Degrees of Unsolvability for Additive BSS Machines

Christine Gaßner
Ernst-Moritz-Arndt-Universität Greifswald, Germany
gassnerc@uni-greifswald.de

Abstract: We consider several halting problems for the additive real BSS machines and the Turing reducibility by additive machines using only a few number of permitted constants. We use several techniques in order to present a problem below the Halting Problem for the additive machines using only the constants 0 and 1 and the equality test, an infinite hierarchy of problems between the latter problem and the Halting Problem for the additive machines using only the constants 0 and 1 and the order test, and finite hierarchies above these problems where the reductions of these finite hierarchies to the Halting Problem for all additive BSS machines are only possible if the machines are encoded in a special way. (Version of December 2009)

Key Words: BSS model, additive machines, oracle machine, Turing degrees
Category: F.1, F.1.1, F.1.2

1 Introduction

The *Turing degrees* with respect to the BSS model are the equivalence classes defined by means of Turing reductions which can be performed by BSS oracle machines. These degrees allow a characterisation of hard problems with respect to the BSS model. In [Meer and Ziegler 2008] and [Meer and Ziegler 2006], the many-one reducibility as well as the Turing reducibility are discussed. For additive machines whose constants are restricted to be 0 and 1, Klaus Meer and Martin Ziegler posed the question whether there are degrees of unsolvability which are lower than the degree of the Halting Problem and higher than one of the set of rational numbers. Inspired by this question we want to consider the Turing reducibility, at first, by additive machines using only 0 and 1 as constants without additional real parameters (that means without real machine constants in $\mathbb{R} \setminus \{0, 1\}$), and later by machines with parameters. Then, in both settings we can say that a problem $\mathcal{A}$ is *easier* (*to solve*) than a problem $\mathcal{B}$, if $\mathcal{A}$ is decidable by an oracle machine using $\mathcal{B}$ as oracle where we allow real parameters in the reduction procedure or we do not allow parameters. If $\mathcal{B}$ is not easier that $\mathcal{A}$, then the problem $\mathcal{A}$ is *strictly easier* to solve than $\mathcal{B}$ and both problems are members of two different Turing degrees. Note that the many-one reducibility as well as the Turing reducibility without real parameters refine the equivalence classes under Turing reducibility with parameters. Since the halting problem for a class of BSS machines cannot be solved by a machine in this class we have in any case two different degrees denoted by $\emptyset$ and $\emptyset'$: The class consisting of all the decidable problems and the class consisting of all those semi-decidable sets which are equivalent to the halting problem. The problems of the latter class are called *complete* for the class of semi-decidable problems with respect to the considered reducibility relation.

Whereas we mainly speak about decidable and semi-decidable (or recognizable) sets, in the classical theory of recursive functions and in the theory of recursive reducibility, the corresponding notions are recursive and recursively enumerable sets. The *recursive sets* $S \subseteq \mathbb{N}$ are decidable by Turing machines that means that their characteristic functions are total recursive functions. A set $S \subseteq \mathbb{N}$ is *recursively enumerable* (or *effectively enumerable*) if a bijective function $f : \mathbb{N} \to S$ is computable by a Turing machine. Therefore, these sets also are the halting sets of Turing machines and, thus, they can be reduced to the Halting Problem for Turing machines with respect to the many-one reducibility relation and they are semi-decidable, that means that their partial characteristic functions are computable. Thus, a set of positive integers is recursive if and only if the set itself and its complement are effectively enumerable. The halting sets of BSS machines can be reduced to the Halting Problem for BSS machines and their partial characteristic functions are computable. Thus the halting sets of BSS machines are semi-decidable, too, and a set is decidable if and only if the set itself and its complement are semi-decidable. However, there are halting sets $S \subseteq \mathbb{R}^\infty$ (e.g. $S = \mathbb{R}$) which are not effectively enumerable by computing a bijective function $f : \mathbb{N} \to S$. Note that this definition of the notion *effectively enumerable* differs from the definition given in [Blum et al. 1989].

For the classical model of computation based on Turing machines, Emil Post showed in [Post 1944] that there are undecidable semi-decidable problems which are not complete with respect to many-one reducibility and he posed the problem whether there are degrees between both corresponding Turing degrees of decidable and complete semi-decidable problems, respectively. The positive answer to this problem was presented by Friedberg (in [Friedberg 1957]) and Muchnik by means of a technique using the finite injury priority method. Some modern versions of the proofs and their relationship to the classical results are presented in [Soare 1987], [Börger 1992], and [Kozen 2006]. The constructions are based on diagonalization techniques which can be used since all halting sets of the Turing machines and all Turing machines are enumerable. In [Meer and Ziegler 2008] Klaus Meer and Martin Ziegler gave a solution to Post's Problem for the full BSS model over the real numbers. This solution significantly differs from its classical, discrete variant where only diagonalization techniques are known to yield the existence of Turing degrees below the classical Halting problem. They proved the existence of an infinite number of semi-decidable Turing degrees below the real Halting problem in the BSS model and the fact that the rational numbers are strictly easier than the real Halting problem. For the additive BSS machines without real constants they showed that the set of quadratic rational numbers is strictly easier than the Halting problem and they posed the question: Does Q have the same degree of undecidability as the Halting Problem for additive BSS machines without real constants?

We want to discuss several variants of this question for any additive real BSS machines where we only restrict the domains of constants used in the reductions. On one hand, we will use the strong reductions without real parameters in order to better understand the hardness of decision problems over the reals. On the other hand, the study of additive machines using real constants shows the great power of single real numbers. Whereas in the classical case, the main ideas are of logical nature, we can also use algebraic and topological properties of the real numbers and, in addition to these ones, some logical techniques.

2 The Model of Computation and Halting Problems

The uniform BSS model of computation was introduced in [Blum et al. 1989] (cp. also [Blum et al. 1998]). The additive BSS machines were considered, for instance, in [Koiran 1994], [Cucker and Koiran 1995], and [Meer and Ziegler 2008]. They can perform labelled instructions of the form $Z_i := Z_j + Z_k$, $Z_i := Z_j - Z_k$, $Z_j := c$, *if* $Z_j = 0$ *then goto* l_1 *else goto* l_2, *if* $Z_j \geq 0$ *then goto* l_1 *else goto* l_2, $Z_{I_j} := Z_{I_k}$, $I_j := 1$, $I_j := I_j + 1$, and *if* $I_j = I_k$ *then goto* l_1 *else goto* l_2. Each assignment of an input $(x_1, \ldots, x_n) \in \bigcup_{i \geq 1} \mathbb{R}^i$ to the registers of a machine $\mathcal{M}$ can be realized by $Z_1 := x_1; \ldots ; Z_n := x_n; I_1 := n; \ldots ; I_{k_\mathcal{M}} := n$. We denote the class of additive BSS machines by $\mathsf{M}_{\mathrm{add}}$.

Let $\mathsf{M}^1_{\mathrm{add}}$ be the class of the additive BSS machines using only the constants 0 and 1 in the instruction $Z_j := c$. For any oracle $\mathcal{O} \subseteq \bigcup_{i \geq 1} \mathbb{R}^i$, let $\mathsf{M}^1_{\mathrm{add}}(\mathcal{O})$ be the class of the additive oracle machines which can execute additional instructions of the form *if* $(Z_1, \ldots, Z_{I_1}) \in \mathcal{O}$ *then goto* l_1 *else goto* l_2. We say that a problem $\mathcal{A} \subseteq \bigcup_{i \geq 1} \mathbb{R}^i$ is *easier than* a problem $\mathcal{B} \subseteq \bigcup_{i \geq 1} \mathbb{R}^i$ (denoted by $\mathcal{A} \preceq \mathcal{B}$ here) if $\mathcal{A}$ is decidable by a machine in $\mathsf{M}^1_{\mathrm{add}}(\mathcal{B})$. The problem $\mathcal{A}$ is *strictly easier than* $\mathcal{B}$ (denoted by $\mathcal{A} \precneqq \mathcal{B}$ here) if $\mathcal{B}$ cannot be decided by a machine in $\mathsf{M}^1_{\mathrm{add}}(\mathcal{A})$.

We will consider the following Halting Problem $\mathbb{H}^1_{\mathrm{add}}$ for the machines in $\mathsf{M}^1_{\mathrm{add}}$ using only the constants 0 and 1 in the instruction $Z_j := c$, the Halting Problem $\mathbb{H}^{1,=}_{\mathrm{add}}$ for the machines in $\mathsf{M}^{1,=}_{\mathrm{add}} \subset \mathsf{M}^1_{\mathrm{add}}$ performing only tests of the form $Z_j = 0$?, and the Halting Problem $\mathbb{H}_{\mathrm{add}}$ for all additive BSS machines. The codes code($\mathcal{M}$) of the machines $\mathcal{M} \in \mathsf{M}^1_{\mathrm{add}}$ are sequences of the codes of the single symbols of their programs where, for some k, each of the single symbols is encoded by k symbols in $\{0, 1\}$ unambiguously. For any BSS machine $\mathcal{M} \in \mathsf{M}_{\mathrm{add}}$, let code($\mathcal{M}$) be the usual code of its program where any real constant is encoded by itself and the other single symbols are encoded by k zeros and ones (for some k). To simplify matters, in the following definitions we write $(n, \mathbf{x}, \mathrm{code}(\mathcal{M}))$ instead of $(n, x_1, \ldots, x_n, s_1, \ldots, s_m)$ if $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathrm{code}(\mathcal{M}) = (s_1, \ldots, s_m)$.

$$\begin{aligned}\mathbb{H}^{1,=}_{\mathrm{add}} &= \bigcup_{n\geq 1}\{(n,\mathbf{x},\mathrm{code}(\mathcal{M})) \mid \mathbf{x}\in\mathbb{R}^n \ \&\ \mathcal{M}\in\mathsf{M}^{1,=}_{\mathrm{add}} \ \&\ \mathcal{M}(\mathbf{x})\downarrow\},\\ \mathbb{H}^{1}_{\mathrm{add}} &= \bigcup_{n\geq 1}\{(n,\mathbf{x},\mathrm{code}(\mathcal{M})) \mid \mathbf{x}\in\mathbb{R}^n \ \&\ \mathcal{M}\in\mathsf{M}^{1}_{\mathrm{add}} \ \&\ \mathcal{M}(\mathbf{x})\downarrow\},\\ \mathbb{H}_{\mathrm{add}} &= \bigcup_{n\geq 1}\{(n,\mathbf{x},\mathrm{code}(\mathcal{M})) \mid \mathbf{x}\in\mathbb{R}^n \ \&\ \mathcal{M}\in\mathsf{M}_{\mathrm{add}} \ \&\ \mathcal{M}(\mathbf{x})\downarrow\}\end{aligned}$$

where $\mathcal{M}(\mathbf{x})\downarrow$ means that $\mathcal{M}$ halts on $\mathbf{x}$. It is easy to see that

$$\mathbb{H}^{1,=}_{\mathrm{add}} \preceq \mathbb{H}^{1}_{\mathrm{add}} \preceq \mathbb{H}_{\mathrm{add}}.$$

We will define a problem below $\mathbb{H}^{1,=}_{\mathrm{add}}$ by means of a priority method often used in the classical recursion theory for constructing recursively enumerable sets and we will present a hierarchy between $\mathbb{H}^{1,=}_{\mathrm{add}}$ and $\mathbb{H}^{1}_{\mathrm{add}}$ as well as hierarchies between $\mathbb{H}^{1}_{\mathrm{add}}$ and $\mathbb{H}_{\mathrm{add}}$. Since we want to derive the latter hierarchies from the halting sets of some additive machines which can also use real constants, the construction is also dependent on a special encoding technique used to encode the real constants. Furthermore, we will shortly characterize the relationship between these problems and the Halting Problem for non-deterministic additive machines.

The question posed by Meer and Ziegler can be formalized as follows.

Question. Does $\mathbb{Q} \precneqq \mathbb{H}^{1}_{\mathrm{add}}$ hold?

3 $\mathbb{Q}$ is Strictly Easier than $\mathbb{H}^{1,=}_{\mathrm{add}}$

In [Gaßner 2008] we considered a list of problems of n-tuples over the real numbers which allow to compute integers from these tuples by additive machines without using machine constants. For

$$\mathbb{L}_n = \{(x_1,\dots,x_n)\in\mathbb{R}^n \mid (\exists(q_0,\dots,q_{n-1})\in\mathbb{Q}^n)(q_0+\textstyle\sum_{i=1}^{n-1} q_i x_i = x_n)\},$$

we showed

$$\mathbb{Q} = \mathbb{L}_1 \precneqq \mathbb{L}_2 \precneqq \cdots \precneqq \mathbb{L} = \bigcup_{n\geq 1}\mathbb{L}_n \preceq \mathbb{H}^{1}_{\mathrm{add}}.$$

More precisely, we have $\mathbb{L}_1 \precneqq \mathbb{L}_2 \precneqq \cdots$ and, for any $k\geq 1$, $\mathbb{L}_k \precneqq \mathbb{L}$. Since the order test is not necessary for recognizing $\mathbb{L}$, we have even $\mathbb{L} \preceq \mathbb{H}^{1,=}_{\mathrm{add}}$. This example shows that we can use algebraic and topological properties for constructing a whole hierarchy of undecidable problems below the Halting Problem $\mathbb{H}^{1,=}_{\mathrm{add}}$.

On the other hand, we can construct a set $\mathcal{A}\subset\mathbb{N}$ by the priority method as given in [Soare 1987] such that $\mathcal{A} \not\preceq \mathbb{Q}$ and $\mathcal{A} \preceq \mathbb{H}^{1,=}_{\mathrm{add}}$ and, consequently, $\mathbb{Q} \precneqq \mathbb{H}^{1,=}_{\mathrm{add}}$ are satisfied. This construction is possible since the machines in $\mathsf{M}^{1}_{\mathrm{add}}(\mathcal{O})$ as well as in $\mathsf{M}^{1,=}_{\mathrm{add}}$ are countable and their codes can be effectively enumerated by additive machines in $\mathsf{M}^{1,=}_{\mathrm{add}}$. We will also use the following lemma.

Lemma 1. *The intersection of the halting set of any machine in* $\mathsf{M}_{\mathrm{add}}^{1,=}$ *and* $\mathbb{N}$ *is effectively enumerable and, consequently, it is a halting set of a machine in* $\mathsf{M}_{\mathrm{add}}^{1,=}$.

Proof. Let $H_{\mathcal{M}}$ be the halting set of any machine $\mathcal{M} \in \mathsf{M}_{\mathrm{add}}^{1,=}$. Then, a machine can enumerate all pairs $(n_1, t_1), (n_2, t_2), \ldots$ for which $\mathcal{M}$ halts on n_i after exactly t_i steps. For the input j, the enumeration machine $\bar{\mathcal{M}}$ compute the first pair (n_1, t_1) from 1 and each further pair (n_{i+1}, t_{i+1}) step-by-step from its enumerated predecessor (n_i, t_i), and outputs the j^{th} integer n_j. $\square$

Our definition of $\mathcal{A}$ follows [Börger 1992]. For any machine $\mathcal{M} \in \mathsf{M}_{\mathrm{add}}^{1}(\mathcal{O})$, let $k_{\mathcal{M}} = 2^{|\mathrm{code}(\mathcal{M})|} + c_{\mathcal{M}}$ where $c_{\mathcal{M}}$ is the integer whose binary code matches with $\mathrm{code}(\mathcal{M}) \in \{0,1\}^{\infty}$. If a number i does not match with a program, then let the corresponding machine $\mathcal{M}_i^{\mathcal{O}}$ be a simple machine that halts in any case and let $k_{\mathcal{M}_i^{\mathcal{O}}} = i$. Thus, we can take the numbers $1, 2, \ldots$ for listing all additive oracle machines $\mathcal{M}_1^{\mathcal{O}}, \mathcal{M}_2^{\mathcal{O}}, \ldots$. For any oracle set $\mathcal{O} \subseteq \mathbb{N}$ we introduce the following special halting problem.

$$\mathbb{K}^{\mathcal{O}} =_{\mathrm{df}} \mathbb{H}_{\mathrm{spec}}(\mathsf{M}_{\mathrm{add}}^{1}(\mathcal{O})) =_{\mathrm{df}} \{k_{\mathcal{M}} \mid \mathcal{M} \in \mathsf{M}_{\mathrm{add}}^{1}(\mathcal{O}) \;\&\; \mathcal{M}(k_{\mathcal{M}}) \downarrow\}.$$

Moreover, let $\mathcal{N}_1, \mathcal{N}_2, \ldots$ be the machines in $\mathsf{M}_{\mathrm{add}}^{1,=}$ where the indices determine the programs of the machines. If a number i does not match with a program without order tests and oracle queries, then let $\mathcal{N}_i$ be a simple machine that halts in any case. Moreover, let $\bar{\mathcal{N}}_i$ enumerate all positive integers n belonging to the halting set of $\mathcal{N}_i$ by the procedure described in the proof of Lemma 1.
We define a new set $\mathcal{A} = \bigcup_{s \geq 0} \mathcal{A}_s$ in stages. Let $\mathcal{A}_0 = \emptyset$. For $s \geq 0$, let $\mathcal{A}_s$ be defined and

$$I_s = \{i \leq s \mid W_{i,s} \cap \mathcal{A}_s = \emptyset \;\&\; (\exists x \in W_{i,s})(2i < x \;\&\; (\forall j \leq i)(a(j,s) < x))\}$$

where, for any $i \leq s$,

- $a(i,s)$ is the greatest integer which is used in a query by the machine $\mathcal{M}_i^{\mathcal{A}_s}$ on i within the first s steps if $\mathcal{M}_i^{\mathcal{A}_s}(i) \downarrow^s$ (where $\downarrow^s$ means that the machine halts within s steps) and 0 if $\mathcal{M}_i^{\mathcal{A}_s}(i) \uparrow^s$ (otherwise) and
- $W_{i,s}$ is the set of the integers computed by $\bar{\mathcal{N}}_i$ on input s within s steps.

Then,

$$\mathcal{A}_{s+1} =_{\mathrm{df}} \begin{cases} \mathcal{A}_s & \text{if } I_s = \emptyset \\ \mathcal{A}_s \cup \{x_0\} & \text{otherwise} \end{cases}$$

where $x_0 = \min\{x \in W_{i_0,s} \mid 2i_0 < x \;\&\; (\forall j \leq i_0)(a(j,s) < x)\}$ for $i_0 = \min I_s$.

Note that, for the input i, the machine $\mathcal{M}_i^{\mathcal{A}_s}$ computes only integers. Consequently, any test $x \geq y$? executed by $\mathcal{M}_i^{\mathcal{A}_s}$ on i is decidable by a machine in $\mathsf{M}_{\mathrm{add}}^{1,=}$. Thus, the set $\mathcal{A}$ has the following properties.

(i) $\mathcal{A}$ is effectively enumerable by an additive machine in $\mathsf{M}_{\mathrm{add}}^{1,=}$.

(ii) $\mathbb{N} \setminus \mathcal{A}$ is infinite.

(iii) The intersection of $\mathcal{A}$ with any infinite halting set of a machine $\mathcal{N}_i$ is not empty.

(iv) $\mathbb{K}^{\mathcal{A}} \preceq \mathbb{K}^{\emptyset}$.

These properties of $\mathcal{A}$ hold by analogy with the classical result (compare e.g. [Kozen 2006]) since any additive machine in $\mathsf{M}_{\mathrm{add}}^{1,=}$ on positive integers can be simulated by some Turing machine and vice versa and the set of partial recursive functions agrees with the set of functions computable by Turing machines. Because of (i), (ii) and (iii) $\mathcal{A}$ is a *simple* set and because of (i) and (iv) $\mathcal{A}$ is a *low* set.

The conditions (i) and (ii) follow immediately from the definition of $\mathcal{A}$. The conditions (iii) and (iv) follow from the positive and the negative conditions for simplicity and lowness.

Conditions for simplicity:

(P_n) If $W_n = \bigcup_{i \geq 1} W_{n,i}$ is infinite, then $A \cap W_n \neq \emptyset$.

Conditions for lowness:

(N_n) If $\mathcal{M}_n^{A_t}(n) \downarrow^t$ for infinitely many t, then $\mathcal{M}_n^{A}(n) \downarrow$.

The latter conditions follow from the fact that $\mathcal{M}_i^{\mathcal{A}}(i) \downarrow$ implies $\mathcal{M}_i^{\mathcal{A}}(i) \downarrow^t$ for some t and $\mathcal{M}_i^{\mathcal{A}_s}(i) \downarrow^t$ for some s and t, and consequently $\mathcal{M}_i^{\mathcal{A}_{\max\{s,t\}}}(i) \downarrow^{\max\{s,t\}}$. The conditions P_n follow from the conditions N_n.

Remark. The goal of the priority method is to define the oracle by satisfying conditions of this kind where it is allowed to injure these conditions finitely many times. Note that, for these conditions, the priorities are determined by $P_1 \gg N_1 \gg P_2 \gg N_2 \gg P_3 \gg N_3 \gg P_4 \gg N_4 \gg \cdots$ where $H_1 \gg H_2$ means that the priority of H_1 is higher than the priority of H_2. In defining $\mathcal{A}_s$ it is allowed to injured conditions N_n although $n \leq s$, that means that $\mathcal{M}_n^{A_s}(n) \uparrow$ holds although we will get $\mathcal{M}_n^{A_t}(n) \downarrow^t$ for infinitely many t and $\mathcal{M}_n^{A}(n) \downarrow$. However, the conditions N_n are only injured in order to satisfy a condition of a higher priority. Moreover, it is easy to see that any condition can only be injured finitely many times. For more details see [Börger 1992] and [Kozen 2006].

Since the set of integers can be reduced to $\mathbb{H}_{\mathrm{add}}^{1,=}$, we have the following lemma by (i).

Lemma 2. $\mathcal{A} \preceq \mathbb{H}_{\text{add}}^{1,=}$.

For Turing machines we have $\Sigma_1^0 \cup \text{co}\Sigma_1^0 \subset \Sigma_2^0 \cap \text{co}\Sigma_2^0$ for the class Σ_1^0 of the semi-decidable sets and the class Σ_2^0 of sets being semi-decidable by machines using the classical Halting Problem $\mathbb{H}$ as oracle. Since $\mathbb{H}$ is complete for Σ_1^0 and the Halting Problem for Turing machines using $\mathbb{H}$ is complete for Σ_2^0, we have $\mathbb{K}^{\emptyset} \not\succeq \mathbb{K}^{\mathbb{K}^{\emptyset}}$. Thus, since $\mathbb{H}_{\text{add}}^{1,=} \preceq \mathcal{A}$ together with $\mathbb{K}^{\emptyset} \preceq \mathbb{H}_{\text{add}}^{1,=}$ would imply $\mathbb{K}^{\emptyset} \preceq \mathcal{A}$ and consequently $\mathbb{K}^{\mathbb{K}^{\emptyset}} \preceq \mathbb{K}^{\mathcal{A}} \preceq \mathbb{K}^{\emptyset}$ by (iv), we get the following.

Lemma 3. $\mathcal{A} \not\succeq \mathbb{H}_{\text{add}}^{1,=}$.

Moreover, we obtain the following lemma since the oracle set $\mathbb{Q}$ is not really helpful for deciding $\mathcal{A}$.

Lemma 4. $\mathcal{A} \not\preceq \mathbb{Q}$.

Proof. Let us assume that $\mathcal{A}$ is decidable by a machine in $\mathsf{M}_{\text{add}}^1(\mathbb{Q})$. Then, $(\mathbb{R} \setminus \mathcal{A}) \cap \mathbb{N}$ is semi-decidable by a machine $\mathcal{M} \in \mathsf{M}_{\text{add}}^1(\mathbb{Q})$. We will modify $\mathcal{M}$ in the following way. We extend the machine by instructions for enumerating all positive integers and compare it with the input at the beginning. If the input is not a positive integer, then the enumeration of integers will not be stopped. If the input is a positive integer, then the instructions of $\mathcal{M}$ can be simulated by a machine in $\mathsf{M}_{\text{add}}^{1,=}$ since all queries of $\mathcal{M}$ are answered in the positive and each order test can be simulated by means of few equality tests. Therefore we remove all oracle instructions in the program of $\mathcal{M}$ and replace the order tests by some instructions without order tests such that the resulting machine belongs to $\mathsf{M}_{\text{add}}^{1,=}$ and $(\mathbb{R} \setminus \mathcal{A}) \cap \mathbb{N}$ is semi-decidable by this machine. Since $\mathcal{A}$ satisfies the property (ii), there is an infinite W_j such that $(\mathbb{R} \setminus \mathcal{A}) \cap \mathbb{N} = W_j$. This contradicts the property (iii). Hence, the assumption is wrong. □

This implies the following.

Proposition 5. $\mathbb{Q} \not\succeq \mathbb{H}_{\text{add}}^{1,=}$.

4 $\mathbb{H}_{\text{add}}^{1,=}$ is Strictly Easier than $\mathbb{H}_{\text{add}}^{1}$

The aim of this section is to define an infinite hierarchy between $\mathbb{H}_{\text{add}}^{1,=}$ and $\mathbb{H}_{\text{add}}^{1}$. K. Meer and M. Ziegler showed that the power of algebraic numbers is great enough to give Turing degrees below the Halting Problem with respect to the BSS model of computation. Here, we will consider only the square roots of prime numbers in order to construct a hierarchy of Turing degrees with respect to the additive BSS machines without additional real parameters. We will show that the roots of different prime numbers yield, on one hand, an infinite number of incomparable Turing degrees and, on the other hand, a possibility to construct a hierarchy of Turing degrees such that the Halting Problem for additive machines without order and without real constants is easier than these degrees. In the proofs we will use the observation that the complement of a problem can be decided by using the problem itself as oracle.

Let $p_1 = 2, p_2 = 3, \dots$ be an enumeration of the prime numbers. Moreover let $\mathcal{K}_i$ be a machine recognizing $\mathbb{R} \setminus \{\sqrt{p_i}\}$ by checking, for any input x, the condition ($x < \frac{r}{q}$ and $\frac{r^2}{q^2} < p_i$) or ($x > \frac{r}{q}$ and $\frac{r^2}{q^2} > p_i$) for all enumerated $(r, q) \in \mathbb{N}^2$ until the condition is satisfied.
Thus, we can consider the following halting problems.

$$\begin{aligned} &\mathbb{P}_i = \{(1, x, \mathrm{code}(\mathcal{K}_i)) \in \mathbb{H}^1_{\mathrm{add}} \mid x \in \mathbb{R} \setminus \{\sqrt{p_i}\}\}, \\ &\mathbb{H}_i = \mathbb{H}^{1,=}_{\mathrm{add}} \cup \textstyle\bigcup_{j \leq i} \mathbb{P}_j. \end{aligned}$$

We assume that all machine $\mathcal{K}_j$ compute the prime number $p = p_j$ from j by executing the same procedure on j. To simplify matters we take instructions realizing the algorithm $k := 1$; $p := 2$; while $k < j$ $\{p := p + 1$; if $(\forall (t, s) \in \{2, \dots, p-1\}^2)(p \neq t \cdot s)$ then $k := k + 1;\}$. Then, the codes of these machines differ only in the code of the integers j. That means that the set of codes $\mathrm{code}(\mathcal{K}_j)$ and the set of pairs $(j, \mathrm{code}(\mathcal{K}_j))$ are decidable. Therefore, we get a hierarchy of the following form.

Lemma 6. $\mathbb{H}^{1,=}_{\mathrm{add}} \preceq \mathbb{H}_1 \preceq \mathbb{H}_2 \preceq \cdots \preceq \mathbb{H}^1_{\mathrm{add}}$.

The following lemmas are useful for showing that $\mathbb{H}_i$ is strictly easier than $\mathbb{H}_{i+1}$.

Lemma 7. *For any* $1 \leq k \leq i$, $\{\sqrt{p_k}\} \preceq \{\sqrt{p_1}, \dots, \sqrt{p_i}\}$.

Proof. Let $\mathcal{K}$ be the following machine. $\mathcal{K}$ queries the oracle $\{\sqrt{p_1}, \dots, \sqrt{p_i}\}$ whether the input x belongs to this oracle and outputs 0 if the answer is in the negative. Otherwise, let $I = \{1, \dots, i\}$ be the set of the indices of the first i prime numbers and let $\mathcal{K}$ simulate the machines $\mathcal{K}_1, \dots, \mathcal{K}_i$ simultaneously. For all enumerated (r, q), $\mathcal{K}$ checks $x < \frac{r}{q}$ and $\frac{r^2}{q^2} < p_j$ as well as $x > \frac{r}{q}$ and $\frac{r^2}{q^2} > p_j$ for any $j \in I$. If one of both conditions is satisfied for some j and, consequently, $x \neq \sqrt{p_j}$, then $I := I \setminus \{j\}$. If $k = j$, then the machine halts and the output is 0. Otherwise, the machine continues its operations. If the only member of I is k, then the output is 1. □

Relationships as $\{\sqrt{p_1}, \dots, \sqrt{p_i}\} \preceq \mathbb{R} \setminus \{\sqrt{p_1}, \dots, \sqrt{p_i}\} \preceq \{\sqrt{p_1}, \dots, \sqrt{p_i}\}$ are characteristic for the Turing reducibility relation. Moreover $\{\sqrt{p_j}\} \preceq \mathbb{P}_j \preceq \{\sqrt{p_j}\}$ is easy to see and we even have the following.

Lemma 8. *For any* $i \geq 1$,

$$\{\sqrt{p_1}, \dots, \sqrt{p_i}\} \preceq \textstyle\bigcup_{j \leq i} \mathbb{P}_j \preceq \{\sqrt{p_1}, \dots, \sqrt{p_i}\}.$$

Proof. $\mathbb{R} \setminus \{\sqrt{p_1}, \ldots, \sqrt{p_i}\} \preceq \bigcup_{j \leq i} \mathbb{P}_j$ holds since $x \in \mathbb{R} \setminus \{\sqrt{p_1}, \ldots, \sqrt{p_i}\}$ is satisfied if and only if $\{(1, x, \mathrm{code}(\mathcal{K}_1)), \ldots, (1, x, \mathrm{code}(\mathcal{K}_i))\} \subset \bigcup_{j \leq i} \mathbb{P}_j$ holds. On the other hand, the set $\{(1, \mathrm{code}(\mathcal{K}_1)), \ldots, (i, \mathrm{code}(\mathcal{K}_i))\}$ is decidable. Consequently, for any input (x_1, x_2, x_3) and for any $j \leq i$, we can check whether $x_1 = 1$, $x_3 = \mathrm{code}(\mathcal{K}_j)$ und $x_2 \neq \sqrt{p_j}$ by using the oracle $\{(1, \sqrt{p_1}), \ldots, (i, \sqrt{p_i})\}$. Thus, by Lemma 7 we also obtain $\bigcup_{j \leq i} \mathbb{P}_j \preceq \{\sqrt{p_1}, \ldots, \sqrt{p_i}\}$. □

Thus, the decidability of the set of codes $\mathrm{code}(\mathcal{K}_i)$ implies also the following.

Lemma 9. *Let $i \geq 1$ and $\mathcal{O} \subseteq \bigcup_{j \geq 1} \mathbb{R}^j$. If $\mathbb{H}_i$ is decidable by an oracle machine in $\mathsf{M}^1_{\mathrm{add}}(\mathcal{O})$, then the problems $\bigcup_{j \leq i} \mathbb{P}_j$ and $\{\sqrt{p_1}, \ldots, \sqrt{p_i}\}$ are also decidable by some machines in $\mathsf{M}^1_{\mathrm{add}}(\mathcal{O})$.*

Lemma 10. *Let $\mathcal{M} \in \mathsf{M}^1_{\mathrm{add}}(\mathbb{H}^{1,=}_{\mathrm{add}})$ be a machine deciding a problem $S \subseteq \mathbb{R}$. Then, there are $n, m \in \mathbb{N}^+$ such that $\mathcal{M}$ rejects the inputs $\sqrt{2}$ and $\frac{n}{m}\pi$ or $\mathcal{M}$ accepts both inputs.*

Proof. For the computation path of an $\mathcal{M} \in \mathsf{M}^1_{\mathrm{add}}(\mathbb{H}^{1,=}_{\mathrm{add}})$ on input $x \in \mathbb{R}$ there is a finite system of conditions of the form

$$k_\nu x + l_\nu \geq 0 \quad \text{and} \quad k_\mu x + l_\mu > 0, \tag{1}$$
$$(j, k_1 x + l_1, \ldots, k_j x + l_j, \mathrm{code}(\mathcal{N})) \in \mathbb{H}^{1,=}_{\mathrm{add}}, \tag{2}$$
$$(j, k_1 x + l_1, \ldots, k_j x + l_j, \mathrm{code}(\mathcal{N})) \notin \mathbb{H}^{1,=}_{\mathrm{add}} \tag{3}$$

($k_i, l_i \in \mathbb{Z}$, $\mathcal{N} \in \mathsf{M}^{1,=}_{\mathrm{add}}$) which is satisfied by an input x if and only if this path is traversed by $\mathcal{M}$ on x. Let us remark that the oracle queries which are not of the form

$$(j, k_1 x + l_1, \ldots, k_j x + l_j, \mathrm{code}(\mathcal{N})) \in \mathbb{H}^{1,=}_{\mathrm{add}}? \tag{4}$$

are always answered in the negative.

Let $x \in \mathbb{R}\setminus\mathbb{Q}$. Then, each equation $k_\lambda x + l_\lambda = 0$ and $k_\lambda x + l_\lambda = 1$, respectively, can only be satisfied if $k_\lambda = 0$. That means that any component of the tuple $\mathrm{code}(\mathcal{N}) \in \{0,1\}^\infty$ in (4) is determined by the constant function $f(x) = 0$ and $g(x) = 1$, respectively.

Moreover, every computation path of $\mathcal{N} \in \mathsf{M}^{1,=}_{\mathrm{add}}$ on an input of the form $(k_1 x + l_1, \ldots, k_j x + l_j)$ can be described by equations and inequalities of the form

$$k_\nu x + l_\nu = 0 \quad \text{and} \quad k_\mu x + l_\mu \neq 0.$$

Since, for any $n, m \in \mathbb{N}^+$, $\frac{n}{m}\pi$ and $\sqrt{2}$ satisfy the same equations of the form $k_\nu x + l_\nu = 0$, every $\mathcal{N} \in \mathsf{M}^{1,=}_{\mathrm{add}}$ halts on $(k_1 \frac{n}{m}\pi + l_1, \ldots, k_j \frac{n}{m}\pi + l_j)$ if and only if it halts on $(k_1\sqrt{2} + l_1, \ldots, k_j\sqrt{2} + l_j)$. Moreover, for any $\varepsilon > 0$, there are $n_\varepsilon, m_\varepsilon \in \mathbb{N}^+$ such that $|\frac{n_\varepsilon}{m_\varepsilon} - \frac{\sqrt{2}}{\pi}| < \frac{\varepsilon}{\pi}$. Thus, there are some $n_0, m_0 \in \mathbb{N}^+$ such that $\frac{n_0}{m_0}\pi$ and $\sqrt{2}$ satisfy the same system (1). □

Thus, we get the following corollaries.

Corollary 11. *The problem* $\{\sqrt{2}\}$ *is not decidable by a machine in* $\mathsf{M}^1_{\text{add}}(\mathbb{H}^{1,=}_{\text{add}})$.

Corollary 12. $\mathbb{H}^{1,=}_{\text{add}} \precneqq \mathbb{H}_1$.

Our next goal is to show that $\mathbb{H}_i$ is strictly easier than $\mathbb{H}_{i+1}$ for any $i \geq 1$. The following corollary results from Lemma 9 and Lemma 7.

Corollary 13. *For any* $i \geq 1$, *if* $\mathbb{H}_{i+1}$ *would be decidable by a machine in* $\mathsf{M}^1_{\text{add}}(\mathbb{H}_i)$, *then* $\mathbb{P}_{i+1}$ *as well as the problem* $\{\sqrt{p_{i+1}}\}$ *would also be decidable by machines in* $\mathsf{M}^1_{\text{add}}(\mathbb{H}_i)$.

Lemma 14. *Let* $\mathcal{M} \in \mathsf{M}^1_{\text{add}}(\mathbb{H}_i)$ *be a machine deciding a problem* $S \subseteq \mathbb{R}$. *Then, there are* $n, m \in \mathbb{N}^+$ *such that* $\mathcal{M}$ *rejects* $\sqrt{p_{i+1}}$ *and* $\frac{n}{m}\pi$ *or* $\mathcal{M}$ *accepts the both inputs.*

Proof. In analogy with the proof of Lemma 10, there are $n, m \in \mathbb{N}^+$ such that $\sqrt{p_{i+1}}$ and $\frac{n}{m}\pi$ satisfy same conditions of the form (1), (2), and (3). Moreover, we have $k_1\sqrt{p_{i+1}} + l_1 \neq \sqrt{p_j}$ for any $j \leq i$ and $k_1\frac{n}{m}\pi + l_1 \neq \sqrt{p_j}$ for any $j \leq i$ and any $m, n \in \mathbb{N}^+$. Thus, questions like $(1, k_1x + l_1, \text{code}(\mathcal{K}_j)) \in \mathbb{H}^{1,=}_{\text{add}}$? are answered in the positive. $\square$

From this lemma we deduce the following result.

Corollary 15. *For any* $i \geq 1$, *the problem* $\{\sqrt{p_{i+1}}\}$ *is not decidable by a machine in* $\mathsf{M}^1_{\text{add}}(\mathbb{H}_i)$.

Since these results are independent of the order of the prime numbers, we get incomparable Turing degrees for the strong Turing reduction without additional real parameters (apart from 0 and 1).

Corollary 16. *For any* $i, j \geq 1$ *where* $i \neq j$, *we have* $\mathbb{P}_i \not\preceq \mathbb{P}_j$ *and* $\mathbb{P}_j \not\preceq \mathbb{P}_i$.

By Corrolary 13 we moreover get the following.

Lemma 17. *For any* $i \geq 1$, $\mathbb{H}_i \precneqq \mathbb{H}_{i+1}$.

Consequently, we have the following.

Proposition 18. $\mathbb{H}^{1,=}_{\text{add}} \precneqq \mathbb{H}_1 \precneqq \mathbb{H}_2 \precneqq \cdots \precneqq \bigcup_{i\geq 1} \mathbb{H}_i \preceq \mathbb{H}^1_{\text{add}}$.

5 $\mathbb{H}^1_{\text{add}}$ is Strictly Easier than $\mathbb{H}_{\text{add}}$

Now we want to demonstrate the power of real machine constants. We will define a special sequence $(c_i)_{i\geq 1}$ of real numbers which, consequently, can be encoded by their indices i.

Recall that, for any BSS machine $\mathcal{M} \in \mathsf{M}_{\text{add}}$, $\text{code}(\mathcal{M})$ is the usual code of its program where any real constant is encoded by itself and the other single symbols are encoded by tuples in $\{0,1\}^k$ (for some k). For any machine $\mathcal{M} \in \mathsf{M}^1_{\text{add}}(\mathcal{O})$, let $k_{\mathcal{M}} = 2^{|\text{code}(\mathcal{M})|} + c_{\mathcal{M}}$ where $c_{\mathcal{M}}$ is the integer whose binary code matches with $\text{code}(\mathcal{M}) \in \{0,1\}^\infty$.

Now, we want to define a sequence of real numbers $c_1, c_2, c_3, \ldots \in]0,1]$ in stages which allow to evaluate some special halting problems of the form

$$\mathbb{H}_{\text{spec}}(\mathsf{M}^1_{\text{add}}(\mathcal{O})) = \{k_{\mathcal{M}} \in \mathbb{N}^+ \mid \mathcal{M} \in \mathsf{M}^1_{\text{add}}(\mathcal{O}) \ \& \ \mathcal{M}(k_{\mathcal{M}}) \downarrow\}.$$

Definition 19. Let $c_1 = 1$.
Stage $i \geq 2$: Let $c_i = \sum_{j=1}^{\infty} \alpha_j 10^{-j}$ where

$$\alpha_j = \begin{cases} 1 & \text{if } j \in \mathbb{H}_{\text{spec}}(\mathsf{M}^1_{\text{add}}(\mathbb{H}^{c_1,\ldots,c_{i-1}}_{\text{add}})) \\ 0 & \text{otherwise.} \end{cases}$$

For any BSS machine $\mathcal{M} \in \mathsf{M}_{\text{add}}$, let $\text{code}^{(i)}(\mathcal{M})$ be the sequence of the codes of the single symbols of the program of $\mathcal{M}$ where the single symbols, including the real numbers $c_1, c_2, \ldots, c_i$, are encoded by tuples in $\{0,1\}^{k+i}$ and any real constant in $\mathbb{R} \setminus \{c_1, c_2, \ldots, c_i\}$ is encoded by itself. In this way we get

$$\mathbb{H}^{(i)}_{\text{add}} = \bigcup_{n\geq 1}\{(n, \mathbf{x}, \text{code}^{(i)}(\mathcal{M})) \mid \mathbf{x} \in \mathbb{R}^n \ \& \ \mathcal{M} \in \mathsf{M}_{\text{add}} \ \& \ \mathcal{M}(\mathbf{x}) \downarrow\}.$$

Moreover, let $\mathsf{M}^{c_1,\ldots,c_i}_{\text{add}}$ be the set of the additive BSS machines using only the constants $c_1, \ldots, c_i$ and

$$\mathbb{H}^{c_1,\ldots,c_i}_{\text{add}} = \bigcup_{n\geq 1}\{(n, \mathbf{x}, \text{code}^{(i)}(\mathcal{M})) \in \mathbb{H}^{(i)}_{\text{add}} \mid \mathcal{M} \in \mathsf{M}^{c_1,\ldots,c_i}_{\text{add}}\}.$$

Lemma 20. *For any $i \geq 2$, $\mathbb{H}^1_{\text{add}} = \mathbb{H}^{c_1}_{\text{add}} \preceq \mathbb{H}^{c_1,c_2}_{\text{add}} \preceq \cdots \preceq \mathbb{H}^{c_1,\ldots,c_i}_{\text{add}} \preceq \mathbb{H}^{(i)}_{\text{add}}$.*

In order to show that $\mathbb{H}^{c_1,\ldots,c_{i-1}}_{\text{add}}$ is strictly easier than $\mathbb{H}^{c_1,\ldots,c_i}_{\text{add}}$, we consider the special Halting Problem for the machines in $\mathsf{M}^1_{\text{add}}(\mathbb{H}^{c_1,\ldots,c_{i-1}}_{\text{add}})$. We want to use the following known result.

Lemma 21. *For $\mathcal{O} \subseteq \bigcup_{i\geq 1} \mathbb{R}^i$, $\mathbb{H}_{\text{spec}}(\mathsf{M}^1_{\text{add}}(\mathcal{O}))$ is not decidable by a machine in $\mathsf{M}^1_{\text{add}}(\mathcal{O})$.*

Lemma 22. *For any $i \geq 2$, $\mathbb{H}_{\text{spec}}(\mathsf{M}^1_{\text{add}}(\mathbb{H}^{c_1,\ldots,c_{i-1}}_{\text{add}}))$ is decidable by a machine asking only the oracle $\mathbb{H}^{c_1,\ldots,c_i}_{\text{add}}$.*

Proof. Let $\mathcal{N}_i \in \mathsf{M}_{\text{add}}^{c_i}$ be a machine starting with $c := c_i$ and $j := 1$ on input $x \in \mathbb{R}$ and repeating $c := 10 \cdot c$; if $c > 1$ then {if $j = x$ then halt; $c := c - 1$; $j := j + 1$;} until the machine halts. Hence, by the definition of c_i, the machine $\mathcal{N}_i$ halts on $k_{\mathcal{M}}$ for $\mathcal{M} \in \mathsf{M}_{\text{add}}^1(\mathbb{H}_{\text{add}}^{c_1,\ldots,c_{i-1}})$ if and only if the $(k_{\mathcal{M}})^{\text{th}}$ digit of c_i after the point is an one and, consequently, if and only if $\mathcal{M}$ halts on $k_{\mathcal{M}}$. Thus, $\mathbb{H}_{\text{spec}}(\mathsf{M}_{\text{add}}^1(\mathbb{H}_{\text{add}}^{c_1,\ldots,c_{i-1}}))$ can be reduced to $\mathbb{H}_{\text{add}}^{c_1,\ldots,c_i}$. □

Therefore we have following lemma and by analogy with this we get also Lemma 24.

Lemma 23. *For any* $i \geq 1$, $\mathbb{H}_{\text{add}}^{c_1,\ldots,c_i} \precneqq \mathbb{H}_{\text{add}}^{c_1,\ldots,c_{i+1}}$.

Lemma 24. *For any* $i \geq 1$, $\mathbb{H}_{\text{add}}^{c_1,\ldots,c_i} \precneqq \mathbb{H}_{\text{add}}^{(i)}$.

Then, we get the following.

Proposition 25. *For any* $i \geq 2$, $\mathbb{H}_{\text{add}}^{c_1} \precneqq \mathbb{H}_{\text{add}}^{c_1,c_2} \precneqq \cdots \precneqq \mathbb{H}_{\text{add}}^{c_1,\ldots,c_i} \precneqq \mathbb{H}_{\text{add}}^{(i)}$.

Now, we symbolize the reductions executed by additive BSS machines using the constants $k_1, \ldots, k_j$ by $\preceq_{\text{add}}^{k_1,\ldots,k_j}$. $\mathcal{A} \equiv_{\text{add}}^{k_1,\ldots,k_j} \mathcal{B}$ means $\mathcal{A} \preceq_{\text{add}}^{k_1,\ldots,k_j} \mathcal{B}$ and $\mathcal{B} \preceq_{\text{add}}^{k_1,\ldots,k_j} \mathcal{A}$.

Proposition 26. *For any* $i \geq 1$, $\mathbb{H}_{\text{add}}^{(i)} \equiv_{\text{add}}^{c_{i+1}} \mathbb{H}_{\text{add}}^{(i+1)}$ *and* $\mathbb{H}_{\text{add}} \equiv_{\text{add}}^{c_1,\ldots,c_i} \mathbb{H}_{\text{add}}^{(i)}$.

$\bigcap_{i\geq 1} \mathbb{H}_{\text{add}}^{(i)}$ contains only codes of machines whose machine constants are encoded by themselves. Therefore, we get the first relation in the following proposition.

Proposition 27. $\bigcap_{i\geq 1} \mathbb{H}_{\text{add}}^{(i)} \preceq \mathbb{H}_{\text{add}} \preceq \bigcup_{i\geq 1} \mathbb{H}_{\text{add}}^{(i)}$.

Remark. We believe that $\mathbb{H}_{\text{add}} \not\preceq_{\text{add}} \bigcap_{i\geq 1} \mathbb{H}_{\text{add}}^{(i)}$ and $\bigcup_{i\geq 1} \mathbb{H}_{\text{add}}^{(i)} \not\preceq_{\text{add}} \mathbb{H}_{\text{add}}$ even if any additive machines with real constants can be used for reducing one problem to the other problem.

6 The Halting Problem for the Non-Deterministic Additive BSS Machines

The non-deterministic machines are able to guess arbitrary real numbers. For the corresponding Halting Problem $\mathbb{H}_{\text{add}}^{\text{ND}}$ holds

$$\mathbb{H}_{\text{add}}^{\text{ND}} \equiv \bigcup_{n\geq 1}\{(n, \mathbf{x}, \text{code}(\mathcal{M})) \mid \mathbf{x} \in \mathbb{R}^n \;\&\; \mathcal{M} \in \mathsf{M}_{\text{add}} \;\&\; (\exists \mathbf{y} \in \mathbb{R}^\infty)\mathcal{M}(\mathbf{x}, \mathbf{y}) \downarrow\}.$$

Since P. Koiran had shown in [Koiran 1994] that, for an input $\mathbf{x}$, a computation path of a non-deterministic additive machine determined by additional real guesses is also traversed on the same input $\mathbf{x}$ if the guesses are restricted to be linear combinations of the input values and the machine constants with rational coefficients. More precisely, for any computation path of length l, each of these guesses can be replaced by a linear combination of the input values and the machine constants whose rational coefficients can be computed in polynomial time (depending on l) if it is possible to guess zeros and ones. Thus, for any $l = 1, 2, \ldots$, a deterministic machine can decide whether a sequence of l instructions is a computation path traversed by a non-deterministic machine on an input. Consequently, the Halting Problem for the deterministic additive BSS machines is not easier than the Halting Problem $\mathbb{H}_{\text{add}}^{\text{ND}}$ for the non-deterministic additive BSS machines.

Proposition 28. $\mathbb{H}_{\text{add}} \preceq \mathbb{H}_{\text{add}}^{\text{ND}}$ *and* $\mathbb{H}_{\text{add}}^{\text{ND}} \preceq \mathbb{H}_{\text{add}}$.

7 Summary

These results can be summarized where $\rightarrow$ symbolizes the reduction $\precneq$, $\leftrightarrow$ symbolizes $\equiv$, and $\xleftrightarrow[k_1,\ldots,k_j]{}$ symbolizes $\equiv_{\text{add}}^{k_1,\ldots,k_j}$.

$$\begin{tikzcd}
\mathbb{H}_{\text{add}}^{\text{ND}} \arrow[d, leftrightarrow] & & & & & & \\
\mathbb{H}_{\text{add}} \arrow[r, leftrightarrow, "c_2"'] \arrow[d, leftrightarrow] & \mathbb{H}_{\text{add}}^{(2)} \arrow[r, leftrightarrow, "c_3"'] \arrow[d, leftrightarrow] & \mathbb{H}_{\text{add}}^{(3)} \arrow[r, leftrightarrow, "c_4"'] \arrow[d, leftrightarrow] & \cdots \arrow[r, leftrightarrow, "c_i"'] & \mathbb{H}_{\text{add}}^{(i)} \arrow[d, leftrightarrow] & \\
\mathbb{H}_{\text{add}}^{1} \arrow[r] & \mathbb{H}_{\text{add}}^{c_1,c_2} \arrow[r] & \mathbb{H}_{\text{add}}^{c_1,c_2,c_3} \arrow[r] & \cdots \arrow[r] & \mathbb{H}_{\text{add}}^{c_1,\ldots,c_i} \arrow[r] & \bigcup_{i\geq 1} \mathbb{H}_{\text{add}}^{c_1,\ldots,c_i}
\end{tikzcd}$$

$$\begin{tikzcd}
\vdots & & & & & & & \\
\mathbb{H}_2 = \mathbb{H}_{\text{add}}^{1,=} \cup \mathbb{P}_1 \cup \mathbb{P}_2 \arrow[u] & & & & & & & \\
\mathbb{H}_1 = \mathbb{H}_{\text{add}}^{1,=} \cup \mathbb{P}_1 \arrow[u] & & & & & & & \\
\mathbb{H}_{\text{add}}^{1,=} \arrow[u] & \cdots & \mathbb{L}_5 \arrow[l] & \mathbb{L}_4 \arrow[l] & \mathbb{L}_3 \arrow[l] & \mathbb{L}_2 \arrow[l] & \mathbb{Q} \arrow[l] & \emptyset \arrow[l] \\
\mathcal{A} \arrow[u] & & & & & & & \\
\emptyset \arrow[u] & & & & & & &
\end{tikzcd}$$

References

[Blum et al. 1998] Blum, L., F. Cucker, M. Shub, and S. Smale: "Complexity and Real Computation"; Springer-Verlag (1998).

[Blum et al. 1989] Blum, L., M. Shub, and S. Smale: “On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines”; Bulletin of the Amer. Math. Soc. 21 (1989), 1–46.

[Börger 1992] Börger, E.: ”Berechenbarkeit, Komplexität, Logik”; Vieweg (1992).

[Cucker and Koiran 1995] Cucker, F. and P. Koiran: ”Computing over the reals with addition and order: Higher complexity classes”; Journal of Complexity 11 (1995), 358–376.

[Friedberg 1957] Friedberg, R.M.: “Two recursively enumerable sets of incomparable degrees of unsolvability”; Proc. Natl. Acad. Sci. 43 (1957), 236–238.

[Gaßner 2008] Gaßner, C.: “A hierarchy below the Halting Problem for additive machines”; Theory of Computer Systems 43 (2008) (3), 464–470.

[Koiran 1994] Koiran, P.: ”Computing over the reals with addition and order”; Theoretical Computer Science 133 (1994), 35–47.

[Kozen 2006] Kozen, D. C.: “Theory of Computation”; Springer-Verlag (2006).

[Meer and Ziegler 2008] Meer, K., and M. Ziegler: “An explicit solution to Post’s Problem over the reals”; Journal of Complexity 24 (2008), 3–15.

[Meer and Ziegler 2006] Meer, K., M. Ziegler: “Uncomputability below the real Halting Problem”; LNCS 3988 (2006), 368–377.

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

[Soare 1987] Soare, R.I. “Recursively Enumerable Sets and Degrees”; Springer-Verlag (1987).