}} \mathbb{B}_{(n_1, \ldots, n_d),k} \right]
\]

belongs to \( \Sigma_{d+1,F} \) according to (Lemma 5.29). Step (*) crucially relies on \( f \) being injective on its domain! Finally, Claim “\( b \Rightarrow d \)” of Theorem 5.28 asserts range(\( f \)) semi-decidable relative to \( \mathcal{O}^{(d)} \) by a machine with an additional real constant \( c_{D+1} \). Closer inspection reveals that both sequences \( (\mathbb{B}_{n,k}) \) and \( (f_{n,k}) \) are uniform, therefore this constant can be disposed of.
5.4.2 Nondeterministic and Analytic Hierarchy

By Corollary 2.31, the Halting Problem for a non-deterministic BCSS machine is the same as that for a deterministic one. Put differently, the nondeterministic class \( \Sigma^1_{1,F} \) (\( 2^R \)) of languages of the form

\[
\{ \vec{x} \in R^k \mid \exists \vec{y} \in R^*: (\vec{x}, \vec{y}) \in L \} = \{ \vec{x} \in R^k \mid \exists k, \ell \in N \exists \vec{y} \in R^\ell \exists n \in N : (\vec{x}, \vec{y}) \in B_{(k,\ell,n)} \}
\]

(\( L \in \Sigma^1_{1,F} \), \( B \) semi-algebraic over \( F \)) coincides with deterministic \( \Sigma^1_{1,F} \) because existential quantifiers commute and the real one “\( \exists \vec{y} \)” can thus be merged into the \( B \)'s by virtue of Tarski’s Fact 2.13. However, when we proceed to oracle computation, we already saw in Example 5.7 that Quantifier Elimination may fail. Indeed, nondeterministic semi-decidability relative to \( H \) can be characterized \cite{Cuck92} Theorem 3.5 by the class \( \Sigma^2_{3,F} \) (\( 2^{R^k} \)) of languages of the form

\[
\{ \vec{x} \in R^k \mid \exists \vec{y} \in R^*: (\vec{x}, \vec{y}) \in L \} = \{ \vec{x} \in R^k \mid \exists \vec{y} \in R^\ell \exists n_1 \exists n_2 \in N : (\vec{x}, \vec{y}) \in B_{(k,\ell,n_1, n_2)} \}
\]

(\( L \in \Sigma^2_{3,F} \), \( B \) semi-algebraic over \( F \)) where we cannot move real “\( \exists \vec{y} \)” past “\( \exists n_2 \)” for further elimination. As a matter of fact, \cite{Cuck92} Theorem 4.4 establishes \( \Sigma^2_{3} \) (\( 2^{R^k} \)) as powerful as to coincide with the class \( \Sigma^3_{1} \) of analytic subsets of \( R^k \); recall Section 2.2.2. Hence \( \Delta^3_{3} \) (\( 2^{R^k} \)) equals the class \( \Delta^3_{2} \) of Borel sets and thus strictly contains the finite deterministic Arithmetical BCSS Hierarchy. Moreover, the real counterpart to Totality \( \text{Tot} \) turns out as \( \Pi^2_1 \)-complete \cite{Cuck92} Theorem 3.9.

For the \( d \)-th nondeterministic level, \cite{Cuck92} arrives at

**Definition 5.34.** For \( F = Q(c_1, \ldots, c_j) \subseteq R \) let \( \Sigma^d_{a,F} \) denote the class of languages of the form

\[
\{ \vec{x} \in R^k \mid \exists N_1 \exists N_2 \exists N_3 \ldots \exists N_d : (\vec{x}, \vec{y}_1, \ldots, \vec{y}_d) \in B_{(k, n_1, n_2, n_3, \ldots, n_d)} \}
\]

with \( B \) semi-algebraic over \( F \).

5.5 Real Computational Universality

The preceding sections have revealed BCSS computability theory to be quite similar to recursion theory for discrete Turing machines. Nevertheless, the current state of the first is still far from achieving the depth and breadth to which the latter has been developed \cite{Odif89, Soa87}. This is due to fundamental differences between the two realms, which require many proofs in the real setting—although for claims parallel to the discrete setting—to proceed considerably differently. For instance, Section 5.3 established explicitly an undecidable real problem strictly easier than the real Halting problem.

As a matter of fact, most classical proofs for the undecidability of an r.e. problem \( P \) proceed by reduction from \( H \)—and thus implicitly establish \( P \) as Turing complete in the sense that \( P \) supports universal computation. The undecidability of a real problem \( \mathbb{P} \) in contrast quite often exploits algebraic properties of BCSS computation; recall, for instance, Example 2.18. It thus does not (and, at least for \( \mathbb{P} = \mathbb{Q} \), provably cannot) imply \( \mathbb{P} \) to be computationally universal. Therefore, where the challenge in the discrete case consisted in finding an undecidable problem not reducible from \( H \), we are in the real case confronted with

**Question 5.35.** Are there ‘natural’ problems \( \mathbb{P} \) BCSS-complete in the sense that both \( \mathbb{P} < H \) and \( H < \mathbb{P} \) hold?
One such problem, $S_1$, has been identified in Scholium 5.31. However, its definition exhibits relations to semi-algebraic geometry and BCSS computability, hence completeness may not come as too great a surprise. This differs significantly from the two milestones in discrete uncomputability mentioned in Section 1.1.2. Both the Word Problem for finitely presented groups and Hilbert’s Tenth Problem arise naturally within pure mathematics; and the importance of their undecidability consists to a considerable part in the (unexpected) relation of these problems to computer science!

Now an (at least straightforward) translation of Hilbert’s Tenth Problem to the reals fails BCSS–completeness, recall Lemma 2.30. So we turn in the following subsections to the

5.5.1 Word Problem for Groups

Groups occur ubiquitously in mathematics, and having calculations with and in them handled by computers constitutes an important tool both in their theoretical investigation and in practical applications, as revealed by the flourishing field of Computational Group Theory [FiKa91, FiKa95, HEoB05]. Unfortunately even the simplest question, namely the equality ‘$a = b$’ of two elements $a, b \in G$ is generally undecidable for groups $G$ that are reasonably presentable to a digital computer, that is, in a finite way—the celebrated result obtained independently by Novikov [Novi59] and Boone [Boon58] in the 1950ies. To a BCSS machine, on the other hand, it is trivially decidable (Example 2.25).

However, whenever we deal with computational questions involving groups of real or complex numbers, the Turing model seems inappropriate anyway. As an example, take the unit circle $S^1$. The word problem for the unit circle is universal for the BCSS model. At a continuous counterpart to the discrete class of finitely presented groups for which the word problem is universal, recall Lemma 2.30. So we turn in the following subsections to the

5.5.2 Classical Setting

We briefly recall the classical word problem:

Definition 5.36. a) Let $X$ be a set. The free group generated by $X$, denoted by $F = \langle X, \circ \rangle$ or more briefly $\langle X \rangle$, is the set $(X \cup X^{-1})^*$ of all finite sequences $\bar{w} = x_1^{\epsilon_1} \cdots x_n^{\epsilon_n}$ with $n \in \mathbb{N}$, $x_i \in X$, $\epsilon_i \in \{-1, +1\}$, equipped with concatenation $\circ$ as the group operation subject to the rules

\[ x \circ x^{-1} = 1 = x^{-1} \circ x \quad \forall x \in X \]  

(5.7)

where $x^1 := x$ and where 1 denotes the empty word, that is, the unit element.

b) For a group $H$ and $W \subseteq H$, denote by

\[ \langle W \rangle_H := \{ w_1^{\epsilon_1} \cdots w_n^{\epsilon_n} : n \in \mathbb{N}, w_i \in W, \epsilon_i = \pm 1 \} \]

the subgroup of $H$ generated by $W$. The normal subgroup of $H$ generated by $W$ is

\[ \langle W \rangle_{hn} := \langle \{ h \cdot w \cdot h^{-1} : h \in H, w \in W \} \rangle_H . \]

For $h \in H$, we write $h/W$ for its $W$–coset $\{ h \cdot w : w \in \langle W \rangle_{hn} \}$ of all $g \in H$ with $g \equiv_W h$.

c) Fix sets $X$ and $R \subseteq \langle X \rangle$ and consider the quotient group $G := \langle X \rangle/\langle R \rangle_n$, denoted by $\langle X \mid R \rangle$, of all $R$–cosets of $\langle X \rangle$.

If both $X$ and $R$ are finite, the tuple $(X, R)$ will be called a finite presentation of $G$; if $X$ is finite and $R$ recursively enumerable (by a Turing machine, that is in the discrete sense; equivalently: semi-decidable), it is a recursive\footnote{21} presentation; if $X$ is finite and $R$ arbitrary, $G$ is finitely generated.

\[ ^{21}\text{This notion seems misleading as } R \text{ is in general not recursive; nevertheless it has become established in literature.} \]
Intuitively, $R$ induces further rules \( \bar{w} = 1 \), \( \bar{w} \in R \) in addition to Equation (5.7); put differently, distinct words \( \bar{u}, \bar{v} \in \langle X \rangle \) might satisfy \( \bar{u} = \bar{v} \) in \( G \), that is, by virtue of $R$. Observe that the rule \( \bar{w}_1 \cdots \bar{w}_k = 1 \) induced by an element \( \bar{w} = (\bar{w}_1 \cdots \bar{w}_n) \in R \) can also be applied as \( \bar{w}_i \cdots \bar{w}_k = \bar{w}_n \cdots \bar{w}_{k+1} \).

**Definition 5.36 (continued).** d) The word problem for \( \langle X \mid R \rangle \) is the task of deciding, given \( \bar{w} \in \langle X \rangle \), whether \( \bar{w} = 1 \) holds in \( \langle X \mid R \rangle \).

The famous work of Novikov and, independently, Boone establishes the Word Problem for finitely presented groups to be Turing-complete:

**Fact 5.37.** a) For any finitely presented group \( \langle X \mid R \rangle \), its associated word problem is semi-decidable (by a Turing machine).

b) There exists a finitely presented group \( \langle X \mid R \rangle \) whose associated word problem is many-one reducible by a Turing machine from the discrete Halting Problem \( H \).

Of course, a) is immediate. For the highly nontrivial Claim b), see for instance one of [Boon58, Novi59, LySs77, Rotm95].

**Example 5.38.** \( \mathcal{H} := \{ (a,b,c,d) \mid \{ a^{-i}ba^i = c^{-i}dc^i : i \in \mathbb{N} \} \} \) is a recursively presented group whose word problem is reducible from \( H \); compare the proof of [LySs77, Theorem §IV.7.2].

In order to establish Fact 5.37(b), we need a finitely presented group. This step is provided by the remarkable

**Fact 5.39 (Higman Embedding Theorem).** Every recursively presented group can be embedded in a finitely generated one.

*Proof.* See, e.g., [LySs77, Section §IV.7] or [Rotm95, Theorem 12.18].

Fact 5.39 ensures the word problem from Example 5.38 to be in turn reducible to that of the finitely presented group \( \mathcal{H} \) is embedded in, because any such embedding is automatically effective:

**Observation 5.40.** Let \( G = \langle X \rangle / \langle R \rangle_n \) and \( H = \langle Y \rangle / \langle S \rangle_n \) denote finitely generated groups and \( \psi : G \rightarrow H \) a homomorphism. Then \( \psi \) is (Turing-)computable in the sense that there exists a computable homomorphism \( \psi' : \langle X \rangle \rightarrow \langle Y \rangle \) such that \( \psi' (\bar{x}) \in \langle S \rangle_n \) whenever \( \bar{x} \in \langle R \rangle_n \); that is, \( \psi' \) maps \( R \)-cosets to \( S \)-cosets and makes the following diagram commute:

\[
\begin{array}{c}
\langle X \rangle \xrightarrow{\psi'} \langle Y \rangle \\
\downarrow \quad \downarrow \\
\langle X \rangle / \langle R \rangle_n \xrightarrow{\psi} \langle Y \rangle / \langle S \rangle_n
\end{array}
\] (5.8)

Indeed, due the homomorphism property, \( \psi \) is uniquely determined by its values on the finitely many generators \( x_i \in X \) of \( G \), that is, by \( \psi(x_i) = w_i / \langle S \rangle_n \) where \( w_i \in \langle Y \rangle \). Setting (and storing in a Turing Machine) \( \psi'(x_i) := w_i \) yields the claim.

### 5.5.3 Presenting Real Groups

Regarding the BCSS-machine as a natural extension of the Turing machine from the discrete to the reals, the following is equally natural a generalization of Definition 5.36(d):

**Definition 5.41.** Let \( X \subseteq \mathbb{R}^* \) and \( R \subseteq \langle X \rangle \). The tuple \( \langle X, R \rangle \) is called a presentation of the real group \( G = \langle X \mid R \rangle \). This presentation is algebraically generated if \( X \) is BCSS-decidable.

\( ^{22} \)Most formally, \( R \) is a set of vectors of vectors of varying lengths. However, by suitably encoding delimiters, we shall regard \( R \) as effectively embedded in single vectors of varying lengths.
and \( X \subseteq \mathbb{R}^N \) for some \( N \in \mathbb{N} \). \( G \) is termed algebraically enumerated if \( R \) is in addition BCSS semi-decidable; if \( R \) is even BCSS-decidable, call \( G \) algebraically presented. The word problem for the presented real group \( G = \langle X | R \rangle \) is the task of BCSS-deciding, given \( \bar{w} \in \langle X \rangle \), whether \( \bar{w} = 1 \) holds in \( G \).

<table>
<thead>
<tr>
<th>Turing</th>
<th>BCSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>finitely generated</td>
<td>algebraically generated</td>
</tr>
<tr>
<td>recursively presented</td>
<td>algebraically enumerated</td>
</tr>
<tr>
<td>finitely presented</td>
<td>algebraically presented</td>
</tr>
</tbody>
</table>

Figure 5.1: Correspondence between Classical Discrete and New Real Group Presentations

Remark 5.42. a) Although \( X \) inherits from \( \mathbb{R} \) algebraic structure such as addition + and multiplication \( \times \), the Definition \([5.36\rangle\) of the free group \( G = (\langle X \rangle, \circ) \) considers \( X \) a plain set only. In particular, (group-) inversion in \( G \) must not be confused with (multiplicative) inversion: \( 5 \circ \frac{1}{5} \neq 1 = 5 \circ 5^{-1} \) for \( X = \mathbb{R} \). This difference may be stressed notationally by writing ‘abstract’ generators \( x_\bar{a} \) indexed with real vectors \( \bar{a} \); here \( x_{\bar{a}}^{-1} \neq x_{1/\bar{a}} \) holds more intuitively.

b) Isomorphic (that is, essentially identical) groups \( \langle X | R \rangle \cong \langle X' | R' \rangle \) may have different presentations \( \langle X, R \rangle \) and \( \langle X', R' \rangle \); see Example \([5.47\rangle\) below. Even when \( R = R' \), \( X \) need not be unique! Nevertheless, we adopt from literature such as \( \text{LySS77} \) the convention\(^{23}\) of speaking of “the group \( \langle X | R \rangle \)”, meaning a group with presentation \( \langle X, R \rangle \).

This, however, requires some care, for instance when \( \bar{w} \) is considered (as in Definition \([5.36\rangle\)) both an element of \( \langle X \rangle \) and of \( \langle X | R \rangle \)! For that reason we prefer to write \( \langle W \rangle_H \) rather than for instance \( \text{Gp}(W) \), to indicate the group in which we consider a subgroup to be generated.

For a BCSS-machine to read or write a word \( \bar{w} \in \langle X \rangle = \langle X \cup X^{-1} \rangle^* \) of course means to input or output a vector \( (\bar{w}_1, \epsilon_1, \ldots, \bar{w}_n, \epsilon_n) \in (\mathbb{R}^N \times \mathbb{N})^n \). In this sense, the Rules \([5.7\rangle\) implicit in the free group are obviously decidable and may w.l.o.g. be included in \( R \).

### 5.5.4 Examples of Presented Real Groups

The purpose of this section is to convince the reader of the reasonableness of Definition \([5.41\rangle\) by showing that it captures many classical groups from pure mathematics. We begin with a trivial

Example 5.43. Every finite or recursive presentation is an algebraic presentation. Its word problem is BCSS-decidable.

As long as \( X \) in Definition \([5.36\rangle\) is at most countable, so will any group \( \langle X | R \rangle \) be. Only proceeding to real groups as in Definition \([5.41\rangle\) can include many interesting uncountable groups in mathematics.

Example 5.44. Let \( \mathbb{S} \) denote the unit circle in \( \mathbb{C} \) with complex multiplication. The following is an algebraic presentation \( \langle X | R_1 \cup R_2 \rangle \) of \( \mathbb{S} \):

- \( X := \{ x_{r,s} : (r, s) \in \mathbb{R}^2 \setminus \{0\} \} \),
- \( R_1 := \{ x_{r,s} \circ x_{a,b}^{-1} : (r, s), (a, b) \neq 0 \wedge rb = sa \wedge ar > 0 \} \),
- \( R_2 := \{ x_{r,s} \circ x_{a,b} \circ x_{u,v}^{-1} : (r, s), (a, b), (u, v) \neq 0 \wedge \}
  \wedge r^2 + s^2 = 1 \wedge a^2 + b^2 = 1 \wedge u = ra - sb \wedge v = rb + sa \} \).

\(^{23}\)This can be justified with respect to the solvability of the word problem in the case of finite presentations and nonuniformly by virtue of Tietze’s Theorem \( \text{LySS77} \) Propositions \([\text{II}.2.1\rangle\) and \([\text{II}.2.2\rangle\).
Intuitively, $R_1$ yields the identification of (generators whose indices represent) points lying on the same half line through the origin. In particular, every $x_{r,s}$ is 'equal' (by virtue of $R_1$) to some $x_{a,b}$ of 'length' $a^2 + b^2 = 1$. $R_2$ in turn identifies $x_{r,s} \circ x_{a,b}$ with $x_{u,v}$ whenever, over the complex numbers, it holds $(r + is) \cdot (a + ib) = u + iv$.

Clearly, the presentation of a group need not be unique; e.g. we also have $\mathbb{S} \cong \langle Y | R_2 \rangle$ where $Y = \{x_{r,s} : r^2 + s^2 = 1\}$. Here is a further algebraic presentation of the same group:

**Example 5.45.** Let $X := \{x_t : t \in \mathbb{R}\}$, $R := \{x_t = x_{t+1}, x_t x_s = x_{t+s} : t, s \in \mathbb{R}\}$. Then $\langle X | R \rangle$ is a 1D (1) algebraic presentation of the group $\langle [0,1), + \rangle$ isomorphic to $(\mathbb{S}, \times)$ via $t \mapsto \exp(2\pi it + ic)$ for any $c \in \mathbb{R}$. Yet none of these isomorphic presentations is BCSS-computable!

Next consider the group $\text{SL}_2(\mathbb{R})$ of real $2 \times 2$ matrices $A$ with $\det(A) = 1$. A straightforward algebraic presentation of it is given as $\langle X | R \rangle$ where $X := \{x_{a,b,c,d} : ad - bc = 1\}$ and $R := \{x_{a,b,c,d} x_{(q,r,s,t)} = x_{(u,v,w,z)} : u = aq + bs + v = av + bt + w = cq + ds + z = cr + dt\}$. Here as well as in the above examples, any group element $\tilde{w} \in \langle X \rangle$ is equivalent (w.r.t. $R$) to an appropriate single generator $x \in X$. This is different for the following alternative, far less obvious algebraic presentation:

**Example 5.46 (Weil Presentation of $\text{SL}_2(\mathbb{R})$).** For each $b \in \mathbb{R}$, write

$$U(b) := \begin{pmatrix} 1 & b \\ 0 & 1 \end{pmatrix}, \quad V := \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \quad S(a) := V \cdot U(\frac{1}{a}) \cdot V \cdot U(a) \cdot V \cdot U(\frac{1}{a}) \in \text{SL}_2(\mathbb{R}).$$

Let $X = \{x_{U(b)} : b \in \mathbb{R}\} \cup \{x_V\}$. Also let $R$ denote the union of the following four families of relations (which are easy but tedious to state formally as subsets of $\langle X \rangle$):

- **SL1:** “$U(\cdot)$ is an additive homomorphism”;
- **SL2:** “$S(\cdot)$ is a multiplicative homomorphism”;
- **SL3:** “$V^2 = S(-1)$”;
- **SL4:** “$S(a) \cdot U(b) \cdot S(1/a) = U(ba^2) \forall a, b$”.

According to [Lan85], $\langle X | R \rangle$ is isomorphic to $\text{SL}_2(\mathbb{R})$ under the natural homomorphism.

In all the above cases, the word problem — in Example 5.44 basically the question of whether $(r,s) = (1,0)$ and in Example 5.45 whether $t = 0$ — is decidable. We next illustrate that, in the real case, different presentations of the same group may affect the solvability of the word problem.

**Example 5.47.** The following are presentations $\langle X | R \rangle$ of $(\mathbb{Q}, +)$:

- **a)** $X = \{x_r : r \in \mathbb{Q}\}$, $R = \{x_{r+s} = x_r \cdot x_s : r, s \in \mathbb{Q}\}$.
- **b)** $X = \{x_{p,q} : p, q \in \mathbb{Z}, q \neq 0\}$, $R = \{x_{p,q} \cdot x_{a,b} = x_{(p+aq,qb)} : p, q, a, b \in \mathbb{Z}\} \cup \{x_{p,q} = x_{(np,nq)} : p, q, n \in \mathbb{Z}, n \neq 0\}$.
- **c)** Let $(b_i)_{i \in I}$ denote an algebraic basis of the $\mathbb{Q}$-vector space $\mathbb{R}$; w.l.o.g. $0 \in I$ and $b_0 = 1$.

Consider the linear projection $P : \mathbb{R} \rightarrow \mathbb{Q}$, $\sum_i r_i b_i \mapsto r_0$ with $r_i \in \mathbb{Q}$.

$$X = \{x_t : t \in \mathbb{R}\}, \quad R = \{x_{t+s} = x_t \cdot x_s : t, s \in \mathbb{R}\} \cup \{x_t = x_{P(t)} : t \in \mathbb{R}\}.$$

Case b) yields an algebraic presentation, a) is not even algebraically generated, but c) is. The word problem is decidable for a): e.g. by effective embedding in $(\mathbb{R}, +)$; and so is it for b), although not for c): $x_t = 1 \Leftrightarrow P(t) = 0$ but both $P^{-1}(0) = \{\sum i \in J b_i q_i : 0 \notin J \subseteq I \text{ finite}, q_j \in \mathbb{Q}\}$ and its complement are totally disconnected and uncountable, hence BCSS-undecidable.  

\(^{24}\)That is, as opposed to a Banach space basis, every vector admits of a representation as a linear combination of finitely many of these (here uncountably many) base elements.
Example 5.48. (Undecidable) Real membership "$t \in Q$" is reducible to the word problem of an algebraically presented real group: Consider

$$X = \{x_r : r \in \mathbb{R}\}, \quad R = \{x_n = x_r, x_{r+k} = x_k, x_rx_s = x(nx_r : r, s \in \mathbb{R}, n \in \mathbb{N}, k \in \mathbb{Z}\}.$$ 

Then $x_r = x_0 \Leftrightarrow r \in \mathbb{Q}$; also, $R \subseteq \mathbb{R}^2 \cup \mathbb{R}^4$ is decidable because $\mathbb{Z} \subseteq \mathbb{R}$ is.

Note that the group with the undecidable word problem given in Example 5.48 is in fact a commutative one! However, this does not establish the BCSS-hardness of the real word problem because $\mathbb{Q}$ is provably easier than the BCSS Halting Problem $\mathbb{H}$ (Section 5.3). In contrast, without the restriction to algebraically presented groups (and thus parallel to Example 5.38), it is easy to find a real group with BCSS-hard word problem:

Example 5.49. Let $X := \{x_r : r \in \mathbb{R}\} \uplus \{t\} \cong \mathbb{R} \uplus \{\infty\}$ and, for $(r_1, \ldots, r_d) \in \mathbb{R}^d$, write $\bar{w}(r_1, \ldots, r_d) := x_{r_d}^{-1} \cdots x_{r_1}^{-1} \cdot t \cdot x_1 \cdots x_d \cdot t$. It is tedious but possible to confirm that

$$\langle X \rangle \times \mathbb{R}^* \ni (\bar{g}, \bar{r}) \mapsto \bar{g}^{-1} \cdot \bar{w}|r| \bar{g} \in \langle X \rangle$$

is injective. Moreover the words $\bar{g}^{-1} \cdot \bar{w}|r| \bar{g}$ are Nielsen-reduced (cf. Fact 5.37(b)) below and thus generate a free normal subgroup in $\langle X \rangle$; recall Definition 5.36(b). Therefore, for r.e. relations $R := \{\bar{w}_r : \bar{r} \in \mathbb{H}\}$ and in $G := \langle X|R \rangle$, it holds $\bar{w}_r = 1$ if and only if $\bar{r}_r \in \mathbb{H}$: a reduction of the real Halting Problem to the Word Problem of $G$.

Notice that $G$'s relations $R$ are just semi-decidable. The construction of an algebraically presented group with BCSS-complete word problem in Section 5.5.9 is the main contribution of the present chapter.

5.5.5 Reducibility to the Real Halting Problem

We first show that, parallel to Fact 5.37(a), the word problem for any algebraically enumerated real group is not harder than the BCSS Halting Problem.

Theorem 5.50. Let $G = \langle X|R \rangle$ denote an algebraically enumerated real group. Then the associated word problem is BCSS-semi-decidable.

The proof is based on the following

Lemma 5.51. For $Y \subseteq \mathbb{R}^*$, it holds: If $Y$ is (semi-)decidable, then so is $\langle Y \rangle$.

Proof. Given a string $\bar{w} = (y_1, \ldots, y_k) \in \mathbb{R}^k$, consider all $2^k - 1$ partitions of $\bar{w}$ into non-empty subwords. For each subword, decide or semi-decide whether it belongs to $Y \cup Y^{-1}$. Accept iff, for at least one partition, all its subwords succeed.

Proof (Theorem 5.50). By Definition 5.36(b+c), $\bar{w} \equiv 1 \Leftrightarrow \bar{w} \in \langle R \rangle_n$, that is, if and only if

$$\exists_n \in \mathbb{N} \exists \bar{x}_1, \ldots, \bar{x}_n \in \langle X \rangle \exists \bar{r}_1, \ldots, \bar{r}_n \in \langle R \rangle : \quad \bar{w} = \bar{x}_1 \bar{r}_1 \bar{x}_1^{-1} \cdot \bar{x}_2 \bar{r}_2 \bar{x}_2^{-1} \cdots \bar{x}_n \bar{r}_n \bar{x}_n^{-1}. \quad (5.9)$$

Since both $X$ and $R$ were required to be semi-decidable, the same holds for $\langle X \rangle$ and $\langle R \rangle$. Since semi-decidability is equivalent to enumerability (Fact 2.17(a)), this carries over to $\langle W \rangle_n$. Indeed, let $f, g : \mathbb{R}^* \to \mathbb{R}^*$ be BCSS-computable with $\langle X \rangle = \text{range}(f)$ and $\langle R \rangle = \text{range}(g)$; then it is easy to construct (but tedious to formalize) from $f$ and $g$ a BCSS-computable function on $\mathbb{R}^*$ ranging over all $n \in \mathbb{N}$, all $\bar{w} \in \langle X \rangle$, all $\bar{x}_1, \ldots, \bar{x}_n \in \langle X \rangle$, and all $\bar{r}_1, \ldots, \bar{r}_n \in \langle R \rangle$. Compose its output with the decidable test "$\bar{w} = \bar{x}_1 \bar{r}_1 \bar{x}_1^{-1} \cdots \bar{x}_n \bar{r}_n \bar{x}_n^{-1}$?" and, if successful, return $\bar{w}$. This constitutes a function on $\mathbb{R}^*$ with range exactly $\langle W \rangle_n$.

\textsuperscript{25}This crucial direction would fail for instance for $\bar{w}_{(r_1, \ldots, r_d)} := x_{r_d}^{-1} \cdots x_{r_1}^{-1} \cdot t \cdot x_1 \cdots x_d \cdot t$. 

---

\textsuperscript{25}This crucial direction would fail for instance for $\bar{w}_{(r_1, \ldots, r_d)} := x_{r_d}^{-1} \cdots x_{r_1}^{-1} \cdot t \cdot x_1 \cdots x_d \cdot t$. 

---
5.5.6 Basics from Group Theory and Their Presentations

This subsection briefly recalls some constructions from group theory and their properties which will be used intensively later on. For a more detailed exposition as well as proofs of the cited results, we refer to the two textbooks [LySs77, Rotm95]. Our notational emphasis for each construction and claim lies on the particular group presentation under consideration—for two reasons: First and as opposed to the discrete case, different presentations of the same group may severely affect its effectivity properties (Example 5.47). Second, sometimes there does not seem to be a ‘natural’ choice for a presentation (Remark 5.53). No assumptions (e.g. effectivity) are made here concerning the set of generators or concerning the relations presenting a group. To start with and just for the record, let us briefly extend the standard notions of a subgroup and a homomorphism to the setting of presented groups: 

**Definition 5.52.** A subgroup $U$ of the presented group $G = \langle X \mid R \rangle$ is a tuple $(V, S)$ with $V \subseteq \langle X \rangle$ and $S = R \cap \langle V \rangle$. This will be denoted by $U = \langle V \mid R_V \rangle$ or, more relaxed, $U = \langle V \mid R \rangle$. A realization of a homomorphism $\psi : G \rightarrow H$ between presented groups $G = \langle X \mid R \rangle$ and $H = \langle Y \mid S \rangle$ is a mapping $\psi : X \rightarrow \langle Y \rangle$ whose unique extension to a homomorphism on $\langle X \rangle$ maps $R$-cosets to $S$-cosets, that is, makes Equation (5.8) commute.

A realization of an isomorphism $\phi$ is a realization of $\psi$ as a homomorphism.

In the above notation, $\langle \psi'(X) \mid S \rangle$ is a presentation of the subgroup $\psi(G)$ of $H$. For an embedding $\psi$, $G$ is classically isomorphic to $\psi(G)$; Lemma 5.55 below contains a computable variation of this fact.

**Remark 5.53.** The intersection $A \cap B$ of two subgroups $A, B$ of $G$ is again a subgroup of $G$. For presented subgroups $A = \langle U \mid R \rangle$ and $B = \langle V \mid R \rangle$ of $G = \langle X \mid R \rangle$, however, $\langle U \cap V \mid R \rangle$ is in general not a presentation of $A \cap B$.

Roughly speaking, a group is free if it is generic in the sense that it satisfies no relations other than the ‘basic’ Rules 5.7.

**Fact 5.54 (Nielsen).** Let $U \subseteq \langle X \rangle$.

a) Suppose $U$ is finite. Then there exists a finite $V \subseteq \langle X \rangle$ such that $\langle U \rangle_{\langle X \rangle} = \langle V \rangle_{\langle X \rangle}$ and $V$ is Nielsen reduced in the sense that it satisfies for all $u, v, w \in V \cup V^{-1}$:

(N0) $u \neq 1$.

(N1) If $uv \neq 1$, then $|uv| \geq \max\{|u|, |v|\}$.

(N2) If $uv \neq 1 \neq vw$, then $|uvw| > |u| - |v| + |w|$

where $|u|$ denotes the length of $u \in \langle X \rangle$.

b) Suppose $U$ is Nielsen-reduced. Then $u_1, \ldots, u_n \in U \cup U^{-1}$ with $u_i u_{i+1} \neq 1$ implies $|u_1 \cdots u_n| \geq n$. In particular the subgroup $\langle U \rangle_{\langle X \rangle}$ of $\langle X \rangle$ is again free and isomorphic to $\langle U \rangle$.

c) Every subgroup of $\langle X \rangle$ is free.

**Proof.** [LySs77, Section I.2].

**Definition 5.55 (Free Product).** Consider two presented groups $G = \langle X \mid R \rangle$ and $H = \langle Y \mid S \rangle$ with disjoint generators $X \cap Y = \emptyset$ — e.g. by proceeding to $X' := X \times \{1\}, Y' := Y \times \{2\}$, $R' := R \times \{1\}, S' := S \times \{2\}$. The free product of $G$ and $H$ is the presented group

$$G \ast H := \langle X \cup Y \mid R \cup S \rangle.$$

Similarly for the free product $\bigast_{i \in I} G_i$ with $G_i = \langle X_i \mid R_i \rangle$, $i$ ranging over arbitrary index set $I$. 


In many situations one wants to identify certain elements of a free product of groups. These are provided by two basic constructions: amalgamation and Higman-Neumann-Neumann (HNN for short) extension, see [HNN49] [LySs77] [Rotm95]. The intuition behind the latter is nicely illustrated, e.g., in [Rotm95, Figure 11.9].

**Definition 5.56 (Amalgamation).** Let $G = \langle X| R \rangle$, $H = \langle Y| S \rangle$ with $X \cap Y = \emptyset$. Let $A = \langle V| R \rangle$ and $B = \langle W| S \rangle$ be respective subgroups and $\phi' : \langle V \rangle \to \langle W \rangle$ the realization of an isomorphism $\phi : A \to B$. The free product of $G$ and $H$ amalgamating the subgroups $A$ and $B$ via $\phi$ is the presented group

$$\langle G \ast H \mid \phi(a) = a^i v a \in A \rangle := \langle X \cup Y \mid R \cup S \cup \{\phi'(\bar{v})\bar{v}^{-1} : \bar{v} \in V \} \rangle. \quad (5.10)$$

**Definition 5.57 (HNN Extension).** Let $G = \langle X| R \rangle$, $A = \langle V| R \rangle$, $B = \langle W| R \rangle$ subgroups of $G$, and $\phi'$ a realization of an isomorphism between $A$ and $B$. The Higman-Neumann-Neumann (HNN) extension of $G$ relative to $A, B$ and $\phi$ is the presented group

$$\langle G; t \mid ta = \phi(a)t' va \notin A \rangle := \langle X \cup \{t\} \mid R \cup \{\phi'(\bar{v})t\bar{v}^{-1}t^{-1} : \bar{v} \in V \} \rangle.$$

$G$ is the base of the HNN extension, $t \notin X$ is a new generator called the stable letter, and $A$ and $B$ are the associated subgroups of the extension.

Similarly, $(G; (t_i)_{i \in I} \mid t_ia = \phi_i(a)t_i'va \in A; \forall i \in I)$ denotes the HNN extension with respect to a family of isomorphisms $\phi_i : A_i \to B_i$ and subgroups $A_i, B_i \subseteq G$, $i \in I$.

Both HNN extensions and free products with amalgamation admit of simple and intuitive characterizations for a word to be equivalent to $1$ in the resulting group. These results are connected to some very famous names in group theory. Proofs can be found, e.g., in [LySs77, Chapter IV] or [Rotm95, Chapter 11].

**Fact 5.58 (Higman-Neumann-Neumann).** Let $G^* := \langle G; t| ta = \phi(a)t' va \in A \rangle$ be an HNN extension of $G$. Then identity $g \mapsto g$ is an embedding of $G$ in $G^*$.

**Fact 5.59 (Britton’s Lemma).** Let $G^* := \langle G; t| ta = \phi(a)t' va \in A \rangle$ be an HNN extension of $G$. Consider a sequence $(g_0, t^{-1}g_1, t^{-1}g_2, \ldots, t^{-1}g_n)$ with $n \in \mathbb{N}$, $g_i \in G$, $c_i \in \{-1, 1\}$. If it contains no consecutive subsequence $(t^{-1}g_i, t)$ with $g_i \in A$ or $(t, g_j, t^{-1})$ with $g_j \in B$, then it holds $g_0 \cdot t^{c_1} \cdot g_1 \cdots t^{c_n} \cdot g_n \neq 1$ in $G^*$.

**Fact 5.60 (Normal Form).** Let $P := \langle G \ast H| \phi(a) = a^i v A \rangle$ denote a free product with amalgamation. Consider $c_1, \ldots, c_n \in G \ast H$, $n \in \mathbb{N}$, such that

- Each $c_i$ is either in $G$ or in $H$;
- Consecutive $c_i, c_{i+1}$ come from different factors;
- If $n > 1$, then no $c_i$ is in $A$ or $B$;
- If $n = 1$, then $c_1 \neq 1$.

Then $c_1 \cdots c_n \neq 1$ in $P$.

### 5.5.7 First Effectivity Considerations

Regarding finitely generated groups, the cardinalities of the sets of generators (that is their ranks) behave additively under free products [LySs77, Corollary §IV.1.9]. Consequently, they can straightforwardly be bounded under both HNN extensions and free products with amalgamation. Similarly, for real groups, we have easy control over the dimension $N$ of the set of generators according to Definition 5.41.
Observation 5.61. For groups $G_i = \langle X_i|R_i \rangle$ with $X_i \subseteq \mathbb{R}^N$ for all $i \in I \subseteq \mathbb{R}$, the free product
\[
\bigstar_{i \in I} G_i = \left\langle \bigcup_{i \in I} (X \times \{i\}) \mid \bigcup_{i \in I} (R \times \{i\}) \right\rangle
\]
is of dimension at most $N + 1$. In the countable case $I \subseteq \mathbb{N}$, the dimension can even be made to not grow at all: by means of a bi-computable bijection $\mathbb{R} \times \mathbb{N} \rightarrow \mathbb{R}$ like $(x, n) \mapsto ([x], n) + (x - [x])$. Similar properties hold for free products with amalgamation and for HNN extensions.

Moreover, free products, HNN extensions, and amalgamations of algebraically generated/enumerated/presented groups are, under reasonable presumptions, again algebraically generated/enumerated/presented:

Lemma 5.62.  
\begin{enumerate}
\item[a)] Let $G_i = \langle X_i|R_i \rangle$ for all $i \in I \subseteq \mathbb{N}$. If $I$ is finite and each $G_i$ algebraically generated/enumerated/presented, then so is $\bigstar_{i \in I} G_i$.
\item[b)] Let $G = \langle X|R \rangle$ and consider the HNN extension $G^* := \langle G; (t_i)_{i \in I} \mid t_i a = \phi_i(a)t_i \forall a \in A_i \forall i \in I \rangle$ with respect to a family of isomorphisms $\phi_i : A_i \rightarrow B_i$ between subgroups $A_i = \langle V_i(R) \rangle, B_i = \langle W_i(R) \rangle$ for $V_i, W_i \subseteq \langle X \rangle$, $i \in I$.
\end{enumerate}

Suppose $I$ is finite, each $G_i$ is algebraically enumerated/presented, $V_i \subseteq \mathbb{R}^*$ is semi-/decidable, and finally each $\phi_i$ is effective as a homomorphism; then $G^*$ is algebraically enumerated/presented as well.

The same applies to $I = \mathbb{N}$, provided that the $V_i$ are uniformly semi-/decidable and effectivity of the $\phi_i$ holds uniformly.

\begin{enumerate}
\item[c)] Let $G = \langle X|R \rangle$ and $H = \langle Y|S \rangle$; let $A = \langle V|R \rangle \subseteq G$ and $B = \langle W|S \rangle \subseteq H$ be subgroups with $V \subseteq \langle X \rangle$, $W \subseteq \langle Y \rangle$, $V \subseteq \mathbb{R}^*$ semi-/decidable, and $\phi : A \rightarrow B$ an isomorphism and effective homomorphism. Then their free product with amalgamation $(5.10)$ is algebraically enumerated/presented whenever $G$ and $H$ are.
\end{enumerate}

Remark 5.63. Uniform (semi-)decidability of a family $V_i \subseteq \mathbb{R}^*$ of course means that every $V_i$ is (semi-)decidable not only by a BCSS-machine $M_i$, but all $V_i$ by one common machine $M$; similarly for uniform computability of a family of mappings. By virtue of (the proof of) [Cuck92 Theorem 2.4], a both necessary and sufficient condition for such uniformity is that the real constants employed by the $M_i$ can be chosen to all belong to one common finite field extension $\mathbb{Q}(c_1, \ldots, c_k)$ over the rationals.

Recall (Observation 5.40) that a homomorphism between finitely generated groups is automatically effective and, if injective, has decidable range and effective inverse. For real groups however, in order to make sense of the prerequisites in Lemma 5.62[a–c], we explicitly have to specify the following.

Definition 5.64. An homomorphism $\psi : \langle X|R \rangle \rightarrow \langle Y|S \rangle$ of presented real groups is called an effective homomorphism if it admits of a BCSS-computable realization $\psi' : X \rightarrow Y$ in the sense of Definition 5.52.

For $\psi$ to be called an effective embedding, it must not only be an effective homomorphism and injective; but $\psi'$ is also required to be injective and have decidable image $\psi'(X)$ plus a BCSS-computable inverse $\chi' : \psi'(X) \subseteq \langle Y \rangle \rightarrow X$.

Effective embeddings arise in Lemmas 5.65 and 5.68[f–i]. For an injective effective homomorphism $\phi$ as in Lemma 5.65, on the other hand, a realization need not be injective; for instance, $\phi'$ might map two equivalent (w.r.t. the relations $R$) yet distinct elements to the same image word.

Proof (Lemma 5.62).  
\begin{enumerate}
\item[a)] If $X_i$ is decidable for each $i \in I$, $I$ finite, then so is $\bigcup_{i \in I} (X_i \times \{i\})$; similarly for semi-decidable/decidable $R_i$. Uniform (semi-)decidability of each $X_i$ means exactly that $\bigcup_{i \in N} (X_i \times \{i\})$ is (semi-)decidable.
b) The set of generators of the HNN extension is decidable as in a). The additional relations \( \phi_i'(v) = v^{-1}t_i^{-1} : v \in V_i \) are semi-/decidable since, due to the presumption, \( V_i \) is and \( \phi_i' : \langle V_i \rangle \to \langle W_i \rangle \) is computable. Uniformity enters as in a).

c) Similarly.

**Lemma 5.65.** Let \( \psi : G = \langle X | R \rangle \to \langle Y | S \rangle = K \) denote an effective embedding.

a) There is an effective embedding \( \chi : \psi(G) \to G \) (i.e. we have an effective isomorphism).

b) If \( V \subseteq \langle X \rangle \) is decidable, then the restriction \( \psi|_H \) to \( H = \langle V | R \rangle \subseteq G \) is again an effective embedding.

c) If \( G \) is algebraically generated and \( K \) algebraically presented then \( \psi(G) \) is algebraically presented as well.

**Proof.** a) Let \( \psi' : X \to \langle Y \rangle \) denote the effective realization of \( \psi \) with inverse \( \chi' \) according to Definition 5.64. The unique extension of \( \psi' \) to a homomorphism has image \( \psi'(\langle X \rangle) = \langle \psi'(X) \rangle \). In a similar way as with Lemma 5.51 we can decide, given \( \bar{w} \in \langle Y \rangle \), whether \( \bar{w} \in \psi'(\langle X \rangle) \). Moreover, if so, we obtain a partition \( \bar{w} = (\bar{v}_1, \ldots, \bar{v}_t) \) with \( \bar{v}_i \in \psi'(X) \). Then calculating \( x_i := \chi'([\bar{v}_i]) \in X \) yields a computable extension of \( \chi' \) to a homomorphism on \( \psi'(\langle X \rangle) \) which satisfies injectivity, has decidable image and \( \psi' \) as inverse. Moreover \( \chi' \) maps \( S \)-cosets to \( R \)-cosets: Take \( \bar{v}_1, \bar{v}_2 \in \psi'(\langle X \rangle) \) with \( \bar{v}_1/S = \bar{v}_2/S \); then \( \bar{u}_i := \chi'(\bar{v}_i) \) have \( \bar{u}_i = \psi'(\bar{u}_i) \) and thus, since \( \psi' \) makes Equation (5.8) commute according to the presumption, \( \bar{v}_1/S = \psi(\bar{u}_1/R) = \psi(\bar{u}_2/R) = \bar{v}_2/S \); now injectivity of \( \psi \) implies \( \bar{u}_1/R = \bar{u}_2/R \).

b) The range \( \psi'(V) \) of the restriction \( \psi'|_V \) coincides with \( \chi'^{-1}(V) \cap \langle \psi'(X) \rangle \). The first term is decidable since \( \chi' \) is computable and \( V \) decidable; the second term is decidable by Definition 5.64 and Lemma 5.51.

c) becomes clear by staring at \( \psi(G) = \langle \psi'(X) | S \rangle \).

### 5.5.8 Benign Embeddings

The requirement in Lemma 5.62\(+c\) that the subgroup(s) \( A \) be recursively enumerable or even decidable, is of course central but unfortunately violated in many cases. For instance, a subgroup of a finitely presented group in general need not even be finitely generated: Consider for instance the commutator \( [G,G] := \langle \{ uvu^{-1}v^{-1} : u, v \in G \} \rangle \) of the free group \( G = \langle \{a, b\} \rangle \) and compare the Remark on p.177 of [LySs77]. Similarly, the algebraically presented real group \( (\mathbb{R}, +) \) has a subgroup (Example 5.47) which is not algebraically generated. Nevertheless, both can obviously be effectively embedded into \( a \), respectively, finitely presented and an algebraically presented group. This suggests the notion of **benign** subgroups, in the classical case (below, Item a) introduced in [Higm61]. Recall that there, the effectiveness of an embedding drops off automatically.

**Definition 5.66.** a) Let \( X \) be finite, \( V \subseteq \langle X \rangle \). The subgroup \( A = \langle V | R \rangle \) of \( G = \langle X | R \rangle \) is (classically) benign in \( G \) if the HNN extension \( \langle X : t \mid ta = at/\forall a \in A \rangle \) can be embedded in some finitely presented group \( K = \langle Y | S \rangle \).

b) Let \( X \subseteq \mathbb{R}^* \), \( V \subseteq \langle X \rangle \). The subgroup \( A = \langle V | R \rangle \) of \( G = \langle X | R \rangle \) is effectively benign in \( G \) if the HNN extension \( \langle G : t \mid ta = at/\forall a \in A \rangle \) admits of effective embedding in some algebraically presented group \( K = \langle Y | S \rangle \).

c) Let \( I \subseteq \mathbb{N} \). A family \( \{ A_i \}_{i \in I} \) of subgroups of \( G \) is uniformly effectively benign in \( G \) if, in the sense of Remark 5.63, there are groups \( K_i \) uniformly algebraically presented and uniformly effective embeddings \( \phi_i : \langle G : t \mid ta = at/\forall a \in A_i \rangle \to K_i \).

The benefit of benignity is revealed in the following
Remark 5.67. In the notation of Definition 5.66, if \( A \) is effectively benign in \( G \) then the word problem for \( A \) is reducible to that for \( K \). \( \Box \\
\)
Moreover in this case, the membership problem for \( A \) in \( G \)—that is the question of whether a given \( \bar{x} \in \langle X \rangle \) is equivalent (w.r.t. \( R \)) to an element of \( A \)—is also reducible to the word problem for \( K \). According to Fact 5.59, \( a := \bar{x}/R \) satisfies \( t \cdot a \cdot t^{-1} \cdot a^{-1} = 1 \Leftrightarrow a \in A \). \( \Box \\
\)

We now collect some fundamental properties that will be used frequently later on. They extend corresponding results from the finite framework. Specifically, Lemma 5.68b) generalizes [LySs77, Lemma §IV.7.7(i)] and Claims d+e) generalize [LySs77, Lemma §IV.7.7(ii)].

**Lemma 5.68.**

a) Let \( A = \langle V|R \rangle \subseteq H = \langle W|R \rangle \subseteq G = \langle X|R \rangle \) denote a chain of (sub-) groups with \( V \subseteq \langle W \rangle \) and \( W \subseteq \langle X \rangle \). If \( W \) is decidable and \( A \) effectively benign in \( G \), then it is also effectively benign in \( H \).

b) If \( G = \langle X|R \rangle \) is algebraically presented and subgroup \( A = \langle V|R \rangle \) has decidable generators \( V \subseteq \langle X \rangle \), then \( A \) is effectively benign in \( G \).

c) If \( A \) is effectively benign in \( G \) and \( \phi : G \rightarrow H \) an effective embedding, then \( \phi(A) \) is effectively benign in \( \phi(G) \).

d) Let \( A \) and \( B \) be effectively benign in algebraically presented \( G \). Then \( A \cap B \) admits of a presentation effectively benign in \( G \).

e) Let \( A, B, G \) as in d); then \( \langle A \cup B \rangle_G \) admits of a presentation\(^{26}\) effectively benign in \( G \).

f) Let \( \langle A_i \rangle_{i \in I} \) be uniformly effectively benign in \( G \) (Definition 5.66c). Then \( \langle \bigcup_{i \in I} A_i \rangle \) admits of a presentation effectively benign in \( G \).

The above claims hold uniformly in that the effective embeddings do not introduce new real constants.

**Proof.**

a) Let \( \psi \) be an effectively realizable embedding of the HNN extension \( \langle X;t \mid t \cdot a = \phi(a) \cdot t \cdot v \cdot a \in A \rangle \) in some algebraically presented \( K = \langle Y|S \rangle \). Since \( W \cup \{ t \} \) is decidable, Lemma 5.65b) ensures the restriction of \( \psi \) to yield an effective embedding of the HNN extension \( \langle W;t \mid t \cdot a = \phi(a) \cdot t \cdot v \cdot a \in A \rangle \) in \( K \).

b) The identity being an effectively realizable embedding \( X \) is decidable, now apply Lemma 5.65b), it suffices to observe that the HNN extension

\[
K := \langle G;t \mid at = ta \cdot v \cdot a \in A \rangle = \langle X;t \mid R \cup \{ \bar{v}t = t \bar{v} \bar{v} \in V \} \rangle
\]

is algebraically presented itself. Indeed, \( X, R, \) and the additional relations parametrized by \( V \) are decidable by presumption.

c) The presented HNN extension under consideration,

\[
\langle \phi'(X); s \mid \phi'(\bar{v})s = s\phi'(\bar{v})\forall \bar{v} \in V \rangle,
\]

is the image under \( \phi \) of \( \langle G;t \mid at = ta \cdot v \cdot a \in A \rangle \) by extending \( \phi'(t) := s \). Due to the presumption, the latter HNN extension embeds in some (finite-dim.) algebraically presented \( K \) via some effective \( \psi \). According to Lemma 5.65c), \( \phi \) admits an effective inverse. Hence the composition \( \psi \circ \phi^{-1} \) constitutes the desired effective embedding of (5.11) in \( K \).

d) By assumption there exist two algebraically presented groups \( K = \langle Y|S \rangle \) and \( L = \langle Z|T \rangle \) together with realizations \( \phi' : X \cup \{ r \} \rightarrow \langle Y \rangle, \psi' : X \cup \{ r \} \rightarrow \langle Z \rangle \) of effective embeddings

\[
\phi : G_A := \langle G;r \mid ar = ra \cdot v \cdot a \in A \rangle = \langle X;r \mid R \cup \{ \bar{v}r = r \bar{v} \cdot \bar{v} \in V \} \rangle \rightarrow K = \langle Y|S \rangle
\]

\[
\psi : G_B := \langle G;r \mid br = rb \cdot v \cdot b \in B \rangle = \langle X;r \mid R \cup \{ \bar{w}r = r \bar{w} \cdot \bar{w} \in W \} \rangle \rightarrow L = \langle Z|T \rangle.
\]

---

\(^{26}\)Possibly different from \( (V \cup W)|R) \)
We shall realize an embedding $\chi$ of the HNN extension $G_C := \langle G; r\mid cr = re\forall c \in C \rangle$ in an algebraically presented group for the presentation\footnote{Notice the arbitrarily broken symmetry between the groups/embeddings $(A, \phi)$ and $(B, \psi)$ involved.} $C := \langle \{\bar{w} \in (W) : \bar{w}/R \in A\} \mid R \rangle$ for $A \cap B$. Observe that $\phi(G) = \langle \phi'(X) \rangle$ and $\psi(G) = \langle \psi'(X) \rangle$ are subgroups of $K$ and $L$, respectively, and isomorphic due to Fact 5.58 with isomorphism $\phi \circ \psi^{-1} : \psi(G) \rightarrow \phi(G)$ realized by $\phi' \circ \psi^{-1}$ according to Lemma 5.65. Definition 5.56 is thus applicable and we are entitled to consider the free group with amalgamation

\[
P := \langle K \ast L \mid \phi(\psi^{-1}(t)) = \ell \forall t \in \psi(G) \rangle = \langle Y \cup Z \mid S \cup T \cup \{\phi'(\psi'^{-1}(z)) = z : z \in \psi'(X)\} \rangle.
\]

$P$ is algebraically presented because of Lemma 5.62c). Moreover, $\phi(G) = \psi(G)$ in $P$ according to (5.12). Also, $s := \phi'(r)$ commutes exactly with $\phi(A)$ and $t := \psi'(r)$ exactly with $\psi(B)$, so $s \cdot t$ commutes exactly with $\phi(A) \cap \psi(B)$. Therefore, $\chi' : X \cup \{r\} \rightarrow \langle Y \cup Z, x \mapsto \psi'(x), r \mapsto s \cdot t \rangle$ respects cosets in the sense of Equation (5.8) and thus realizes an embedding $\chi : \langle G; r\mid cr = re\forall c \in C \rangle \rightarrow P$ as desired.

e) With notations as in d), it holds

\[
\psi(\langle A \cup B \rangle_G) = \phi(\langle A \cup B \rangle_G) = \langle \phi(A) \cup \phi(B) \rangle_p = \langle \phi(A) \cup \psi(B) \rangle_p = \langle \phi(r \cdot G \cdot r^{-1}) \cup \psi(r \cdot G \cdot r^{-1}) \rangle_p \cap \phi(G);
\]

the first line because $\phi$ and $\psi$ are injective homomorphisms coinciding on $G$; the second because $A$ and only $A$ commutes with $r$ in $G_A$ due to Britton’s Lemma (Fact 5.59), similarly for $B$ in $G_B$. Now $\phi(G)$ is algebraically presented due to Lemma 5.65 and thus effectively benign in $P$ by Claim b). Similarly, $\langle \phi(r \cdot G \cdot r^{-1}) \cup \psi(r \cdot G \cdot r^{-1}) \rangle_p$ has decidable generators and is thus effectively benign in $P$ as well. Claim d) now ensures the effective benignity of $\phi(\langle A \cup B \rangle)$ in $P$; and therefore also in $\phi(G) \subseteq P$ according to Claim a) combined with Lemma 5.65c). Claim c) combined with Lemma 5.65d) finally yields the effective benignity of $\langle A \cup B \rangle$ in $G$.

f) Let $(\phi_i)_{i \in I}$ denote the uniformly computable realizations of embeddings $\phi_i : G_i := \langle G; r\mid ar = ra\forall a \in A_i \rangle \rightarrow K_i$. Fix $j \in I$. Similar to Equation (5.12) and the proof of e), we have

\[
\phi_j(\langle \bigstar_{i \in I} A_i \rangle_G) = \langle \bigcup_{i \in I} \phi_i(r \cdot G \cdot r^{-1}) \rangle_p \cap \phi_j(G),
\]

\[
P := \langle \bigstar_{i \in I} K_i \mid \phi_i(\phi_j^{-1}(t)) = t \forall t \in \phi_j(G) \forall i \in I \rangle
\]

where (by uniformity, see Lemma 5.62b) $P$ and $\phi_j(G)$ are algebraically presented, and $\langle \bigcup_{i \in I} \phi_i(r \cdot G \cdot r^{-1}) \rangle_p$ has decidable generators of bounded dimension, compare Observation 5.61.

### 5.5.9 Reduction from the Real Halting Problem

With the preparations and tools from the previous sections, it is now easy to proceed to the main result of this chapter: the construction of an algebraically presented group to whose word problem the real Halting Problem can be reduced to.

Let $\mathbb{H} \subseteq \mathbb{R}^*$ denote the real Halting Problem, semi-decided by some (constant-free) universal BCSS Machine $\mathbb{M}$. Denote by $n \mapsto \gamma_n$ an effective enumeration of all computational paths of $\mathbb{M}$, $A_n \subseteq \mathbb{H} \cap I^{d(n)}$ the set of inputs accepted at path $\gamma_n$.

**Definition 5.69.** Let $X := \{x_r : r \in \mathbb{R}\} \cup \{k_n : n \in \mathbb{N}\} \cup \{s\}$, $G := \langle X \rangle$, and

\[
U := \langle k_n^{-1} \cdot w_r \cdot k_n : n \in \mathbb{N}, r' \in A_n \rangle \quad \text{where} \quad w_{r_1, \ldots, r_d} := x_{r_d}^{-1} \cdot \cdots \cdot x_{r_1}^{-1} \cdot s \cdot x_{r_1} \cdots x_{r_d}.
\]

Finally let $V := \langle U; (k_n) \rangle \cap \langle s; x_r : r \in \mathbb{R} \rangle$. 


Lemma 5.70.  a) $U$ is decidable.

b) $U$ and $V$ are effectively benign in $G$.

c) The words $k_n^{-1}w_r k_n$ are Nielsen-reduced (and thus freely generate $U$ by virtue of Fact 5.54b).

d) It holds $V = \langle \bar{w}_r : \exists n \in \mathbb{N} : \bar{r} \in A_n \rangle$.

Proof.  a) Observe that $\bigcup_{n \in \mathbb{N}} (\{ n \} \times A_n) \subseteq \mathbb{R}^+$ is decidable: From $k_n$, compute the description $\gamma_n$ of the computational path of $M$ corresponding to (and thus a semi-algebraic description of) $A_n$. Knowing $n$, it is simply a matter of evaluating the path (i.e., the system of polynomial inequalities) in order to decide membership of a given $\bar{r}$ to $A_n$.

b) Apply Lemma 5.68b+d).

c) Compare [LySs77, p.223].

d) For $n \in \mathbb{N}$ and $\bar{r} \in A_n$, $\bar{w}_r = k_n \cdot (k_n^{-1} \cdot \bar{w}_r \cdot k_n) \cdot k_n^{-1}$ belongs to $\langle U ; (k_n) \rangle$ and to $\langle s ; x_r : r \in \mathbb{R} \rangle$; hence the inclusion “$V \supseteq \ldots$” follows. Conversely by c), a word in $\langle U ; (k_n) \rangle$ devoid of $k_n$ can arise only from $k_n^{-1} \cdot \bar{w}_r \cdot k_n \in \langle U \rangle$ after elimination of $k_n^{-1}$ and $k_n$; thus “$V \subseteq \ldots$” holds as well.

Corollary 5.71.  Let $K$ denote the algebraically presented group which the HNN extension $\langle G ; t | tv = vt \forall v \in V \rangle$ effectively embeds in (Lemma 5.70b) by some effective embedding $\psi$. Then, according to Fact 5.59 and Lemma 5.70d), $\bar{v} := \psi(t \cdot \bar{w}_r \cdot t^{-1} \cdot \bar{w}_r^{-1})$ equals 1 in $K$ iff $\bar{r} \in H = \bigcup_n A_n$.

The simplicity of this construction of an SAP group with BSS-complete word problem may not be as surprising when viewed in the discrete setting, yielding a decidable yet infinitely presented group with Turing-complete word problem:

Example 5.72.  Let $H|_n := \{ e \in \mathbb{N} : \text{TM no.$e$ terminates after at most n steps} \}$. Let $G := \langle x, y, s \rangle$, $U := \langle y^n \cdot x^c \cdot s \cdot x^{-c} \cdot y^{-n} ; n \in \mathbb{N}, e \in H|_n \rangle$, $V := \langle U ; y \rangle \cap \langle x, s \rangle$. Then the HNN extension of $G$ relative to $V$ is embeddable in a group $K = \langle Y|S \rangle$ with Turing-decidable (although not necessarily finite) $Y, S$. 


Chapter 6

Concluding Comparison

Real Hypercomputation combines real computability with (discrete) hypercomputation in order to study models of real number computation beyond the Church–Turing Hypothesis. A particular focus lies on (extensions of) the two major relevant notions: Recursive Analysis (Chapter 4) and BCSS machines (Chapter 5).

Discrete hypercomputers can usually be characterized in terms of undecidable oracles; their iteration gives rise to the Arithmetical Hierarchy of subsets of integers (Section 3.1.2). Oracle-access extends real models, too, and gives rise to a taxonomy of certain subsets of reals according to their degree of undecidability: the Arithmetical Hierarchy for BCSS machines (Section 5.4) and, in the case of Recursive Analysis, the effective Borel Hierarchy (Section 4.1). Here, one also has an Arithmetical Hierarchy of single reals (Section 4.2).

6.1 Discrete versus Real Hypercomputation

In the discrete realm, every integer function becomes computable relative to some appropriate oracle. This is different in the real case: a discontinuous function as simple as the sign remains uncomputable in Recursive Analysis relative to any oracle (Corollary 4.21); and so does square root and exponentiation in the BCSS model (Theorem 5.11).

This leads to a second difference between the theories of discrete and of real hypercomputers: the latter naturally gives rise to non-oracle extensions, as for instance

- in the BCSS case: enhancing the set of operational primitives $+, -, \times, \div, <$ (Section 5.1);
- for Recursive Analysis: weakened notions of function evaluation (Section 4.3.1) or, equivalently, effective measurability rather than effective continuity (Section 4.3.4).

Another non-oracle extension brings us to the third major difference between real and discrete hypercomputation: for the latter, nondeterminism can be (reasonably required to be bounded and thus) simulated deterministically. Over the reals, on the other hand, nondeterminism is (inherently unbounded and) of surprising computational power strictly exceeding the (finite) Arithmetical Hierarchy; recall Sections 4.5 and 5.4.2.

Finally the fourth difference: Not surprising in view of its origin (Section 1.1), arguments in discrete Computability Theory can typically be described as 'logical' (classic example: diagonalization) and combinatorial. On the other hand, proofs in computability, and even more in hypercomputation, on reals (have to) additionally take into account topological and algebraic properties of $\mathbb{R}$ (recall Section 2.2), leading to a much more diverse and rich choice of methods and tools—which brings us back to the fifth difference:

Establishing a discrete problem $P$ as undecidable generally proceeds by reduction from the Halting problem $H$ whose undecidability in turn follows by diagonalization. In particular, such a proof implies $P$ to at least as hard as $H$; and it requires considerable effort to construct a (say,
semi-decidable yet) undecidable problem strictly easier than $H$ (Section 3.1.4). Over the reals, on the other hand, the methods and tools available in addition to diagonalization show problems undecidable without implying reducibility from the Halting problem (Section 5.3); and indeed in the BCSS setting it becomes a nontrivial task to devise a ‘natural’ problem equivalent to $\mathbb{H}$.

### 6.2 Superrecursive Analysis versus BCSS Hypercomputers

The two major models of real computability focus on complementary aspects of practical computation on $\mathbb{R}$ (Section 2.8) and, thus not surprisingly, are incomparable; recall Observation 2.64. One might think that Part b) of the latter can be removed by turning to BCSS hypercomputers in the sense of Section 5.1; however, this is not the case because it holds

**Fact 6.1.** To every finite (or even countably infinite but uniform) family of real operations $(f_n)$, computable in the sense of Recursive Analysis, there exists a $(\rho \to \rho)$-computable real function which is uncomputable to a BCSS machine over $(\mathbb{R}, +, -, \times, \div, <, f_1, \ldots, f_n)$.

**Proof.** See [Weih00, Exercise 9.7.4].

On the other hand, super-Turing notions of Recursive Analysis and BCSS computability do exhibit surprising similarities. Two of them are mentioned above:

- the natural appearance of non-oracle extensions to BCSS machines as well as in Recursive Analysis;
- the surprising power of nondeterminism in both models.

Also, the effects of access to (iterations of) Halting problems bear strong similarities, namely leading to Arithmetical Hierarchies of real subsets (Sections 5.4 and 4.1), single real numbers (Section 4.2), and real functions (Section 4.3.2).

#### 6.2.1 Comparing Arithmetical Hierarchies

As a matter of fact, the Arithmetical Hierarchies of Euclidean subsets induced by BCSS machines and by Recursive Analysis do permit a nice way of comparing the (semi-decision, as opposed to function computation) power of both models:

a) The first simply ‘interleaves’ the classical Borel Hierarchy: $\Sigma_d \subseteq \Pi_d \subseteq \Sigma_{d+1}$ (Corollary 5.32). In particular, the (finite) hierarchies themselves coincide: $\bigcup_d \Sigma_d = \bigcup_d \Pi_d$.

b1) The latter gives rise to the effective refinement $\Sigma_{d-1} \not\subseteq \Sigma_d \not\subseteq \Sigma_d$ of the classical Borel Hierarchy. In particular, the effective hierarchy $\bigcup_d \Sigma_d$ is a strict subclass of $\bigcup_d \Sigma_d$ of merely countable magnitude.

b2) On the other hand, every $S \in \Sigma_d$ belongs, for some appropriate oracle $O$, to the relativized class $\Sigma_d[O]$.

The rest of this work proceeds by exploiting hypercomputation as a means to compare BCSS machines and Recursive Analysis: Section 6.3 gives a more detailed account of which oracle $O$ is needed in Item b2) above, depending on which BCSS machine is employed in a); whereas Section 6.4 reveals both BCSS model and Recursive Analysis to be included in a certain class of Analytic Machines which (Section 6.4.2) can in turn be simulated by Type-2 nondeterminism.
6.2.2 Effect of Non-/Uniformity

A Turing machine’s state control (‘program’) must be finite (Observation 3.14a); it is thus in general impossible to combine an arbitrary countable collection of Turing machines \((M_n)\) into a single one (i.e. for the purpose of dove-tailed simulating): This requires (and conversely follows from) the sequence \((M_n)\) to be uniform. Correspondingly, in Recursive Analysis there exist sequences \((a_n)\) which are uncomputable although every single member \(a_n\) is computable.

To a BCSS machine, on the other hand, any collection \((M_n)\) or \((a_n)\) can be encoded and stored as a single real number for later retrieval (Example 2.25): The issue of non-uniformity does not apply to discrete information here. Instead, a sequence \((M_n)\) of BCSS machines may in general be impossible to combine into a single one for a different reason: because they contain and require an infinite number of real constants to store.

Dropping the uniformity condition from a Turing machine (e.g. by permitting it an arbitrary infinite program) results in ultimate computational power: it can now decide any discrete language (and, as you may recall, this motivated Definition 3.18iv). A BCSS machine with an arbitrary countable number of real constants, on the other hand, while strictly more powerful than a standard one (Example 2.29), is still restricted to operating in (the real closure of) a countably generated field extension of \(\mathbb{Q}\), i.e. a very small part of \(\mathbb{R}\) only (Fact 2.5).

Question 6.2. How powerful are BCSS hypercomputers with countably many real constants?

6.3 BCSS to Hyperrecursive Analysis: Equality is a Jump

BCSS machines can easily compute discontinuous functions like the simple sign one—which in Recursive Analysis is not \((\rho \to \rho)\)-computable, even relative to any oracle. Restricting our considerations to (semi-) decidability, again the BCSS model’s power easily reaches topologically as high as Borel class \(\Sigma_2\); whereas Recursive Analysis remains within \(\Sigma_1\) (Fact 2.49), again even relative to any oracle. Thus the explicit restriction to open BCSS (semi-)decidable subsets of \(\mathbb{R}^k\) is inevitable for a meaningful comparison. Finally, we had better require the BCSS machine \(M\) under consideration to use only computable real constants \(c\), otherwise BCSS-decidable open set \(\mathbb{R} \setminus \{c\}\) becomes arbitrarily intractable to Recursive Analysis.

6.3.1 Open BCSS Semi-Decidable Subsets

Respecting these (thus necessary) hypotheses, one arrives at

**Question 6.3.** Is every Euclidean open set \(U \subseteq \mathbb{R}^k\), if semi-decidable by a BCSS-machine \(M\) with computable real constants only, r.e. open (in the sense of Section 2.4.4)?

A Type-2 Machine can enumerate all computational paths of \(M\) and for each leaf \(v\) exhaust the interior of \(X_v\), its associated set \(X_v\) of inputs \(\vec{x}\) arriving at \(v\) (recall Section 2.3.2), by open rational balls: this yields r.e. enumerability of \(U\) in case that \(M\) does not use equality; compare [Zho98, Theorem 3.2]. In general, one similarly sees r.e. enumerability of \(\bigcup_v X_v\) which, however, may be a proper subset of \(U\): for instance, in case that \(M\) distinguishes (but, since we require the domain to be open, accepts all) the three outcomes of sign tests: negative, positive, and zero. It is therefore this capability of recognizing equality which makes general BCSS computations not only often numerically unstable in practice but also theoretically hard to compare to Recursive Analysis.

As a matter of fact, Boldi and Vigna have answered Question 6.3 in the negative: they constructed in [BoVi99, Section 6] an open Euclidean subset \(U\), even decidable by a BCSS machine without constants, which is not r.e. open.

**Example 6.4.** There exists an open Euclidean subset \(U\), even decidable by a BCSS machine without constants, which is not r.e. open.
On the other hand, the answer to Question 6.3 is in the affirmative when oracle-access is additionally granted to the (classical and non-iterated) Halting problem—provided that $U$ is BCSS-decidable. Slightly more generally, relativized to the real constants employed by $M$, \cite[Sect. 5]{BoVi99} establishes

**Fact 6.5.** Suppose that $U \subseteq \mathbb{R}^k$ is open and its complement semi-decidable by a BCSS-machine with real constants $c_1, \ldots, c_D$. Let $O = (c_1, \ldots, c_D)$ denote a joint discrete (e.g. binary) encoding of these constants. Then $U$ is r.e. open relative to $O'$.

It follows from the following (Type-1) semi-decidability result:

**Lemma 6.6.** Fix $c_1, \ldots, c_D \in \mathbb{R}$. There exists a Turing machine $M^O$ with oracle access to $O = (c_1, \ldots, c_D)$ which, on input of (a discrete encoding of the control part of) a BCSS machine $M$ with constants $c_1, \ldots, c_D$, terminates iff $\text{dom}(M) \neq \emptyset$.

The proof of the latter is a real (!) joy to read as it constitutes a particularly nice illustration of how real hypercomputation differs from discrete (Section 6.1): it employs, in addition to recursion theoretic degrees of real numbers\footnote{Miller pointed out \cite{Mil04} that continuous real functions need not have Turing degrees.} \cite[Section 4]{BoVi99}, the following combination of tools from topology and algebra:

**Claim 6.7.** Let $c_1, \ldots, c_d \in \mathbb{R}$ be algebraically independent and $c_{d+1}, \ldots, c_D$ algebraic over $F := \mathbb{Q}(c_1, \ldots, c_d)$. Let $E \subseteq \mathbb{R}$ denote the smallest real closed field containing $F$ and $O := (c_1, \ldots, c_D)$.

a) Suppose $S \subseteq \mathbb{R}^k$ is semi-algebraic over $E$; then $E \cap S$ is dense in $S$.

b) A Turing machine can decide the equality $y = 0$? of an element $y \in E$ given in the form of (the coefficients of) a quoly-polynomial $p \in \mathbb{Q}[X_1, \ldots, X_d][Y]$ and rational numbers $a, b$ such that $y$ is the unique zero of $p(c_1, \ldots, c_d) \in F[Y]$ within the open interval $(a,b)$.

c) Moreover, arithmetic operations like $(y, z) \mapsto y+z$ in $E$ can be performed within the encoding from b) using Gröbner Base techniques; refer to \cite[Section 2]{BoVi99}.

d) In particular, an oracle Turing machine $M^O$ can simulate a given BCSS machine $M$ with constants $c_1, \ldots, c_D$ on any input $\vec{y} \in \mathbb{R}^k$ given as in b): Tests for inequality $f_u(\vec{y}) > 0$ performed by $M$ are semi-decidable (Observation 2.48) because $f_u \in \mathbb{Q}(c_1, \ldots, c_D; \vec{Y})$ is $(\rho^k \to \rho)$-computable relative to $O$; and equality $f_u(\vec{y}) = 0$ can even be decided according to $b+c$.

e) The set (of encodings as in b) of elements $\vec{y} \in \mathbb{R}^k$ is recursively enumerable relative to $O$.

We acknowledge that (the proofs of) Claims b+c+d resemble and motivated (those of) Theorem 2.55.

### 6.3.2 Continuous BCSS-Computable Functions

We now extend Fact 6.5 from the semi-decidability of Euclidean subsets to real function computability. The former requires restriction to open sets, the latter to continuous functions.

**Theorem 6.8.** Let $f : \subseteq \mathbb{R}^k \to \mathbb{R}$ be continuous and computable by a BCSS machine $M$ with real constants $c_1, \ldots, c_D$. Then $f$ is $(\rho^k \to \rho)$-computable relative to $(c_1, \ldots, c_D)'$.

Recall that Definition 2.42 does not require $(\rho^k \to \rho)$-computation of $f$ to diverge on inputs $\vec{x} \notin \text{dom}(f)$; so while $\text{dom}(f)$ under the above hypothesis automatically satisfies BCSS semi-decidability, additional presumptions about that of its complement (like in Fact 6.5) are dispensable here.
6.4. ROBUST QUASI-STRONGLY $\delta$–Q–ANALYTIC FUNCTIONS

Proof. A Turing machine can easily unroll the computations of $\mathbb{M}$ into a tree as described in Section 2.3.2 and enumerate its leaves $v$ together with symbolic descriptions (in the form of systems of polynomial (in)equalities over $F := \mathbb{Q}(c_1, \ldots, c_D)$) of the corresponding sets $X_v$ and the polynomials $g_v$ computed therein. Given $(\vec{q}_n)_n$ with $|\vec{x} - \vec{q}_n| \leq 2^{-n}$ and give $m \in \mathbb{N}$, we want to find some $n \in \mathbb{N}$ and (to output) some $y \in \mathbb{Q}$ such that it holds:

$$\text{for all leaves } v \text{ and for all } \vec{z} \in X_v \cap B(\vec{q}_n, 2^{-n}) \text{ algebraic over } F: \quad g_v(\vec{z}) \in B(y, 2^{-m}). \quad (6.1)$$

Now oracle access to $O := \langle c_1, \ldots, c_D \rangle$ permits enumeration, in the sense of Claim 6.7(b), of all $\vec{z} \in E^k \cap X_v \cap B(\vec{q}_n, 2^{-n})$ and $\rho$-evaluation of $g_v$ on them. Therefore, “$g_v(\vec{z}) \in B(y, 2^{-m})$” is co-r.e. in $\vec{z}$ relative to $O$ (a minor generalization of Observation 2.48; see for instance [16, Lemma 4.1c]); and $\Pi_1[O]$ satisfies invariance under (relatively) effective universal quantification with respect to $v$ and to $\vec{z}$, compare Lemma 3.11(b). Therefore, relative to the jump $O'$, a Turing machine can decide (6.1) and effectively search for $n$ and $y$ satisfying this equation.

By continuity of $f$ and density of $E^k \cap S := X_v \cap B(\vec{q}_n, 2^{-n})$ (Claim 6.7a), any such $(n, y)$ found will satisfy $g_v(\vec{z}) \in B(y, 2^{-m})$ for all $\vec{z} \in B(\vec{q}_n, 2^{-n})$ and thus be a valid $2^{-m}$-approximation to $f(\vec{x})$. Conversely, since $f$ is continuous, there exists for any $\vec{x} \in \text{dom}(f)$ and $m \in \mathbb{N}$ some $n \in \mathbb{N}$ and $y \in \mathbb{Q}$ such that, for all $\vec{z} \in B(\vec{q}_n, 2^{-n}) \cap \text{dom}(f)$, it holds $f(\vec{z}) \in B(y, 2^{-m})$; and in particular for all restrictions of $f|_{\mathcal{X}_v} = g_v$ to $X_v \cap B(\vec{q}_n, 2^{-n})$: hence the above search will eventually succeed.

Granting the additional jump in Theorem 6.8 seems reasonable in view of Example 6.4 since the domains of BCSS-computable functions are exactly the BCSS semi-decidable sets. In fact, one jump is in general necessary even for total functions:

**Example 6.9.** There exists a continuous function $f : \mathbb{R} \to \mathbb{R}$ computable by a constant-free BCSS machine which is not $(\rho \to \rho)$-computable.

**Proof.** Let $h : \mathbb{N} \to \mathbb{N}$ denote a recursive non-repeating enumeration of all terminating Turing machines, i.e. such that $H = h[N] \in \Sigma_1 \setminus \Delta_1$. Let $\varphi : \mathbb{R} \to \mathbb{R}$ denote a computable piecewise linear ‘hat’ function vanishing outside $[0, 1]$ and having height $\max_x \varphi(x) = \varphi(\frac{1}{2}) = 1$ and consider the non-overlapping superposition of shifts of such hats, the $m$-th one scaled by $a_m := 2^{-h(m)}$; refer to Equation 4.10 and Figure 4.9 on page 85. Again, $f$ is a continuous function on $\mathbb{R}$ vanishing for $x \leq 0$ and for $x \geq 1$; but, since $(a_m)_m$ converges by construction to 0 only noneffectively, $f$ does not admit effective evaluation at $x = 0$—formally: it has no recursive modulus of continuity.

In contrast, a constant-free BCSS machine can compute $x \mapsto f(x)$: first use the equality test to detect the cases $x \geq 1$ and $x \leq 0$ and handle them by output of $f(x) = 0$; for $0 < x < 1$ determine the unique $m \in \mathbb{N}$ such that $x \in (2^{-m}, 2^{-m+1}]$, calculate $h(m)$ by Turing machine simulation, and finally output $2^{-h(m)} \cdot \varphi(2^m x - 1)$.

6.4 Robust Quasi-Strongly $\delta$–$\mathcal{Q}$–Analytic Functions

constitute a structurally interesting class because one the one hand, it contains discontinuous functions, while on the other hand it satisfies closure under composition (cmp. Lemma 6.11 below); also it strictly includes both all functions computable according to Recursive Analysis as well as all (even discontinuous) BCSS–computable ones; see Fact 6.12 below. This model of real hypercomputation can therefore be regarded as a synthesis of those from Sections 2.3 and 2.4.2.

On the other hand, the expressive power of TTE permits this model to be incorporated in the form of another real number representation extending Example 2.58a+d), namely by replacing in Example 2.58a+d) the condition “$|q_n - x| \leq 2^{-n}$ with the asymptotic relaxation “$|q_n - x| \leq O(2^{-n})$”:

**Definition 6.10 (Asymptotic Cauchy Representation).** A $\hat{\rho}$–name for $x \in \mathbb{R}$ is some $(q_n)_n \subseteq \mathbb{Q}$ such that:

$$\exists N, C \in \mathbb{N} \forall n \geq N : \quad |q_n - x| \leq C \cdot 2^{-n}. \quad (6.2)$$

\[\text{Provided only computable real constants are employed, compare Section 2.3.4.}\]
We first show that this definition is uniformly equivalent to the one fixing \( C = 1 \); that is a \( \hat{\rho} \)-name encodes \( x \in \mathbb{R} \) in the form of a sequence of rational approximations which converge fast but with the exception of some initial segment of finite (yet unknown) length.

**Lemma 6.11.** a) Given a rational sequence \((q_n)_n\) satisfying Equation \(6.2\), one can compute a similar one satisfying Equation \(\hat{6.2}\) with \( C = 1 \).

b) It holds \( \rho \leq \hat{\rho} \leq \rho_{Cn} \); non-uniformly, a real number \( x \) is \( \rho \)-computable iff it is \( \hat{\rho} \)-computable.

c) A function \( f : \subseteq \mathbb{R} \rightarrow \mathbb{R} \) is \((\rho \rightarrow \hat{\rho})\)-computable iff it is computable by a quasi-strongly \( \delta \)-\( \mathbb{Q} \)-analytic machine.

d) \((\rho \rightarrow \hat{\rho})\)-computability is equivalent to \((\hat{\rho} \rightarrow \rho)\)-computability.

In particular, the class of \((\rho \rightarrow \hat{\rho})\)-computable functions is closed under composition.

**Proof.** a) Output every second element of the input, that is the sequence \( \tilde{\rho} \). Indeed, it follows from Equation \(6.2\) that \( |q_{2n} - x| \leq C \cdot 2^{-2n} \leq 2^{-n} \) holds for all \( n \geq N := \max\{N, \lceil \log_2 C \rceil \} \).

b) is immediate.

c) Observe that the robustness of the program \( \pi \) required in [ChHo99, top of p.157] amounts to the argument \( x \in \mathbb{R} \) of \( f \) being accessible by rational approximations \( q_n \in \mathbb{Q} \) of error \( |q_n - x| \leq 2^{-n} \), that is, in terms of a \( \rho \)-name. The output \( y = f(x) \) on the other hand proceeds by way of two sequences \( (p_m)_m, (\epsilon_m)_m \subseteq \mathbb{Q} \) such that \( \epsilon_m \to 0 \) and \( |p_m - y| \leq \epsilon_m \) hold for all sufficiently large \( m \). By effectively proceeding to an appropriate subsequence, we can w.l.o.g. suppose \( \epsilon_m = 2^{-m} \), hence \( (p_m) \) is \( \hat{\rho} \)-name of \( y \).

d) According to a), every \((\hat{\rho} \rightarrow \rho)\)-computable function is also \((\rho \rightarrow \hat{\rho})\)-computable. For the converse implication, take the Type-2 Machine \( \mathcal{M} \) converting \( \rho \)-names for \( x \in \mathbb{R} \) to \( \hat{\rho} \)-names for \( y = f(x) \). Let \( (q_n) \) satisfy Equation \(6.2\) for some unknown \( N \in \mathbb{N} \).

Now simulate \( \mathcal{M} \) on \( (q_n)_{n \geq 0} \), implicitly supposing that it is a valid \( \rho \)-name, i.e., that \( N = 0 \). Simultaneously check the consistency of Condition \(6.2\), that is, verify \( |q_n - q_k| \leq 2^{-n+1} \forall k \geq n \geq N \). If (or, rather, when) the latter fails for some \( (k_0, n_0) \), \( \mathcal{M} \) has output only finitely (say \( M_0 \in \mathbb{N} \)) many \( p_m \in \mathbb{Q} \). In that case, restart \( \mathcal{M} \) on \( (q_n)_{n \geq 1} \), presuming \( N = 1 \) while, again, checking whether this presumption is consistent with \(6.2\); but this time throw away the first \( M_0 \) elements of the sequence printed by \( \mathcal{M} \). Continue analogously for \( N = 2, 3, \ldots \).

We claim that this yields the output of a \( \hat{\rho} \)-name for \( y \). Since \( (q_n) \) is a valid \( \hat{\rho} \)-name, a feasible \( N \) will eventually be found. Before that happens, the several partial runs of \( \mathcal{M} \) have produced only finitely (say \( M_0 \in \mathbb{N} \)) many rational numbers \( p_m \); and after that, the final simulation generates according to the presumption a valid \( \hat{\rho} \)-name for \( y \). The first \( M_0 \) entries in this sequence \( (p_m)_m \) may have been exchanged by outputs of previous simulation trials; however, according to Definition \(6.10\) the representation \( \hat{\rho} \) is immune to such finite modifications.

\[ \square \]

### 6.4.1 Analytic Machines vs. BCSS-Computation or Recursive Analysis

**Fact 6.12.** a) Let \( f : \subseteq \mathbb{R}^k \rightarrow \mathbb{R} \) be \((\rho^k \rightarrow \rho)\)-computable; then it is also \((\rho^k \rightarrow \hat{\rho})\)-computable.

b) Let \( f : \subseteq \mathbb{R}^k \rightarrow \mathbb{R} \) be computable by a BCSS machine with constants from \( \Delta_1(\mathbb{R}) \) only; then it is also \((\rho^k \rightarrow \hat{\rho})\)-computable.

**Proof.** a) Since \( \rho \leq \hat{\rho} \) (Lemma \(6.11\)), \((\rho \rightarrow \rho)\)-computability implies \((\rho \rightarrow \hat{\rho})\)-computability which in turn ensures \((\rho \rightarrow \hat{\rho})\)-computability (Lemma \(6.11\)).
6.4. ROBUST QUASI-STRONGLY $\delta$-Q-ANALYTIC FUNCTIONS

b) has been established in [ChHo99, Theorem 3]. Instead of repeating the (very nice) proof, we illustrate the concept of conservative branching it employs with the example below. □

Example 6.13. Heaviside’s Function $h: \mathbb{R} \to \{0, 1\}$ is $\beta \to \hat{\beta}$-computable: Given $x \in \mathbb{R}$ by means of $(q_n) \subseteq \mathbb{Q}$ with $[6,2]$ and unknown $N \in \mathbb{N}$, let $p_n := 0$ if $q_n \leq 2^{-n}$ and $p_n := 1$ otherwise. Indeed if $x \leq 0$ then, for all $n \geq N$, $q_n \leq 2^{-n}$ and thus $p_n = 0 = f(x)$. If on the other hand $x > 0$, $x > 2^{-M}$ for some $M \in \mathbb{N}$; then for all $n \geq \max\{M + 1, N\}$, $q_n > 2^{-n}$ so $p_n = 1 = f(x)$. □

6.4.2 Analytic and Nondeterministic Machines

Quasi-strongly $\delta$-Q-analytic machines (and thus also both major models of real number computation) in turn can be simulated by a nondeterministic Type-2 machine:

Corollary 6.14. Let $f : \subseteq \mathbb{R} \to \mathbb{R}$ be $\beta \to \hat{\beta}$-computable relative to some oracle $\mathcal{O} \in \Delta^1_1$. Then $f$ is nondeterministically Type-2 computable.

Proof. The nondeterministic simulation can answer queries to $\mathcal{O}$ due to Theorem 4.72. As $\rho \equiv^n \hat{\rho} = \hat{\mu}$ by Corollary 4.71 and Lemma 6.11b), the claim follows. □

We generalize this result from the real case to arbitrary represented spaces below:

6.4.3 Finitely Revising Type-2 Computation

In order to characterize quasi-strong $\delta$-Q-analytic computability in terms of TTE, Definition 6.10 has weakened real representation $\rho$ to $\hat{\rho}$. We now introduce a generic way of turning a representation $\alpha$ into $\hat{\alpha}$ having similar properties as $\hat{\rho}$. Recall from Section 4.4.2 the ‘jump’ (meta-)mapping $\alpha \mapsto \alpha’ = \alpha \circ \iota$ on representations: it generalized the classical Definition 3.6 of revising computation from finite to the Type-2 setting: in both cases the requirement was that the sequence $(\bar{\tau})_m$ of ‘preliminary’ outputs converges to the ‘true’ one $\bar{\sigma}$—with respect to the discrete topology for the case of finite strings, with respect to Cantor topology for infinite ones. As already indicated in Section 4.4.1 Definition 3.6 a slightly different notion may make sense as well:

Definition 6.15. Let the finitely revising representation $\hat{\upsilon} : \{0, 1\}^\omega \to \{0, 1\}^\omega$ encode (via a computable standard pairing $(\cdot, \cdot) : \{0, 1\}^\omega \times \omega \to (0, 1)^\omega$ of) an infinite string $\bar{\sigma} \in \{0, 1\}^\omega$ as a sequence of infinite strings $(\bar{\tau})_m$ stabilizing (uniformly rather than character-wise) to $\bar{\sigma}$, i.e. such that

$$\exists M \in \mathbb{N} : \forall m \geq M : \bar{\tau}_m = \bar{\sigma}.$$ (6.3)

Indeed, this can also be regarded as a generalization of Definition 3.6 because a sequence of finite strings converges iff it stabilizes. For a representation $\alpha : \subseteq \{0, 1\}^\omega \to A$, call $\alpha \circ \hat{\upsilon}$ its finitely revising jump. Here come two counterparts to Lemmas 4.51 and 4.53

Lemma 6.16. It holds

a) $\iota \preceq \hat{\upsilon} \preceq \iota’$; an infinite string $\bar{\sigma}$ is $(\iota)$-computable iff it is $\bar{\upsilon}$-computable.

b) $\hat{\upsilon} \circ \hat{\upsilon} = \hat{\upsilon}$

c) $\iota’ \circ \hat{\upsilon} \equiv \iota’ = \hat{\upsilon} \circ \iota’$

d) $\prod_{n=1}^N \hat{\upsilon} \equiv \left( \prod_{n=1}^N \iota \right) \circ \hat{\upsilon}$ for $N \in \mathbb{N}$.

e) $\left( \prod_{n \in \mathbb{N}} \iota \right) \circ \hat{\upsilon} \preceq \prod_{n \in \mathbb{N}} \hat{\upsilon} \preceq \left( \prod_{m \in \mathbb{N}} \iota \right) \circ \iota’$.

f) There is a $(\prod_{n \in \mathbb{N}} \bar{\upsilon})$-computable sequence in $(0, 1)^\omega$ which is not $(\prod_{n \in \mathbb{N}} \iota)$-computable; and there is a $(\prod_{m \in \mathbb{N}} \iota \circ \iota’)$-computable sequence in $(0, 1)^\omega$ lacking $(\prod_{n \in \mathbb{N}} \bar{\upsilon})$-computability.
Lemma 6.17. Fix representations $\alpha \subseteq \{0, 1\}^\omega \to A$ and $\beta \subseteq \{0, 1\}^\omega \to B$.

a) For any function $f \subseteq A \to B$, $(\alpha \to \beta \circ \bar{\tau})$-computability is equivalent to $(\alpha \circ \bar{\tau} \to \beta \circ \bar{\tau})$-computability.

b) Every $(\alpha \to \beta)$-computable function $f$ is also $(\alpha \circ \bar{\tau} \to \beta \circ \bar{\tau})$-computable; even uniformly in $f$.

c) An $(\alpha \circ \bar{\tau})$-computable function need not be $(\alpha \to \beta)$-continuous.

Proof. It suffices to treat the case $(A, \alpha) = (B, \beta) = (\{0, 1\}^\omega, \bar{\tau})$.

a) By Lemma 6.16a), every $(\bar{\tau} \circ \tau)$-computable function is also $(\bar{\tau} \to \bar{\tau})$-computable. For the converse implication, take the Type-2 Machine $M$ converting $\bar{\tau}$-names for $x \in \mathbb{R}$ to $\bar{\tau}$-names for $y = f(x)$. Let $(\bar{\sigma}_m)$ be given with $\bar{\sigma}_m = \bar{\sigma}_M$ for all $m \geq M$, $M \in \mathbb{N}$ unknown.

Now simulate $M$ on $\bar{\sigma}_1$ (implicitly supposing $M = 1$) and simultaneously check that $\bar{\sigma}_1 = \bar{\sigma}_m$ for all $m \geq 1$. If (or, rather, when) the latter turns out to fail, restart under the presumption $M = 2$ and so on. The check will however succeed after finitely many tries (after the ‘true’ $M$ used in the input is reached). We thus obtain a finite sequence of output strings, that is a valid $\bar{\tau}$-name for $f(\bar{\sigma})$.  

b) Reducibility $(\bar{\tau} \leq \bar{\tau} \circ \bar{\tau})$ follows from a). For the converse reduction, let $(\bar{v}_k)_k$ denote a sequence in $\{0, 1\}^\omega$ with $\bar{v}_k = \langle (\bar{\tau}_m)_m \rangle$ for all $k \geq K \in \mathbb{N}$ and $\bar{\tau}_m = \bar{\sigma}$ for all $m \geq M \in \mathbb{N}$. Then $\bar{\tau}_{k,m} := \langle (\cdot)^{-1}(\bar{v}_k) \rangle_m \in \{0, 1\}^\omega$ satisfies $\bar{\tau}_n = \bar{\sigma}$ for all $n \geq N := \langle K, M \rangle \in \mathbb{N}$. It is easy to computationally extract $\bar{\tau}_n$ for all $n$ where it is defined and to otherwise define it arbitrarily.

c) Similarly, consider $(\bar{v}_k)_k$ with $\lim_k \bar{v}_k = \langle (\bar{\tau}_m)_m \rangle$ and $\lim_m \bar{\tau}_m = \bar{\sigma}$. Then, by Claim 4.49

$d)$ One reduction has already been shown in c) above. For the converse, suppose $\bar{\tau}_{n,m} = \bar{\sigma}_n$ for all $m \geq M_n$ and in particular beyond $M := \max\{M_1, \ldots, M_N\}$. Then $\langle (\bar{\tau}_{n,m})_{n=1,\ldots,N} \rangle$ stabilizes for $m \geq M$ and is thus the desired $(\prod_{n=1}^N i) \circ \bar{\tau}$-name.

f) For the Halting Problem $H \subseteq \mathbb{N}$, let $\bar{\tau}_m := 1^\omega$ if $m \in H$ and $\bar{\tau}_m := 0^\omega$ if $m \not\in H$. Semi-decidability of $H$ yields an algorithm $\bar{\tau}$-computing $\bar{\tau}_m$ from given $m$: For each $k \in \mathbb{N}$ enumerate the first $k$ elements of $H$ and output $\bar{\tau}_{k,m} := 1^\omega$ if $m$ is among them, otherwise output $\bar{\tau}_{k,m} := 0^\omega$. Since every $n \in H$ occurs in the enumeration at some position $K \in \mathbb{N}$, $\bar{\tau}_{k,m} = \bar{\tau}_{K,m}$ for all $k \geq K$; whereas in case $m \not\in H$ we have $\bar{\tau}_{k,m} = \bar{\tau}_{1,m}$ for all $k$. Thus, $(\bar{\tau}_m)_m$ is $(\prod_n \bar{\tau})$-computable indeed; were it $(\prod_n i) \circ \bar{\tau}$-computable, it would also be $(\prod_n i \circ \bar{\tau})$-computable by a contradiction of undecidability of $H$. 

This time let each $\bar{\tau}_m \in \{0, 1\}^\omega$ encode $H$ independently of $m$ by setting $\tau_{m,n} := 1$ if $n \in H$ and $\tau_{m,n} := 0$ if $n \not\in H$. Each such string is obviously $i$-computable, e.g. by Remark 4.15, and the entire (because constant) sequence $\langle \bar{\tau}_m \rangle_m$ is $(\prod_n i) \circ \bar{\tau}$-computable (Lemma 4.53). But its $(\prod_{n \in \mathbb{N}} i \circ \bar{\tau})$-computability would imply $\bar{\tau}$-computability (and thus $\bar{\tau}$-computability by a) contradiction.
b) The apply operator \((F, \tilde{\sigma}) \rightarrow F(\tilde{\sigma})\) is \((\wp^\omega \times \tilde{\tau} \rightarrow \tilde{\tau})\)-computable: if \(\tilde{\tau}_m = \tilde{\sigma}\) for all \(m \geq M\), then also \(F(\tilde{\tau}_m) = F(\tilde{\sigma})\) for all \(m \geq M\).

c) See Example 6.13 \(\square\)

### 6.5. Omissions

Scientific topics which may be regarded as belonging to Real Hypercomputation are numerous. The selection made for the present work is certainly a subjective one. Among the many relevant omissions I plead guilty of are:

- Higher recursion theory \([\text{Sack}90]\)
- Infinite-time Turing machines \(\text{(recall Section 3.3.1)}\)
- Analog computers \(\text{(recall Section 2.6.4)}\)
- Refined hierarchies of real numbers \([\text{Zhen}07]\)
- And certainly many more.

#### 6.4.4 Finitely Revising Real Representation

Similarly to Proposition 4.66, we have

**Proposition 6.18.** The asymptotic Cauchy Representation \(\hat{\rho}\) from Definition 6.10 is uniformly equivalent to \(\rho \circ \hat{\tau}\).

*Lemma 6.17(a) thus generalizes Lemma 6.11(b) and Lemma 6.17(b) includes Fact 6.12(b).*

*Proof. \(\rho \circ \hat{\tau} \leq \hat{\rho}\): Given \((\tilde{\tau}_m)_m\) with \(\tilde{\tau}_m = \tilde{\sigma}\) for all \(m \geq M\), consider the following decidable property \(P(m, n)\):

An initial segment of \(\tilde{\tau}_m\) (of length \(\ell(m, n)\), say) decodes to a (finite) rational sequence \(q_1, \ldots, q_n\) with \(|q_i - q_j| \leq 2^{-i+1}\) for all \(1 \leq i \leq j \leq n\); moreover, \(\tilde{\tau}_m, \tilde{\tau}_{m+1}, \ldots, \tilde{\tau}_{m+n-1}\) all coincide up to the first \(\ell(m, n)\) symbols.

Our algorithm starts with \(m := 1\) and outputs, iteratively for \(n = 1, 2, \ldots, q_n\) as defined above as long as \(P(1, n)\) holds; otherwise it proceeds to \(m := 2\) and, now continuing with \(n, n+1, n+2, \ldots\), appends \(q_n\) to the output—possibly until \(P(2, n)\) fails; and so on.

In each phase \(m < M\), i.e. when \(\tilde{\tau}_{m'} \neq \tilde{\tau}_M\) for some \(m' < M\), the above coincidence condition will fail for some \(n\) and then increase \(m\). This means that after output of finitely many \(\text{(say } N\text{)}\) elements, one arrives at phase \(M\) where \(\tilde{\tau}_{m'} = \tilde{\sigma}\) for all \(m' \geq M\) is a valid \(\rho\)-name; therefore \(P(M, n)\) is true for all \(n\), that is, the algorithm appends to the output an infinite rational sequence \(q_N, q_{N+1}, \ldots\) satisfying \(|q_i - q_j| \leq 2^{-i+1}\) for all \(N \leq i \leq j\). This constitutes (modulo dropping the very first element in order to change error bound \(2^{-i+1}\) to \(2^{-i}\)) a \(\hat{\rho}\)-name as desired.

\(\hat{\rho} \leq \rho \circ \hat{\tau}:\) Given a rational sequence \((q_n)_n\) satisfying Equation (6.2). Let \(m := 1\) and, for \(n = 1, 2, \ldots\), output finite initial segments to identical strings \(\tilde{\tau}_1, \tilde{\tau}_2, \ldots, \tilde{\tau}_n\) consisting of encodings of \((q_2, q_3, \ldots, q_n)\) as long as all \(1 \leq i \leq j \leq n\) satisfy \(|q_i - q_j| \leq 2^{-i+1}\). In case \(N = 1\), this iteration will continue indefinitely and output a \(\rho \circ \hat{\tau}\)-name as desired (notice that, as above, we dropped the very first rational approximation in order to compensate for the factor 2 in \(2^{-i+1}\)). Otherwise extend arbitrarily the partial imprints of \(\tilde{\tau}_1, \tilde{\tau}_2, \ldots, \tilde{\tau}_n\); simultaneously output, now for \(n, n+1, n+2, \ldots\), output initial segments to strings \(\tilde{\tau}_n, \tilde{\tau}_{n+1}, \ldots, \tilde{\tau}_{n+m}\) encoding \((q_{n+1}, q_{n+2}, \ldots, q_{n+m})\) as long as \(|q_{n+i} - q_{n+j}| \leq 2^{-i}\) for all \(1 \leq i \leq j \leq m\), and so on until some \(n \geq N\) is reached from when on condition \(|q_{n+i} - q_{n+j}| \leq 2^{-i}\) is always fulfilled. \(\square\)
Bibliography


BIBLIOGRAPHY


[Rick00] J. Rickard: “Re: The degree of this field”, Internet Usenet (2000); http://www.math.niu.edu/~rusin/known_math/00_incoming/sqrt_q


[Turi36] A.M. TURING: “On Computable Numbers, with an Application to the Entschei-

[Turi37] TURING, A.M.: “On Computable Numbers, with an Application to the Entschei-


[Zhen07] X. ZHENG: “On the hierarchy of $\Delta_2$ real numbers”, to appear in Theoretical Informatics and Application. 139


Own Publications (preceeding Dissertation)


Own Publications (succeeding Dissertation)


## Index

(\(\alpha \rightarrow \beta\))-computable, see also representation, 36
(\(\alpha \rightarrow \beta\))-continuous, see also representation, 36
\(<\), see reducible, BCSS
\(\preceq\), see representation reducibility
\(\succeq^n\), see reducible, nondeterministic
\([\alpha \rightarrow \beta]\), 38
\([\rho \rightarrow \rho_\prec]\), see representation of LSC
\([\rho_\prec \rightarrow \rho]\), see representation of MLSC
\([\rho \rightarrow \rho]\), see Weierstraß representation
\(\mathcal{A}\), see algebraic reals
\(B\), open Euclidean ball, 18
\(\chi\), see function, characteristic
\(\delta_{\Sigma^a}\), see representation on Borel’s Hierarchy
\(f : \subseteq A \rightarrow B\), see function, partial
\(\text{Fin}\), see Problem, Finiteness
\(H\), see Halting Problem
\(\mathbb{H}\), see Halting Problem, real
\(i\), see representation of \(\{0, 1\}^\omega\)
\(i'\), see representation of \(\{0, 1\}^\omega\)
\(j\), see representation of \(\{0, 1\}^\omega\)
\(\lambda\), see function, characteristic
\(LSC\), see function, semi-continuous
\(MLSC\), see function, nondecreasing, see function, semi-continuous
\(\nu\), see representation of \(\mathbb{Q}\)
\(\mathcal{O}\), see open sets, class of
\(\psi_\nu\), see representation of closed sets
\(\rho\), see representation, Cauchy
\(\rho_\succeq\), see representation, right
\(\rho_\prec\), see representation, asymptotic Cauchy
\(\rho_\prec\), see representation, left
\(\rho_{\text{Cu}}\), see representation, naive
\(\rho_{\rightarrow}\), see also \((\alpha \rightarrow \beta)\), 31
\(\text{sup}_K\), 20
\(T\), see transcendental reals
\(\theta_\prec\), see representation of \(\mathcal{O}\)
\(\text{Tot}\), see Problem, Totality

admissible, see representation, admissible algebraic

admissible element, 16

field extension, 16, 28

independence, 18, 107

real, 16, 109

topology, 25

Analytic

Machine, 10 [113] [135] [137]

set, 19

Apple, 12, 13

apply operator, 80, 82, 86, 90

Aristotle, 60

arithmetic hierarchy, see Hierarchy, Arithmetical

automaton

Büchi, 100

cellular, 57

finite, 100

axiom

of choice, 16

Zermelo-Fraenkel-atic set theory, 16

Baire

Category, see Theorem, Baire

René–Louis, 19

space, 32, 66

BCSS Machine, 21

linear, 112

nondeterministic, 28, 29, 118

universal, see universal real machine

with exponentiation, 105

with oracle, 106

Bell’s Inequality, 53

Bézout, Étienne, 20

Black Hole, 13, 14, 53, 60

Borel

\(\sigma\)-algebra, 19, 118

Félix Édouard Justin Émile, 19

Hierarchy, see Hierarchy, Borel

Brouwer, Luitzen Egbertus Jan, 20, 60

Cantor

Georg Ferdinand Ludwig Philipp, 36

set, 19, 23, 24, 34, 115

space, 19, 31, 36, 38

topology, 36, 37, 89, 137

Cantor space, 90

characteristic function, see function, characteristic

Church
INDEX

∼’s Thesis, 42
Alonzo, 12
Church–Turing Hypothesis, 12, 13, 52, 54, 59
extended, 12
harnessable, 13, 60
physical, 12, 13
strong, 12
closed, see ψ
co-r.e., 9, see also r.e., 34
Cohen, Paul Joseph, 16
computability
non-uniformly, 35, 36
computable
BCSS, 22, 27
relatively, 106
α∼, see representation
(α→β)∼ function, see (α→β)
binarily ∼ real number, 31, see also representation, binary real, 31
Cauchy∼ real number, 30, see also representation, Cauchy, 30, 31, 33, 69
left ∼ real number, see also left representation, 34
relatively, 70
Markov, see Markov computability
naively ∼ real number, 30, see also representation, naive, 30, 69
non–uniformly, 31, 33, 36
real function, 33, see also effectively continuous, see also effectively evaluable, see also Weierstraß computable relatively, 72
real number, 31, see also computable, binarily, see also computable, Cauchy, see also computable, left, 36
relatively, 69
real sequence, 35, 38
relatively, 48, see also oracle, 49, 65
revisingly, 48, 59, 99
uniformly, 31, 33, 36, 58
upper ∼ real number, 34
Weierstraß∼ real function, see also Weierstraß representation, 31, 33
continuous
(α→β)∼, see (α→β)–continuous effectively, see effectively continuous
ε–δ, 58
function, see function, continuous sequentially, 38
uniformly, see uniform continuity
Conway, John Horton, 57, see also Life, Game of Continental
Coulomb
Charles-Augustin de, 55
Law, see law, Coulomb’s cube
Klee–Minty, 22
decidable, 9
BCSS–, 22, 28, 106
relatively, 106
semi–, see semi–decidable
Dedekind cut, 34, 36, 37
degree
algebraic, 16, 26
field extension, 16, 28
polynomial, 25
quynomial, 25
transcendence, 18, 28, 108
differential equation
algebraic, 44
ordinary, 15, 41, 55
partial, 15, 88
Dirac, 14
domain
invariance of, 26
type, 42
effectively
Borel semi-measurable, 85
Borel-measurable, 85
continuous, 23, 33, 85
evaluable, see also (ρ→ρ), 33
semi-continuous, 85, 86
semi-evaluable, 33, see also (ρ→ρ<), 85
Einstein, Albert, 13, 14, 53
Electricity, 52
Ellipsoid Method, 11
entanglement, 53
enumerable, see r.e.
EPR, see paradox, Einstein–Podolski–Rosen equation
Bailey–Borwein–Plouffe, 30
differential, see differential equation
Diophantine, see Diophantine Equation
Dirac, 14
system of linear ∼-s, 22
Euclid, 11, 110
Exact Computation Paradigm, 14, 45
Fermat
∼’s Last Theorem, 10
Pierre de, 10
Feynman, Richard Phillips, 53
function
INDEX

algebraic differential, 44
Borel semi-measurable, 85, 98
Borel-measurable, 85, 98
characteristic, 22, 33, 34
    padded, 101
computable real, see computable real function
continuous, 20, 25, 32, see also Main Theorem, 35
    \((\alpha \rightarrow \beta) \sim, \) see \((\alpha \rightarrow \beta)\)-continuous
    effectively, see effectively continuous
    uniformly, see uniform continuity
generalized, 88
graph, 22, 23
Heaviside, 32, 39, 43, 73, 79, 88, 99
injective, 25
multi-valued, 39
nondecreasing, 73, 74, 81, 98
open, 13
pairing, see pairing function
partial, 10, 22, 26, 36
rational, see quynomial
\((\alpha \rightarrow \beta)\)-realization, 36
semi-continuous, 20, 73, 74, 80, 81, 88
surjective, 36
totally discontinuous, 26

Gauß
    Elimination, 11, 22, 110
Johann Carl Friedrich, see Lemma or Elimination
    Lemma, 17
Global Positioning System, see GPS
Gödel
    Incompleteness Theorem, 9
    index, 10, 43, 44, 88
    Kurt, 9, 16
    Prize, 11
Gödelization, 10, 23, 29, see also Gödel index
Goldbach Conjecture, 10
GPS, 23
Grzegorczyk, Andrzej, 33

Halting Problem, 9, 10, 10, 12, 13, 27, 30
    47, 52, 54, 56, 58, 60, 138
for C64, 11
for iMac, 11
iterated, 49, 114
numerical, 88
real, 23, 109, 114, 118, 123, 129
    linear, 112
    nondeterministic, 118
Heaviside, Oliver, see function, Heaviside

Hierarchy
    Analytical, 51
    Arithmetical, 49, 67, 71, 89, 97, 115
    BCSS, 114, 117
    of real numbers, 69, 70, 102
    of real representations, 70
    relativized, 65
    Borel, 19, 51, 67, 115, 117
    effective, 66, 68, 101
    Chomsky, 58, 100
    Hyperarithmetical, 51, 101
    Polynomial, 51
    Projective, 19
Hilbert
    \(\sim\)’s Program, 9
    \(\sim\)’s Tenth Problem, 11, 28, 119
    David, 9, 60
    space, 54
hypercomputer, 47
Hypothesis
    Church–Turing, see Church–Turing Hypothesis
    Continuum, 16
    Riemann, see Riemann Hypothesis
inequality
    Bell, 53
    system of linear inequalities, 22
    system of polynomial (in)equalities, see semi-algebraic
jump, 49, 52
Kleene, Stephen Cole, 49, 52
law
    Coulomb’s, 55
    Moore’s, 59
    Newton’s, see Newton of physics, 14
    of thermodynamics, see also thermodynamics
Lebesgue
    Henri Léon, 19
Leibniz, Gottfried Wilhelm, 60
Lemma
    Gauß, see Gauß Lemma
    König, 42, 55
    Limit, 48, 69
    Padding, 10, 48
library
    Core, 45
    iRRAM, 45
    LEDA, 45
Life, Game of, 57
Limited Principle of Omniscience, 14
Lindemann
~Weierstraß Theorem, 18
Ferdinand von, 18
linear
system of ~ equations, 22
system of ~ inequalities, 22
system of ~ inequalities, 23
LPO, see Limited Principle of Omniscience
machine
BCSS, see BCSS Machine
analog, 44
analytic, 137, see Analytic Machine
hyper, see hypercomputer
infinite–time, 56
nondeterministic, 26
BCSS, see BCSS Machine
fair, 56
Turing, 28, 56
Type-2, 99, 103, 137
random access ~, see Random Access Machine
parallel, 15
revising, 48
Turing, see Turing machine
universal, see universal machine
Main Theorem of Recursive Analysis, 32, see also function, continuous, 33, 37, 38
40, 43, 65, 72, 89
Malament–Hogarth Space–Time, 53, see also Relativity, Theory of General, 56
Mandelbrot Set, 23
Markov
Andrey Andreyevich Jr., 43
computability, 37, 43
Principle, 44
Maxwell, James Clerk, 52
Mechanics
classical, 54
Newtonian, 13, 52, 54
Quantum, see Quantum Mechanics
Michelson&Morley experiment, 52
name, see representation
Neumann, John von, 11
Newton
Gravitation, 55
Isaac, see also Mechanics, Newtonian, 55
Iteration, 23
Law of Motion, 15, 55
nondeterminism, see machine, nondeterministic
ODE, see differential equation, ordinary open
function, 43
set, 29, 33
class of ~s, 18, 37
representation of ~s, see representation of open sets
oracle, 47, see also computable, relatively
ordinal, 19
Church–Kleene, 51, 67
recursive, 51, 67
Painlevé, Paul, 55
pairing function
integer, 10
mixed, 23
real, 23, 25, 109
paradigm
Exact Computation, 44
paradox
Einstein–Podolski–Rosen, 53
Twins, 53
parallel
cellular automaton, see automaton, cellular
computing, 57
infinite ~ism, 54, 57
quantum ~ism, see Quantum parallelism
random access machine, see Random Access Machine, parallel
Podolski, Boris, 53
Poincaré, Jules Henri, 55
Post
~’s Problem, see Problem, Post’s
Emil Leon, 19, 52
Normal Form Theorem, see Theorem, Post
PRAM, see Random Theorem, Post
PRAM, see Random Access Machine, parallel
Principle
Limited ~, of Omniscience, see Limited Principle of Omniscience
Markov’s, 44
Prize
Gödel, 11
Millennium, 35
Problem
diagonal, 10
Finiteness, 48, 50, 117
Halting, see Halting Problem
Hilbert’s Tenth, see Hilbert’s Tenth Problem
integer factorization, 53
Linear Programming, see linear system of inequalities
Non-emptiness, 29
Post’s, 52
  linear, 112
  real, 109
real decision, 22
Totality, 48, 68, 67, 118
Word, see Word Problem

quantifier
  alternating, 43, 49, 51
  bounded, 50
  constructive interpretation, 42, 43
  elimination, 20, 28, 96, 107, 118
  polynomial, 51
  swapping, 35

Quantum
  computer, 53, 54
  entanglement, see entanglement
Field Theory, 13
Gravitation, 12, 60
Leap, 60
Mechanics, 13, 53, 54
parallelism, 60
system, 54
qubit, 53
quolynomial, 23, 24, 27, 29, 106, 108, 111
  113, 117, 134, 135

r.e., 9 see also semi-decidable, 34
BCSS, 22, 128
  relativized, 106, 108, 117
co-, see co-r.e.
open, 34 see also $\theta_r$, 65
RAM, see Random Access Machine
Random Access Machine, 11, 15
parallel, 15
realization, see function realization
recursive, 9 see also decidable, 93
recursively enumerable, see r.e.
reducible
BCSS, 111
  many-one, 47, 49, 115
non-deterministic, see representation reducibility
Turing, 47
relative computability, see computable, relatively
Relativity, see also Einstein, Albert
Theory of Special and General, 13, 14
representation, 36, 37
  admissible, 38, 72, 92
equivalence, 36
jump, 92
  finitely revising, see revising
of $\{0, 1\}^\omega$, 90
of $C(R)$
  Weierstrass, 37
of closed sets, 67
of $\mathcal{L}_2$, 80, 85
of MLSC, 84
of open sets, 37 see also r.e. open
of $\mathbb{Q}$, 37
of $\mathbb{R}$
  Cauchy, 37, 70
  left, 37
  naive, 37, 70
  right, 37
on Borel’s Hierarchy, 67
reducibility, 36
  nondeterministic, 101, 102
  revising, 90
  finitely, 137
  standard, 93

Riemann
  Georg Friedrich Bernhard, 10
  Hypothesis, 10
Rosen, Nathan, 53

semi-algebraic, 20
  basic, 20, 45, 146
  countably, 20, 67, 115
function, 26
over a real subfield, 20, 27, 35, 114, 116
  117
semi-decidable, 9 see also r.e.
BCSS, 22, 28, 29, 115, 123
  relativized, 107, 108, 117
BCSS
  relativized, 106
Recursive Analysis, 34 see also r.e. open
semi-continuous, see continuous, semi
Shannon, Claude Elwood, 44
Simplex Algorithm, 22
software verification, automated, 10, 48
space
  Baire, see Baire space
  Cantor, see Cantor space
  Hilbert, see Hilbert space
  vector $\sim$, 14
Sundman, Karl Frithiof, 55
Suslin
  Mikhail Yakovlevich, 19
  Theorem, 19
Tarski, Alfred, 20, 28, 118
tertium non datur, 13
theorem
  Baire, 19, 25, 114
Cook’s, 51
Fermat’s Last, see Fermat’s Last Theorem
Gödel’s Incompleteness, 9
Lindemann-Weierstraß, 18
Main ∼, of Recursive Analysis, see Main Theorem of Recursive Analysis
Post Normal Form, 49, 116
Rice’s, 10
SMN, 10, 23, 38, 43, 44
UTM, 10, 23, 38, 43, 44
Weierstraß Approximation, 20, 37, 42

effective, 33, 83
Thermodynamics, 13
topology, 18, see also open set
algebraic, 25
Cantor, see Cantor topology
Euclidean, 18, 38
final, 38
second countable, 18, 92
transcendental, 16, 107, 109
transfinite, 19
TTE, 36, see also representation
Turing
Alan Mathison, 9, 12, 30, 31, 47, 55
complete, 118
degree, 47, 92
hypothesis, see Church–Turing Hypothesis
machine, see anywhere
nondeterministic, see machine, nondeterministic
probabilistic, 12
reducible, 47
Twins’ Paradox, 53
Type-2 Theory of Effectivity, see TTE

uniform
computability, see computable, uniformly
continuity, 85
convergence, 31, 92
non∼
computability, see computable, non–uniformly
universal
computation, 13, 92, 148
Life, 57
mechanical system, 54
real machine, 23, 115, 129
Turing machine, 10, 54, 99
simulation overhead, 59

Weierstraß
–computable, see computable, Weierstraß
Approximation Theorem, see Theorem, Weierstraß
Karl Theodor Wilhelm, 20, 42
Lindemann-∼, see Lindemann-Weierstraß
Word Problem, 11, 12, 119, 130
ZFC, 16
Zuse, Konrad, 12