Math. Struct. in Comp. Science (2017), *vol.* 27, *pp.* 1386–1413. © Cambridge University Press 2016
doi:10.1017/S0960129516000116 First published online 20 June 2016

Computation over algebraic structures and a classification of undecidable problems

CHRISTINE GAßNER

Institut für Mathematik und Informatik, Ernst-Moritz-Arndt-Universität, Walther-Rathenau-Straße 47, D-17487, Greifswald, Germany
Email: gassnerc@uni-greifswald.de

Received 30 September 2015

We consider a uniform model of computation over algebraic structures resulting from a generalization of the Turing machine and the BSS model of computation. This model allows us to gain more insight into the reasons for unsolvability of algorithmic decision problems from different perspectives. For example, classes of undecidable problems can be introduced in several ways by analogy with the classical arithmetical hierarchy and, for many structures, the different definitions lead to different hierarchies of undecidable problems. Here, we will investigate some classes of a hierarchy that is defined semantically by our deterministic oracle machines and that can be syntactically characterized by formulas whose quantifiers range only over an enumerable set. Starting from machines over algebraic structures endowed with some relations and containing an infinite recursively enumerable sequence of individuals, we will also consider this hierarchy for BSS RAM's over the reals and some undecidable problems defined by algebraic properties of the real numbers.

1. Introduction

The arithmetical hierarchy is a classification of arithmetically definable sets that results from the complexity of formulas. These sets are defined by formulas in the language of first-order logic with the symbols $+$ for the addition on $\mathbb{N}$, R for a decidable relation and the non-negative integers as constants. This means that each of these sets contains those non-negative integers which satisfy a set of quantified formulas in the Peano arithmetic that is based on the Peano axioms. The *classes* Σ_n^0 *and* Π_n^0 *of the arithmetical hierarchy* are defined as follows (cf. Soare 1987, p. 60) where we use the symbol $\in$ in any formula $B \in \mathsf{C}$ (of our metalanguage) in order to express that a set B belongs to a class C.

The arithmetical hierarchy. Let $B \subseteq \mathbb{N}$.

i. Let B *be in* Σ_0^0 (and *in* Π_0^0) if and only if B is recursive (i. e. iff B is decidable by a Turing machine).
ii. For $n \geqslant 1$, let B *be in* Σ_n^0 if there is a recursive relation $R \subseteq \mathbb{N}^{n+1}$ such that

$$x \in B \Longleftrightarrow (\exists y_1)(\forall y_2)(\exists y_3)\cdots(Qy_n)R(x, y_1, \ldots, y_n)$$

holds, where Q stands for the symbol $\exists$ if n is odd and for the symbol $\forall$ if n is even.
iii. For $n \geqslant 1$, let B *be in* Π_n^0 if there is a recursive relation $R \subseteq \mathbb{N}^{n+1}$ such that

$$x \in B \Longleftrightarrow (\forall y_1)(\exists y_2)(\forall y_3)\cdots(Qy_n)R(x, y_1, \ldots, y_n)$$

holds, where Q stands for the symbol $\forall$ if n is odd and for the symbol $\exists$ if n is even.

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

iv. For $n \geqslant 1$, let B *be in* Δ_n^0 if $B \in \Sigma_n^0 \cap \Pi_n^0$.
v. B is *arithmetical* if $B \in \bigcup_{n\in\mathbb{N}}(\Sigma_n^0 \cup \Pi_n^0)$.

A characterization of this hierarchy by Turing degrees was summarized in Post's Theorem (cf. Soare 1987, p. 64). In this theorem, the empty set $\varnothing$ represents the class of all recursive sets and, for any $A \subseteq \mathbb{N}$, the set A' generated from A by the jump operator represents the class of all sets that are Turing equivalent to the Halting Problem for oracle Turing machines using the oracle set A. Moreover, we have $A^{(0)} = A$ and $A^{(n+1)} = (A^{(n)})'$ for all $n \geqslant 0$.

Post's theorem. For any $n \geqslant 0$, B belongs to Δ_{n+1}^0 if and only if B is decidable by some oracle Turing machine using $\varnothing^{(n)}$ as oracle set. Moreover, the following properties are equivalent:

i. B belongs to Σ_{n+1}^0.
ii. B is recursively enumerable by an oracle Turing machine using a set in Π_n^0 as oracle.
iii. B is recursively enumerable by an oracle Turing machine using a set in Σ_n^0 as oracle.
iv. B is recursively enumerable by an oracle Turing machine using $\varnothing^{(n)}$ as oracle set.
v. B is one–one reducible to $\varnothing^{(n+1)}$.

This close connection between the arithmetical hierarchy and the *jump hierarchy* $\varnothing, \varnothing', \varnothing'', \ldots$ results from the recursive enumerability of the integers and the fact that the elements of relations $R \subseteq \mathbb{N}^n$ and decision problems $P \subseteq \mathbb{N}^n$ such as several halting problems can be represented in a simple way by single integers. For example, any triple of an input, a number of steps and the code of a machine can be encoded by a single positive integer and so on. By Post's Theorem it is possible to introduce the arithmetical hierarchy in several ways. On the one hand, we can take the syntactic approach and define the classes of the hierarchy by quantified formulas. On the other hand, we can take the semantic approach and define the hierarchy in the context of oracle Turing machines. This kind of approach is based on an abstract machine model and an interpretation of the individual instructions. In both ways, we get the same hierarchy since every projection of a relation R is recursively enumerable by means of R as oracle. Both approaches are also useful for introducing various classes of decision problems with respect to the BSS RAM model over an arbitrary structure. One possibility is to define the classes of problems step-by-step by means of deterministic oracle machines, and alternatively we can define new classes syntactically by describing problems by quantified formulas with quantifiers ranging over the entire universe of the underlying structure. However, one gets, in general, different hierarchies when one moves beyond Turing machines and the underlying set of individuals and, thus, the range of the quantifiers are not enumerable. This is confirmed, for instance in Cucker (1992), where the considered model of computation is the BSS model over the ordered ring (or field) of all real numbers. Cucker defined two hierarchies $\Sigma_0, \Sigma_1, \ldots$ (on p. 381) and $\Sigma_0^{\mathrm{N}}, \Sigma_1^{\mathrm{N}}, \ldots$ (on p. 388) in the context of deterministic real oracle machines and non-deterministic real oracle machines, respectively, and he gave a characterization of these hierarchies by formulas in the framework of infinitary logic. As in the classical setting, the classes Σ_1 and Σ_1^{N} at the lowest level contain exactly the problems semi-decidable by BSS machines. For $n > 1$, the other classes Σ_n and Σ_n^{N} differ from each

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

other since the power of non-deterministic BSS oracle machines which can guess arbitrary real numbers is very closely related to the power of the quantifier alternation over the reals. Cucker used that the individual classes of the first hierarchy can be described by means of quantifier-free formulas of infinite length, whereas the analogous description for the classes of the second hierarchy requires quantified formulas (of infinite length). In accordance with this observation, there are analogies between the completeness of various sets at various levels in the classical arithmetical hierarchy and the completeness of certain problems in the BSS theory, but some of these problems are complete for classes of the first hierarchy and other problems are complete with respect to the second one.

We want to take the investigation of Cucker into account and give two definitions for hierarchies over arbitrary algebraic structures in analogy with the classical case such that, for the structure $(\mathbb{N}; \mathbb{N}; +; =)$, the definitions coincide with the usual descriptions of the arithmetical hierarchy. The first hierarchy will be defined with the help of oracle machines and the second one by means of quantified formulas.

Cucker refers in his work in particular to Moschovakis (1980). Some projective hierarchies have already been considered for several structures in Moschovakis (1969), too. However, there, the notion of computability over abstract structures is based on a machine-independent concept that – at least at first sight – shares only a few common features with our approach, especially since in contrast to our model Moschovakis considered partial multiple-valued functions as basic functions.

Our claims of a practicable concept of algorithm are closely linked to the expectations that certain operations and relations can be executed in a certain sequence of steps and that – on the base of this idea – the process can be represented by flow charts. Therefore, we choose a generalization of models which combines the characteristic properties of Real RAM models (that are known from computational geometry and introduced in Preparata and Shamos (1985)) with the requirement to process inputs of arbitrary length uniformly. Thus, on the one hand, our BSS-RAM model of computation reflects our intuitive idea of what an algorithm for processing real or complex numbers in mathematics is. And, on the other hand, this model of computation over algebraic structures endowed with non-trivial relations is suitable for analysing programs in high-level programming languages and designing efficient programs for processing inputs of arbitrary length by computers. Building on such expectations a series of models of computation over several structures were developed. However until now, the models were not systematically compared. At this point, we want to mention only two important early papers whose approach for specifying the notion of algorithm marked and mark, directly or indirectly, the further development. First, the model of computation over several structures introduced in Scott (1967) seems to be a forerunner of the development of abstract machines over arbitrary structures (cf. Börger 1992, p. 10). Second, the uniform BSS model of computation, defined by means of flow charts (cf. Blum *et al.* 1989, 1998), is extremely useful because of its simplicity and the ease of uniform treatment of all instances of an abstract type.

Uniform processing. Perhaps the most important feature of the BSS machines is that they can process inputs of arbitrary length uniformly. This was – as recently confirmed in Grädel (2007, p. 217) – a novelty at the time when the model was developed. This model

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

is helpful when investigating the complexity of mathematical problems even though its principal capabilities – also in comparison with other models over algebraic structures – had not really been studied systematically in all their scope and consequence. We think that the ability of any single machine to process inputs of *any* length uniformly is of the utmost importance. In particular, over the ring of reals, we thus have a model for describing the complexity of many practical relevant problems and, moreover, one can circumvent several problems which arise from the fact that tuples of real numbers cannot be effectively encoded by single reals (cf. Friedman and Mansfield 1992, Theorem 17). Hence, the uniformity is also a key feature of our more general model of computation over algebraic structures. It leads to a strengthening of certain classical methods on a higher level of abstraction, and it will also make clear which methods are suitable to provide analogues to the classical results and which ones are not.

Looking beyond this particular feature, our investigations – of course – are also related to a large number of other computability theories over structures, many of which are discussed, cited and compared, for instance, in Tucker and Zucker (2001) and Blum *et al.* (1998).

As is generally known, at an earlier stage the Turing machine and the class of partial recursive functions (produced by recursion and the μ-operator) and then also the register machines for processing non-negative integers were introduced in connection with the need to create a precise notion of computability. It turned out very quickly that, as many textbooks describe, these classical models of computation lead to the same class of computable functions. Apart from the properties that any tuple of integers can be represented by a single integer and that this integer can be represented and effectively encoded by a sequence of zeros and ones and (effectively) enumerated, most features of algorithms used in the proofs are of a very general nature and independent of the special underlying structure. More generally, it can be shown that in using stacks for storing certain parameters (the arguments of the recursively defined functions) recursions can be illustrated by flow charts. Conversely, every possible change of configurations of an abstract machine can be described by making suitable case distinctions, and every sequence of such changes can be defined by recursions, and moreover a stopping after reaching a termination condition can be expressed by means of the μ-operator (cf. Kleene 1952, for the classical case). For arbitrary structures, including the ring of reals, there are corresponding relationships between the BSS–RAM model and the safe-recursive functions considered in Bournez *et al.* (2003) and Bournez *et al.* (2006) and also a certain connection between the BSS–RAM model and the search-recursive functions that were already introduced by Moschovakis in 1969.

The articles just mentioned deal with machine-independent characterizations of computable functions by means of recursions and an operator similar to the μ-operator where in the papers of Bournez *et al.* the description of several complexity classes stands in the foreground. Moschovakis restricted his model – for the purpose of uniformity – to structures that are closed under a pairing operation so that any tuple of individuals can be represented by a single individual. Apart from this restriction, we can say that, for many algebraic structures, there is a close relationship between our model and his model and in particular between the ν-operator (defined on p. 449) and non-deterministic guessing.

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

However, in contrast to the BSS model, it is, for instance, possible to compute a function f with the function values $f(2) \to \sqrt{2}$ and $f(2) \to -\sqrt{2}$ by v within Moschovakis' model of computation over the ring of reals. To some degree Moschovakis leaves it to the reader whether by using the v-operator all suitable elements (e.g. all zeros of a polynomial) can be guessed – by analogy with the non-deterministic BSS machines – or whether a suitable element is searched along an arbitrary well ordering by an oracle. However, only the first case was pursued and, thus, all computable functions are multi-valued functions. In this respect, one can say that – with regards to more recent models – our model has more common properties with the models considered in Hemmerling (1998), Poizat (1995) and Asveld and Tucker (1982), in order to mention just a few, while other approaches such as the one in Tavana and Weihrauch (2011) have more affinity with the model of Moschovakis.

For the non-deterministic decision of problems, the mentioned differences play a relatively insignificant role and, therefore, we can confirm that, for the ordered ring of the reals, the projective hierarchy such as introduced in Moschovakis (1969) corresponds to the second hierarchy defined in accordance with the hierarchy $\Sigma_1^{\mathrm{N}}, \Sigma_2^{\mathrm{N}}, \ldots$ in Cucker (1992). That is also the reason why some closure properties formulated by Moschovakis for the individual classes of the projective hierarchy can be directly transferred to our second hierarchy, which we will not discuss in the present paper, and indirectly even to the first hierarchy, which is more closely examined below.

After a short definition of the model and the hierarchies, we will only consider structures $\mathcal{A} = (U; (d_j)_{j\in J_0}; (f_j)_{j\in J_1}; (R_j)_{j\in J_2}, =)$ containing a finite number of operations and relations and, to simplify matters, an infinite recursively enumerable subset of individuals that we denote by $\bar{\mathbb{N}} = \{\bar{0}, \bar{1}, \ldots\}$. For these structures, some classes of the first hierarchy, defined by deterministic oracle machines, will be studied.

Then, the problems in the classes of this hierarchy can also be described by quantified formulas of first-order logic over an extension of the structure $\mathcal{A}$ that may contain variables for all tuples of the individuals, symbols for the constants, the operations and the relations (predicates) of $\mathcal{A}$, and additional symbols R for decidable relations (or more precisely for decision problems $P \subseteq \bigcup_{i\in\mathbb{N}^+} U^i$) and $R_{\mathbb{N}}$. $R_{\mathbb{N}}$ is the symbol for a unary relation satisfied by the elements $\bar{0}, \bar{1}, \ldots$ that, in general, does not belong to $\mathcal{A}$. To simplify matters, in the following we do not distinguish between the symbols and their interpretations and we will also write $\vec{y} \in R$ instead of $R(\vec{y})$, for relations R over U, and $y \in \bar{\mathbb{N}}$ instead of $R_{\mathbb{N}}(y)$ or $y \in R_{\mathbb{N}}$. Moreover, in these formulas of our metalanguage, we write $(\vec{x}\,.\,\vec{y})$ and $(s\,.\,\vec{x})$ instead of $(x_1, \ldots, x_n, y_1, \ldots, y_m)$ and $(s, x_1, \ldots, x_n)$, respectively, and so on. If we need computable indices for sequences $X_0, X_1, \ldots$ of individuals, tuples, or functions, then we let $X_{\bar{i}} =_{\mathrm{df}} X_i$ and we will use several notations for such sequences: $(X_i)_{i\in\mathbb{N}}$, $(X_{\bar{i}})_{i\in\mathbb{N}}$ and $(X_i)_{i\in\bar{\mathbb{N}}}$. Sometimes, we write $\bar{i} \leqslant \bar{j}$ if $i \leqslant j$ and we use $\min\{\bar{i}, \bar{j}\} =_{\mathrm{df}} \overline{\min\{i, j\}}$ and so on.

The paper is arranged as follows: In the second section, we give a short overview of the used model of computation and the basic notions and concepts. In the third section, we introduce two hierarchies for the BSS RAM's in analogy with the arithmetical hierarchy and in the fourth section, for structures $\mathcal{A}$ with only a finite number of operations and relations, with all their individuals as constants, and with an effectively enumerable set, we present some complete problems for the classes of the first hierarchy. In section 5, we

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

transfer the Limit Lemma in order to characterize the class $\mathcal{A}\text{-}\Delta_2^0$ for these structures. In section 6, we discuss some interesting features of problems in the class $\mathcal{A}\text{-}\Delta_2^0$ for several structures $\mathcal{A}$ over the reals and we end the paper with a summary in section 7.

2. The model of computation

Let $\mathcal{A} = (U; (d_j)_{j\in J_0}; (f_j)_{j\in J_1}; (R_j)_{j\in J_2}, =)$ be any algebraic structure with the constants $d_j \in U$, the operations f_j, and the relations R_j. For $j \in J_1$, f_j is an operation of arity $n_{f_j} \geqslant 1$. For $j \in J_2$, R_j is a relation of arity n_{R_j}. Every *$\mathcal{A}$-machine* $\mathcal{M}$ (or *machine $\mathcal{M}$ over $\mathcal{A}$*) is equipped with registers $Z_1, Z_2, \ldots$ for single elements of U and with a finite number of registers $I_1, I_2, \ldots, I_{k_\mathcal{M}}$ for indices in $\mathbb{N}^+ =_{\mathrm{df}} \mathbb{N} \setminus \{0\}$. For each input $(x_1, \ldots, x_n)$ from the input space $U^\infty =_{\mathrm{df}} \bigcup_{i\in\mathbb{N}^+} U^i$ (U^∞ contains only tuples and it does not contain infinite sequences), before beginning the execution of the program, the sequence $x_1, \ldots, x_n, x_n, x_n, \ldots$ is assigned to the registers $Z_1, Z_2, \ldots$ and the index registers get the content n. After the input procedure the machine executes its *program* defined by a finite sequence of labelled instructions until the output instruction is reached. The labels may be positive integers and the allowed instructions are the following. The *computation*, *copy* and *branching* instructions have the form $Z_j := f_k(Z_{j_1}, \ldots, Z_{j_{n_{f_k}}})$, $Z_j := d_k$ $Z_{I_j} := Z_{I_k}$, and *if cond then goto l_1 else goto l_2* where *cond* can be of the form $R_k(Z_{j_1}, \ldots, Z_{j_{n_{R_k}}})$ or $Z_i = Z_j$ and l_1 and l_2 must be labels occurring elsewhere in the program. The index registers which we need for indirect addressing can be modified by $I_j := 1$ and $I_j := I_j + 1$, and their contents can be evaluated by means of branching instructions of the form *if $I_j = I_k$ then goto l_1 else goto l_2*. They are also used in the *copy instructions* $Z_{I_j} := Z_{I_k}$ and in the output procedure, and they are important for *oracle machines* using some oracle $\mathcal{O} \subseteq U^\infty$. To realize the evaluation of *queries* of the form $\vec{z} \in \mathcal{O}?$, we allow to execute oracle instructions of the form

$$\textit{if } (Z_1, \ldots, Z_{I_1}) \in \mathcal{O} \textit{ then goto } l_1 \textit{ else goto } l_2, \tag{1}$$

where the length of the tuple $\vec{z} = (z_1, \ldots, z_m)$ in a query is determined by the content m of the index register I_1. If the *output* instruction is reached and the content of I_1 is m, then the tuple of the values $z_1, \ldots, z_m$ of the first m registers $Z_1, \ldots, Z_m$ is the output and the machine halts. Thus, for any input $\vec{x} \in U^\infty$, a machine goes through a computation path that is uniquely described by a sequence of labels of the instructions of its program. We assume that every assignment of a (machine) constant to a register, every computation of a value of a function of $\mathcal{A}$, and every evaluation of a relation of $\mathcal{A}$ in the tests can be executed within one step. Therefore, let the *length* of any computation path be the number of instructions that need to be performed along this path. The input–output behaviour of an $\mathcal{A}$-machine determines a *partial function* $f : U^\infty \to U^\infty$ for which $f(\vec{x})$ is undefined, if the machine does not halt on input $\vec{x}$. Such a function is called *computable by an $\mathcal{A}$-machine* and we say also that it is *computable over $\mathcal{A}$*.

Now, let $\mathcal{A}$ be a fixed structure with a finite number of operations and relations and $\{d_j \mid j \in J_0\} = U$ and let $\bar{\mathbb{N}}$ be an infinite subset of U whose elements we denoted by $\bar{0}, \bar{1}, \ldots$. Hence, the members $\bar{0}, \bar{1}, \ldots$ denote constants of $\mathcal{A}$. Moreover, let the *function* succ, defined for all $n \in \mathbb{N}$ by $\mathrm{succ}(\bar{n}) = \overline{n+1}$ (and undefined otherwise), be computable

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

over $\mathcal{A}$. Consequently, the partial *function* pred given by $\text{pred}(\bar{n}) = \overline{n-1}$ for all $n \in \mathbb{N}^+$ and $\text{pred}(\bar{0}) = \bar{0}$ is also computable over $\mathcal{A}$ and the set of all successors $\text{succ}^i(\bar{n})$ of $\bar{n}$ is effectively enumerable. More precisely, we can say that $\bar{\mathbb{N}}$ and $\bar{\mathbb{N}}^\infty$ are effectively enumerable over $\mathcal{A}$ and they are semi-decidable by some $\mathcal{A}$-machines $\mathcal{M}_{\bar{\mathbb{N}}}$ and $\mathcal{M}_{\bar{\mathbb{N}}^\infty}$ in the sense of the following definitions.

A set $S \subseteq U^\infty$ is *recursively enumerable* (or *effectively enumerable*) *over* $\mathcal{A}$, if there are a function $f : U^\infty \to U^\infty$ that is computable by an $\mathcal{A}$-machine and a tuple of constants, $\vec{s} \in U^\infty$ such that $S = \{\vec{s}, f(\vec{s}), f^2(\vec{s}), \ldots\}$. A problem $P \subseteq U^\infty$ is *semi-decidable over* $\mathcal{A}$ (or *semi-decidable by an* $\mathcal{A}$*-machine*), if its partial characteristic function $\bar{\chi}_P : U^\infty \to \{\bar{1}\}$ is computable by an $\mathcal{A}$-machine, and the problem P is *decidable over* $\mathcal{A}$ if its characteristic function $\chi_P : U^\infty \to \{\bar{0}, \bar{1}\}$ is computable over $\mathcal{A}$. Therefore, the notions *recursive* and *recursively enumerable* used in the classical recursion theory mean *decidability* and *semi-decidability*, respectively, *over* $(\mathbb{N}; \mathbb{N}; +; =)$. Thus, as in the classical theory, the *halting sets* $H_{\mathcal{M}}$ of all $\mathcal{A}$-machines $\mathcal{M}$ are semi-decidable, and a set is decidable over $\mathcal{A}$ if it and its complement are semi-decidable over $\mathcal{A}$. On the other hand, for structures such as the ring of reals there are halting sets $S \subseteq \mathbb{R}^\infty$ (e.g. $S = \mathbb{R}$) which are not effectively enumerable since they are not countable. Moreover, if the universe U is not countable, then the set $\bar{\mathbb{N}} \subseteq U$ is not large enough to provide representations for all machines with all possible machine constants. Therefore, for any $\mathcal{A}$-machine $\mathcal{M}$, let its *code* $c_{\mathcal{M}} =_{\text{df}} \text{code}(\mathcal{M}) \in U^\infty$ be the sequence of the codes of all single symbols in the order they appear in the program of $\mathcal{M}$ and, in these codes, let any constant in U be represented by itself (after a suitable prefix). This prefix and the other single symbols of the programs are unambiguously encoded by tuples of k symbols in $\{\bar{0}, \bar{1}\}^k$ where k is only dependent on $\mathcal{A}$. Because, we assume that any program has only one input instruction and only one output instruction and it begins with the input instruction and it ends with this output instruction, the set of codes is prefix- and suffix-free, which means that there is no code of a machine that is a prefix or a suffix of any other valid machine code.

Let $\mathsf{M}_{\mathcal{A}}$ be the class of all $\mathcal{A}$-machines and let $\mathsf{M}_{\mathcal{A}}(\mathcal{O})$ be the class of all $\mathcal{A}$-machines using the oracle $\mathcal{O} \subseteq U^\infty$. From the unambiguousness of the representation of the programs, it follows.

Lemma 2.1. $\text{CODES}_{\mathcal{A}} =_{\text{df}} \{c_{\mathcal{M}} \mid \mathcal{M} \in \mathsf{M}_{\mathcal{A}}\}$ as well as the set of the codes of all oracle machines over $\mathcal{A}$ are prefix- and suffix-free and decidable over $\mathcal{A}$.

For these classes, we will also consider the halting problems defined by

$$\mathbb{H}_{\mathcal{A}}^{[\mathcal{O}]} = \{(\vec{x}\,.\,c_{\mathcal{M}}) \mid \vec{x} \in U^\infty \;\&\; \mathcal{M} \in \mathsf{M}_{\mathcal{A}}[(\mathcal{O})] \;\&\; \mathcal{M}(\vec{x})\downarrow\},$$

where we write $(\vec{x}\,.\,c_{\mathcal{M}})$ instead of $(x_1, \ldots, x_m, s_1, \ldots, s_l)$ if $\vec{x} = (x_1, \ldots, x_m)$ and $c_{\mathcal{M}} = \text{code}(\mathcal{M}) = (s_1, \ldots, s_l)$. $\mathcal{M}(\vec{x})\downarrow$ and $\mathcal{M}(\vec{x})\uparrow$ are used to express the facts that $\mathcal{M}$ halts on $\vec{x}$ or not. Since we consider only structures with a finite number of operations and relations, by Lemma 2.1 we get:

Lemma 2.2. $\mathbb{H}_{\mathcal{A}}^{[\mathcal{O}]}$ is semi-decidable by a machine in $\mathcal{M} \in \mathsf{M}_{\mathcal{A}}[(\mathcal{O})]$.

We denote the class of all problems $P \subseteq U^\infty$ that are semi-decidable (recognizable) over $\mathcal{A}$ by $\text{SDEC}_{\mathcal{A}}$ and the class of all problems that are semi-decidable by a machine in

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

$\mathsf{M}_{\mathcal{A}}(\mathcal{O})$ by $\mathrm{SDEC}_{\mathcal{A}}^{\mathcal{O}}$ where we say that a problem is *semi-decidable by an oracle machine* $\mathcal{M}$ if $\mathcal{M}$ computes the partial characteristic function of this problem. Moreover, let $\mathrm{SDEC}_{\mathcal{A}}^{\mathsf{C}} = \bigcup_{\mathcal{O}\in\mathsf{C}} \mathrm{SDEC}_{\mathcal{A}}^{\mathcal{O}}$ and let $\mathrm{DEC}_{\mathcal{A}}$, $\mathrm{DEC}_{\mathcal{A}}^{\mathcal{O}}$ and $\mathrm{DEC}_{\mathcal{A}}^{\mathsf{C}}$ be the corresponding classes of problems decidable without or by means of oracles. All further definitions and notations refer in any case to the computation over the fixed structure $\mathcal{A}$. Hence, by analogy with the classical case (cf. Soare 1987, p. 50 and p. 19), let the Turing reduction $\preceq_T$ and the one–one reduction $\preceq_1$ be defined by means of oracle machines over $\mathcal{A}$ and total injective reduction functions that can be computed by $\mathcal{A}$-machines, respectively. Consequently, it is easy to see that we have

$$\mathcal{O} \preceq_T \mathbb{H}_{\mathcal{A}}^{\mathcal{O}}, \quad \mathbb{H}_{\mathcal{A}} \preceq_T \mathbb{H}_{\mathcal{A}}^{\mathcal{O}}, \quad \mathcal{O} \preceq_1 \mathbb{H}_{\mathcal{A}}^{\mathcal{O}}, \quad \mathbb{H}_{\mathcal{A}} \preceq_1 \mathbb{H}_{\mathcal{A}}^{\mathcal{O}}.$$

3. A hierarchy of undecidable problems and Post's theorem for structures $\mathcal{A}$

We consider the following two hierarchies for every fixed $\mathcal{A}$ where n can be any non-negative integer (such that the classes Σ_n considered in Cucker 1992 coincide with the classes $(\mathbb{R};\mathbb{R};\cdot,+,-;\geqslant)$-$\Sigma_n^0$ and the classes Σ_n^{N} with the classes $(\mathbb{R};\mathbb{R};\cdot,+,-;\geqslant)$-$\Sigma_n^{\mathrm{ND}}$).

1. A hierarchy defined semantically by deterministic machines:

$$\begin{aligned}
\Sigma_0^0 &= \mathcal{A}\text{-}\Sigma_0^0 = \mathrm{DEC}_{\mathcal{A}},\\
\Pi_n^0 &= \mathcal{A}\text{-}\Pi_n^0 = \{U^\infty \setminus P \mid P \in \Sigma_n^0\},\\
\Delta_n^0 &= \mathcal{A}\text{-}\Delta_n^0 = \Sigma_n^0 \cap \Pi_n^0,\\
\Sigma_{n+1}^0 &= \mathcal{A}\text{-}\Sigma_{n+1}^0 = \mathrm{SDEC}_{\mathcal{A}}^{\Sigma_n^0}.
\end{aligned}$$

2. A hierarchy defined syntactically by formulas:

$$\begin{aligned}
[\mathcal{A}\text{-}]\Sigma_0^{\mathrm{ND}} &= \mathrm{DEC}_{\mathcal{A}},\\
[\mathcal{A}\text{-}]\Pi_n^{\mathrm{ND}} &= \{U^\infty \setminus P \mid P \in \Sigma_n^{\mathrm{ND}}\},\\
[\mathcal{A}\text{-}]\Delta_n^{\mathrm{ND}} &= \Sigma_n^{\mathrm{ND}} \cap \Pi_n^{\mathrm{ND}},\\
[\mathcal{A}\text{-}]\Sigma_{n+1}^{\mathrm{ND}} &= \{P \subseteq U^\infty \mid (\exists Q \in \Pi_n^{\mathrm{ND}})\forall \vec{x}(\vec{x} \in P \Longleftrightarrow (\exists \vec{y} \in U^\infty)((\vec{y}\,.\,\vec{x}) \in Q))\}.
\end{aligned}$$

The second hierarchy can be described semantically by non-deterministic machines. That is the reason that we use the superscripts ND and 0 to differentiate between the classes of both hierarchies. Moreover, by analogy with the arithmetical hierarchy, the first one can be characterized by a *jump hierarchy over* $\mathcal{A}$ as follows where

$$\mathbb{H}_{\mathcal{A}}^{(0)} =_{\mathrm{df}} \varnothing \text{ and } \mathbb{H}_{\mathcal{A}}^{(n)} =_{\mathrm{df}} \mathbb{H}_{\mathcal{A}}^{\mathbb{H}_{\mathcal{A}}^{(n-1)}} \text{ for } n > 0.$$

Theorem 3.1 (Post's Theorem for computation over $\mathcal{A}$). Let $\mathcal{A}$ be any algebraic structure with a finite number of operations and relations.

For any $n \geqslant 0$, the two following properties are equivalent.

- B belongs to Δ_{n+1}^0,
- B is decidable by an $\mathcal{A}$-machine using $\mathbb{H}_{\mathcal{A}}^{(n)}$ as oracle set.

Moreover, the following properties are equivalent.

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

i. B belongs to Σ^0_{n+1}.
ii. B is semi-decidable by an $\mathcal{A}$-machine using a set in Π^0_n as oracle set.
iii. B is semi-decidable by an $\mathcal{A}$-machine using a set in Σ^0_n as oracle set.
iv. B is semi-decidable by an $\mathcal{A}$-machine using $\mathbb{H}^{(n)}_{\mathcal{A}}$ as oracle set.
v. B is one–one reducible to $\mathbb{H}^{(n+1)}_{\mathcal{A}}$.

Obviously, by definition, we have $\mathbb{H}_{\mathcal{A}} = \mathbb{H}^{\varnothing}_{\mathcal{A}} = \mathbb{H}^{(1)}_{\mathcal{A}}$. Summarizing, the classes of the first hierarchy have the following form.

Corollary 3.1. For any algebraic structure $\mathcal{A}$ with a finite number of operations and relations and any $n \geqslant 1$, we have

$$\begin{aligned} (i)\ \Sigma^0_n &= \mathrm{SDEC}^{\mathbb{H}^{(n-1)}_{\mathcal{A}}}_{\mathcal{A}} = \{P \subseteq U^\infty \mid P \preceq_1 \mathbb{H}^{(n)}_{\mathcal{A}}\}, \\ (ii)\ \Pi^0_n &= \{P \subseteq U^\infty \mid P \preceq_1 U^\infty \setminus \mathbb{H}^{(n)}_{\mathcal{A}}\}, \\ (iii)\ \Delta^0_n &= \mathrm{DEC}^{\mathbb{H}^{(n-1)}_{\mathcal{A}}}_{\mathcal{A}} = \{P \subseteq U^\infty \mid P \preceq_T \mathbb{H}^{(n-1)}_{\mathcal{A}}\}. \end{aligned}$$

In Corollary 3.1, the relationship (ii) results directly from (i) since $A \preceq_1 B$ means that there is a computable total injective function $f : U^\infty \to U^\infty$ such that, for all $\vec{x} \in U^\infty$, $\vec{x} \in A$ holds if and only if $f(\vec{x}) \in B$ and, consequently, $\vec{x} \in U^\infty \setminus A$ holds if and only if $f(\vec{x}) \in U^\infty \setminus B$.

By the usual diagonalization techniques (cf. Corollary 8 in Gaßner 2009) the following can be shown.

Proposition 3.1. For any $n \geqslant 1$, $\mathbb{H}^{(n)}_{\mathcal{A}} \not\preceq_T \mathbb{H}^{(n-1)}_{\mathcal{A}}$.

Whereas Post's Theorem for BSS RAM's can be directly derived from the definitions, the proof of the following proposition requires more insights into the execution process of BSS RAM's. In general, the proofs of the classical theory can be only partially transferred directly for two reasons.

1. In the classical case, the syntactic and semantic techniques are often mixed in line with the following facts taken for granted.
   - The class of all semi-decidable sets agrees with the class of all enumerable sets in the classical case.
   - The set $\mathbb{N}$ is decidable in the classical case.
2. Here, we may only assume the following.
   - All sets enumerable over $\mathcal{A}$ are semi-decidable over $\mathcal{A}$.
   - The set $\bar{\mathbb{N}}$ is semi-decidable over $\mathcal{A}$.

Proposition 3.2 (A syntactical characterization of Σ^0_n).
For $n \geqslant 0$, $\Sigma^0_{n+1} = \{P \subseteq U^\infty \mid (\exists Q \in \Pi^0_n)\forall \vec{x}(\vec{x} \in P \iff (\exists s \in \bar{\mathbb{N}})((s\,.\,\vec{x}) \in Q))\}$.

Proof. At first, we want to show the inclusion $\supseteq$. Therefore, let

$$P = \{\vec{x} \in U^\infty \mid (\exists s \in \bar{\mathbb{N}})((s\,.\,\vec{x}) \in Q)\},$$

for some $Q \in \Pi^0_n$. For this Q, we have $(U^\infty \setminus Q) \preceq_1 \mathbb{H}^{(n)}_{\mathcal{A}}$ by Corollary 3.1, that means that there is a computable function $f : U^\infty \to U^\infty$ with $f(s\,.\,\vec{x}) \notin \mathbb{H}^{(n)}_{\mathcal{A}}$ if and only if $(s\,.\,\vec{x}) \in Q$.

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

This implies that P is semi-decidable by means of the following program of an oracle machine using $\mathbb{H}_{\mathcal{A}}^{(n)}$ as oracle.

1. Input $\vec{x} \in U^{\infty}$;
2. Let $s = \bar{0}$;
3. compute $f(s.\vec{x})$;
4. if $f(s.\vec{x}) \in \mathbb{H}_{\mathcal{A}}^{(n)}$ then $\{s := \mathrm{succ}(s)$; goto $3\}$ else halt.

Therefore, by the equivalence of i and iv in Theorem 3.1, we have $P \in \Sigma_{n+1}^{0}$.

Now, let us assume that P is any problem in Σ_{n+1}^{0} and let $\mathcal{M} \in \mathsf{M}_{\mathcal{A}}(\mathbb{H}_{\mathcal{A}}^{(n)})$ be an oracle machine semi-deciding P by means of the oracle $\mathbb{H}_{\mathcal{A}}^{(n)}$.

For $n = 0$, we can take the decidable set

$$Q = \{(\bar{t}.\vec{x}) \mid t \in \mathbb{N} \ \&\ \vec{x} \in U^{\infty} \ \&\ \mathcal{M}(\vec{x}) \downarrow^{t}\}$$

where, here and in the following, $\mathcal{L}(\vec{y}) \downarrow^{t}$ and $\mathcal{L}(\vec{y}) \uparrow^{t}$ (as well as $\mathcal{L}(\vec{y}) \downarrow^{\bar{t}}$ and $\mathcal{L}(\vec{y}) \uparrow^{\bar{t}}$) mean that the machine $\mathcal{L}$ halts on $\vec{y}$ within t steps or not. (Note, that $\mathcal{L}(\vec{y}) \uparrow^{0}$ holds in every case.)

Now, let $n \geqslant 1$. In order to define a suitable set Q for which $U^{\infty} \setminus Q$ is semi-decidable by a machine in $\mathsf{M}_{\mathcal{A}}(\mathbb{H}_{\mathcal{A}}^{(n-1)})$, we use the following observations.

1. If a machine $\mathcal{M}$ in $\mathsf{M}_{\mathcal{A}}(\mathcal{O})$ executes an oracle instruction of the form (1), then the next step of $\mathcal{M}$ is dependent on the content $\vec{z} = (z_1, \ldots, z_m)$ of the registers $Z_1, \ldots, Z_{I_1}$. If $\mathcal{O}$ is the oracle $\mathbb{H}_{\mathcal{A}}^{(n)}$ and $\vec{z}$ is not of the form $(\vec{y}.c_{\mathcal{L}})$ for any $\vec{y} \in U^{\infty}$ and any machine $\mathcal{L} \in \mathsf{M}_{\mathcal{A}}(\mathbb{H}_{\mathcal{A}}^{(n-1)})$, then the query $\vec{z} \in \mathbb{H}_{\mathcal{A}}^{(n)}?$ is answered in the negative. In the other cases, the answer of $(\vec{y}.c_{\mathcal{L}}) \in \mathbb{H}_{\mathcal{A}}^{(n)}?$ is dependent on whether $\mathcal{L}(\vec{y}) \downarrow$ or $\mathcal{L}(\vec{y}) \uparrow$ and consequently, by Proposition 3.1, the question is not always decidable by means of the oracle $\mathbb{H}_{\mathcal{A}}^{(n-1)}$.
2. It is possible to simulate a finite number of steps of $\mathcal{L} \in \mathsf{M}_{\mathcal{A}}(\mathbb{H}_{\mathcal{A}}^{(n-1)})$ on $\vec{y}$ by a universal machine using $\mathbb{H}_{\mathcal{A}}^{(n-1)}$ as oracle and, consequently, $\mathcal{L}(\vec{y}) \downarrow^{t}$ and $\mathcal{L}(\vec{y}) \uparrow^{t}$ are decidable by means of the oracle $\mathbb{H}_{\mathcal{A}}^{(n-1)}$.
3. For any $\vec{x}$, the input $\vec{x}$ is accepted by $\mathcal{M}$ if and only if there is a finite accepting computation path traversed by $\mathcal{M}$ for this input. Such a path of length l is uniquely determined and, consequently, there is a maximal and uniquely fixed sequence of all possible queries

$$(\vec{y}^{(1)}.c_{\mathcal{L}_1}) \in \mathbb{H}_{\mathcal{A}}^{(n)}?, (\vec{y}^{(2)}.c_{\mathcal{L}_2}) \in \mathbb{H}_{\mathcal{A}}^{(n)}?, \ldots, (\vec{y}^{(k)}.c_{\mathcal{L}_k}) \in \mathbb{H}_{\mathcal{A}}^{(n)}? \qquad (k < l) \qquad (2)$$

   asked during the computation of $\mathcal{M}$ on $\vec{x}$ in this order before the machine halts.
4. If $\mathcal{M}$ accepts $\vec{x}$, then the accepting behaviour of $\mathcal{M}$ on $\vec{x}$ results from the behaviour of several encoded machines that can be uniquely described by the feature $F_{\mathcal{M},\vec{x}} = (\bar{l}, \bar{\beta}_1, \ldots, \bar{\beta}_k)$ as follows. Let $\alpha_1, \alpha_2, \ldots, \alpha_k$ be the finite sequence of 0's and 1's representing the answers *yes* (= 1) and *no* (= 0) of the oracle queries in (2) in this order. For $i \leqslant k$, let $\beta_i = 0$ iff $\alpha_i = 0$ and let $\beta_i = t > 0$ iff the ith oracle query $(\vec{y}^{(i)}.c_{\mathcal{L}_i}) \in \mathbb{H}_{\mathcal{A}}^{(n)}?$ is answered in the positive and we have $\mathcal{L}_i(\vec{y}^{(i)}) \uparrow^{t-1}$ and $\mathcal{L}_i(\vec{y}^{(i)}) \downarrow^{t}$ for $\mathcal{L}_i \in \mathsf{M}_{\mathcal{A}}(\mathbb{H}_{\mathcal{A}}^{(n-1)})$.

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

5. Let, *cantor* : $\bar{\mathbb{N}}^{\infty} \to \bar{\mathbb{N}}$ be the computable function for determining uniquely the Cantor-number for any tuple in $\bar{\mathbb{N}}^{\infty}$ in the known way. Then, any tuple $F_{\mathcal{M},\vec{x}} = (\bar{l}, \beta_1, \ldots, \beta_k)$ can be computed from its Cantor-number $cantor(\bar{l}, \beta_1, \ldots, \beta_k)$.
6. There is a machine $\bar{\mathcal{L}} \in \mathsf{M}_{\mathcal{A}}(\mathbb{H}_{\mathcal{A}}^{(n-1)})$ that, on inputs $(j\,.\,\vec{y}^{(i_1)}\,.\,c_{\mathcal{L}_{i_1}}\,.\,\ldots\,.\,\vec{y}^{(i_j)}\,.\,c_{\mathcal{L}_{i_j}}) \in U^{\infty}$, can simultaneously simulate all these $\mathcal{L}_i$'s on $\vec{y}^{(i)}$ with the help of the oracle $\mathbb{H}_{\mathcal{A}}^{(n-1)}$ and halts if and only if an output instruction is simulated.
7. The behaviour of $\mathcal{M}$ on $\vec{x}$ can be checked by a machine $\mathcal{N}_{\mathcal{M}} \in \mathsf{M}_{\mathcal{A}}(\mathbb{H}_{\mathcal{A}}^{(n-1)})$ in the following way so that $\mathcal{N}_{\mathcal{M}}$ does not halt on $(s\,.\,\vec{x}) = (cantor(l, b_1, \ldots, b_k)\,.\,\vec{x})$ if and only if $\mathcal{M}$ accepts the input $\vec{x}$ and $(l, b_1, \ldots, b_k)$ is the tuple $F_{\mathcal{M},\vec{x}}$ that provides the background information about the oracle queries made by $\mathcal{M}$ on $\vec{x}$.

   – On input $(s\,.\,\vec{x})$, first $\mathcal{N}_{\mathcal{M}}$ simulates $\mathcal{M}$ on $\vec{x}$, with one restriction:

     – Instead of the queries $(\vec{y}^{(i)}\,.\,c_{\mathcal{L}_i}) \in \mathbb{H}_{\mathcal{A}}^{(n)}?$, $\mathcal{N}_{\mathcal{M}}$ checks whether $\mathcal{L}_i(\vec{y}^{(i)}) \downarrow^{b_i}$ and $\mathcal{L}_i(\vec{y}^{(i)}) \uparrow^{\mathrm{pred}(b_i)}$.

   – If the output instruction of $\mathcal{M}$ is not reached after considering l instructions of $\mathcal{M}$, then $\mathcal{N}_{\mathcal{M}}$ halts.

   – If the output instruction of $\mathcal{M}$ is reached after the 'pseudo'-simulation of at most l instructions of $\mathcal{M}$, then $\mathcal{N}_{\mathcal{M}}$ simultaneously simulates those of the machines $\mathcal{L}_i$ on $\vec{y}^{(i)}$ for which $b_i = \bar{0}$ is given and halts if and only if the output instruction of one of these machines $\mathcal{L}_i$ is simulated.

In view of these considerations, we can give a program for $\mathcal{N}_{\mathcal{M}}$ such that $\mathcal{M}$ accepts $\vec{x}$ if and only if there is an $s \in \bar{\mathbb{N}}$ such that $\mathcal{N}_{\mathcal{M}}$ does not halt on $(s\,.\,\vec{x})$. Let $\mathcal{N}_{\mathcal{M}} \in \mathsf{M}_{\mathcal{A}}(\mathbb{H}_{\mathcal{A}}^{(n-1)})$ be the following machine.

1. Input $(s\,.\,\vec{x})$;
2. check (i. e. semi-decide) whether $s \in \bar{\mathbb{N}}$ holds by executing the program of $\mathcal{M}_{\bar{\mathbb{N}}}$;
3. write down the code $c_{\mathcal{M}}$ of $\mathcal{M}$;
4. compute $l, b_1, \ldots, b_k$ such that $s = cantor(l, b_1, \ldots, b_k)$;
5. $j := \bar{0}$; let $\vec{z}$ be the empty tuple;
6. simulate – apart from the oracle instructions given in the form

$$\textit{if } (\vec{y}^{(i)}\,.\,c_{\mathcal{L}_i}) \in \mathbb{H}_{\mathcal{A}}^{(n)} \textit{ then goto } l_1^{(i)} \textit{ else goto } l_2^{(i)} \tag{3}$$

   – the program of $\mathcal{M}$ on $\vec{x}$ and evaluate the condition in (3) as follows:

   – if $b_i \neq \bar{0}$, $\mathcal{L}_i(\vec{y}^{(i)}) \uparrow^{\mathrm{pred}(b_i)}$ and $\mathcal{L}_i(\vec{y}^{(i)}) \downarrow^{b_i}$, then goto instruction $l_1^{(i)}$ in $c_{\mathcal{M}}$;

   – if $b_i \neq \bar{0}$ and $(\mathcal{L}_i(\vec{y}^{(i)}) \downarrow^{\mathrm{pred}(b_i)}$ or $\mathcal{L}_i(\vec{y}^{(i)}) \uparrow^{b_i})$, then halt;

   – if $b_i = \bar{0}$, then $\{j := \mathrm{succ}(j)$; add $(\vec{y}^{(i)}.\mathcal{L}_i)$ to $\vec{z}$; $i_j := i$; goto instruction $l_2^{(i)}$ in $c_{\mathcal{M}}\}$.

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

7. if the output instruction of $\mathcal{M}$ is not reached after the simulation and evaluation of l instructions of $\mathcal{M}$,
   - then halt;
   - else simulate $\bar{\mathcal{L}} \in \mathsf{M}_{\mathcal{A}}(\mathbb{H}_{\mathcal{A}}^{(n-1)})$ on $(j.\vec{z})$ for $\vec{z} = (\vec{y}^{(i_1)}.c_{\mathcal{L}_{i_1}}.\ldots.\vec{y}^{(i_j)}.c_{\mathcal{L}_{i_j}})$ and halt if the output instruction of one of these machines is simulated (if no output instruction is simulated then do not halt).

Consequently, we have $\mathcal{N}_{\mathcal{M}} \in \mathsf{M}_{\mathcal{A}}(\mathbb{H}_{\mathcal{A}}^{(n-1)})$ and, for its halting set $H_{\mathcal{N}_{\mathcal{M}}}$, this implies $H_{\mathcal{N}_{\mathcal{M}}} \leq_1 \mathbb{H}_{\mathcal{A}}^{(n)}$ and $H_{\mathcal{N}_{\mathcal{M}}} \in \Sigma_n^0$. Moreover, for $Q = U^\infty \setminus H_{\mathcal{N}_{\mathcal{M}}} \in \Pi_n^0$, we have the following.

- $\vec{x} \in P$ if and only if $\mathcal{M}$ halts on $\vec{x}$ if and only if there is an $s \in \bar{\mathbb{N}}$ with $(s.\vec{x}) \in Q$.

□

Corollary 3.2 (A syntactical characterization of Π_n^0).
For $n \geqslant 0$, $\Pi_{n+1}^0 = \{P \subseteq U^\infty \mid (\exists Q \in \Sigma_n^0)\forall\vec{x}(\vec{x} \in P \iff (\forall s \in \bar{\mathbb{N}})((s.\vec{x}) \in Q))\}$.

This allows us to show the following.

Proposition 3.3. For any algebraic structure $\mathcal{A}$ with a finite number of operations and relations, we have a proper hierarchy

$$\Sigma_0^0 \subset \Sigma_1^0 \subset \Sigma_2^0 \subset \cdots.$$

Proof. Let $n \geqslant 1$. By Proposition 3.1, $\mathbb{H}_{\mathcal{A}}^{(n)} \in \Sigma_n^0$ is not decidable by an $\mathcal{A}$-machine using $\mathbb{H}_{\mathcal{A}}^{(n-1)}$ as oracle set and, thus, we have $\mathbb{H}_{\mathcal{A}}^{(n)} \notin \Delta_n^0$ and, consequently,

$$\mathbb{H}_{\mathcal{A}}^{(n)} \notin \Pi_n^0. \tag{4}$$

Now, let us assume that we had $\mathbb{H}_{\mathcal{A}}^{(n)} \in \Sigma_{n-1}^0$. Then, we would also have $Q = \{(s.\vec{x}) \mid (s,\vec{x}) \in \bar{\mathbb{N}} \times \mathbb{H}_{\mathcal{A}}^{(n)}\} \in \Sigma_{n-1}^0$ and thus $\mathbb{H}_{\mathcal{A}}^{(n)} \in \Pi_n^0$ in contradiction to the fact (4). □

Summarizing, we can say that this section has established a distinction between the hierarchies defined above: one is defined by quantifiers over U, and the other is characterized by quantifiers over $\bar{\mathbb{N}}$.

4. Some complete problems

Let $\mathcal{A}$ be again a structure with only a finite number of operations and relations and let all individuals be constants.

The problem whether any given machine computes a total function $f : \mathbb{N} \to \mathbb{N}$ is known as Π_2^0-complete with respect to the classical recursion theory (cf. Soare 1987, p. 66). Cucker transferred this result to the BSS model and showed that the problem whether any given deterministic BSS machine computes a total function $f : \mathbb{R}^\infty \to \mathbb{R}^\infty$ is Π_2^N-complete for the mentioned hierarchy defined by means of non-deterministic oracle machines. For our purpose it is useful to start with the following complete problem for the fixed structure $\mathcal{A}$ where we call a function $f : U^\infty \to U^\infty$ *total on* $\bar{\mathbb{N}}^\infty$ if $f(\vec{x}) \downarrow$ holds for all $\vec{x} \in \bar{\mathbb{N}}^\infty$ (i.e. if $f(\vec{x})$ is defined for all $\vec{x} \in \bar{\mathbb{N}}^\infty$).

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

Proposition 4.1. The problem

$$\mathrm{TOTAL}_{\mathbb{N}} = \{c_{\mathcal{N}} \mid \mathcal{N} \in \mathsf{M}_{\mathcal{A}} \ \&\ \mathcal{N} \text{ computes a total function on } \bar{\mathbb{N}}^{\infty}\}$$

is Π_2^0-complete.

Proof. The problem $\mathrm{TOTAL}_{\mathbb{N}}$ is in Π_2^0 since $\bar{\mathbb{N}}^{\infty}$ is effectively enumerable and, thus, there is a machine in $\mathsf{M}_{\mathcal{A}}(\mathbb{H}_{\mathcal{A}})$ that accepts any input $c_{\mathcal{N}} \in \mathrm{CODES}_{\mathcal{A}}$ if and only $c_{\mathcal{N}} \notin \mathrm{TOTAL}_{\mathbb{N}}$. By Lemma 2.1, the latter means that the complement $U^{\infty} \setminus \mathrm{TOTAL}_{\mathbb{N}}$ is semi-decidable by a machine in $\mathsf{M}_{\mathcal{A}}(\mathbb{H}_{\mathcal{A}})$. Let $P \subseteq U^{\infty}$ and $P \in \Pi_2^0$. By definition its complement $R = U^{\infty} \setminus P$ is semi-decidable by some machine $\mathcal{M} \in \mathsf{M}_{\mathcal{A}}(\mathbb{H}_{\mathcal{A}})$. We will give, for any $\vec{x}$, a machine $\mathcal{N}_{\vec{x},\mathcal{M}}$ such that R can be reduced to the complement of $\mathrm{TOTAL}_{\mathbb{N}}$ by a computable mapping $\vec{x} \mapsto c_{\mathcal{N}_{\vec{x},\mathcal{M}}}$, and then, as a direct consequence we will get that P can be reduced to $\mathrm{TOTAL}_{\mathbb{N}}$.

For any $\vec{x}$, let $\mathcal{N}_{\vec{x},\mathcal{M}} \in \mathsf{M}_{\mathcal{A}}$ be the machine executing the following program where $\mathcal{N}_{\mathcal{M}}$ is the machine that was considered in the proof of Proposition 3.2.

1. Input $(l, b_1, \ldots, b_k) \in U^{k+1}$;
2. check (i. e. semi-decide) whether $(l, b_1, \ldots, b_k) \in \bar{\mathbb{N}}^{k+1}$ holds by executing the program of $\mathcal{M}_{\bar{\mathbb{N}}^{\infty}}$;
3. execute the program of $\mathcal{N}_{\mathcal{M}}$ on the input $(\mathit{cantor}(l, b_1, \ldots, b_k) \,.\, \vec{x})$.

Note, that such a machine $\mathcal{N}_{\vec{x},\mathcal{M}}$ can use $x_1, x_2, \ldots$ as machine constants since all individuals in $\mathcal{A}$ are permitted to be machine constants. $\mathcal{N}_{\vec{x},\mathcal{M}}$ halts only if the input is from $\bar{\mathbb{N}}^{\infty}$ and $\mathcal{N}_{\mathcal{M}}$ halts on $(\mathit{cantor}(l, b_1, \ldots, b_k) \,.\, \vec{x})$. Thus, $\vec{x}$ belongs to R and is accepted by $\mathcal{M}$ if and only if there is an input $(l, b_1, \ldots, b_k) \in \bar{\mathbb{N}}^{k+1}$ for which $\mathcal{N}_{\vec{x},\mathcal{M}}$ does not halt. That means that $\vec{x} \in R$ iff $c_{\mathcal{N}_{\vec{x},\mathcal{M}}} \notin \mathrm{TOTAL}_{\mathbb{N}}$. □

From this, we can derive further complete problems where $H_{\mathcal{M}}$ is the halting set of $\mathcal{M}$.

Proposition 4.2. The problems

$$\mathrm{INCL1} = \{(c_{\mathcal{M}} \,.\, c_{\mathcal{N}}) \mid \mathcal{M}, \mathcal{N} \in \mathsf{M}_{\mathcal{A}} \ \&\ H_{\mathcal{M}} \cap \bar{\mathbb{N}}^{\infty} \subseteq H_{\mathcal{N}} \cap \bar{\mathbb{N}}^{\infty}\}$$

and

$$\mathrm{INCL2} = \{(c_{\mathcal{M}} \,.\, c_{\mathcal{N}}) \mid (\mathcal{M}, \mathcal{N}) \in \mathsf{M}_{\mathcal{A}}(\mathbb{H}_{\mathcal{A}}) \times \mathsf{M}_{\mathcal{A}} \ \&\ H_{\mathcal{M}} \cap \bar{\mathbb{N}}^{\infty} \subseteq H_{\mathcal{N}} \cap \bar{\mathbb{N}}^{\infty}\}$$

are Π_2^0-complete.

Proof.

a. INCL2 is in Π_2^0 since its complement is semi-decidable by a machine in $\mathsf{M}_{\mathcal{A}}(\mathbb{H}_{\mathcal{A}})$ executing the following algorithm where $\vec{n}^{(0)}, \vec{n}^{(1)}, \ldots$ is an effective enumeration of all tuples in $\bar{\mathbb{N}}^{\infty}$ such that $\vec{n}^{(i)} = \vec{n}^{(j)}$ implies $i = j$ for all $i, j \in \mathbb{N}$.
   1. Input $\vec{x} \in U^{\infty}$;
   2. decide whether $\vec{x} = (c_{\mathcal{M}} \,.\, c_{\mathcal{N}})$ for some $(\mathcal{M}, \mathcal{N}) \in \mathsf{M}_{\mathcal{A}}(\mathbb{H}_{\mathcal{A}}) \times \mathsf{M}_{\mathcal{A}}$; if not, then halt;
   3. enumerate all tuples $(t \,.\, \vec{n}^{(j)})$ and do, for any $(t \,.\, \vec{n}^{(j)})$, the following: decide, by using $\mathbb{H}_{\mathcal{A}}$, whether $\mathcal{M}(\vec{n}^{(j)}) \downarrow^t$

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

– if ($\mathcal{M}(\vec{n}^{(j)})\downarrow^t$ and $\mathcal{N}(\vec{n}^{(j)})\downarrow$) or $\mathcal{M}(\vec{n}^{(j)})\uparrow^t$ then {goto 3 and take the next $(t\,.\,\vec{n}^{(j)})$};
– if $\mathcal{M}(\vec{n}^{(j)})\downarrow^t$ and $\mathcal{N}(\vec{n}^{(j)})\uparrow$ then halt.

Consequently, INCL1 is also in Π_2^0.

b. Let $\mathcal{M}_0$ be a machine with $H_{\mathcal{M}_0} = \bar{\mathbb{N}}^\infty$. $\mathrm{TOTAL}_{\mathbb{N}} \preceq_1$ INCL1 and $\mathrm{TOTAL}_{\mathbb{N}} \preceq_1$ INCL2 hold since the reductions to these problems are possible by $\vec{x} \mapsto (c_{\mathcal{M}_0}\,.\,\vec{x})$.

□

As the following proof shows, the Π_2^0-completeness of $\mathrm{TOTAL}_{\mathbb{N}}$ implies the Σ_2^0-completeness of $\mathrm{FIN}_{\mathbb{N}}$.

Proposition 4.3. The problem

$$\mathrm{FIN}_{\mathbb{N}} = \{c_{\mathcal{M}} \mid \mathcal{M} \in \mathsf{M}_{\mathcal{A}} \;\&\; H_{\mathcal{M}} \cap \bar{\mathbb{N}}^\infty \text{ is finite}\}$$

is Σ_2^0-complete.

Proof. Let $\vec{n}^{(0)}, \vec{n}^{(1)}, \ldots$ be again an effective enumeration of all tuples in $\bar{\mathbb{N}}^\infty$ such that $\vec{n}^{(i)} = \vec{n}^{(j)}$ implies $i = j$.

First, we want to give a reduction of $U^\infty \setminus \mathrm{FIN}_{\mathbb{N}}$ to $\mathrm{TOTAL}_{\mathbb{N}}$ which implies that $U^\infty \setminus \mathrm{FIN}_{\mathbb{N}} \in \Pi_2^0$. Let $\mathcal{M} \in \mathsf{M}_{\mathcal{A}}$. Obviously, $c_{\mathcal{M}} \notin \mathrm{FIN}_{\mathbb{N}}$ if, for every i, there is a $j > i$ with $\mathcal{M}(\vec{n}^{(j)})\downarrow$. Let $\mathcal{N}_{\mathcal{M}} \in \mathsf{M}_{\mathcal{A}}$ be the machine checking the behaviour of $\mathcal{M}$ by executing the following program.

1. Input $\vec{x} = (x_1, x_2, \ldots, x_n) \in U^\infty$;
2. check (i.e. semi-decide) whether $\vec{x} \in \bar{\mathbb{N}}^\infty$ holds by executing the program of $\mathcal{M}_{\bar{\mathbb{N}}^\infty}$;
3. $s := x_1$;
4. $s := \mathrm{succ}(s)$;
5. if $(\mathcal{M}(\vec{n}^{(j)})\downarrow^s$ for some $j = \mathrm{succ}(x_1), \ldots, s)$ then halt else goto 4.

The machine $\mathcal{N}_{\mathcal{M}}$ halts on the input $\vec{x}$ if and only if $\vec{x}$ belongs to $\bar{\mathbb{N}}^\infty$ and there is a successor j of x_1 such that $\mathcal{M}(\vec{n}^{(j)})\downarrow$. Consequently, $c_{\mathcal{M}} \notin \mathrm{FIN}_{\mathbb{N}}$ if and only if $c_{\mathcal{N}_{\mathcal{M}}} \in \mathrm{TOTAL}_{\mathbb{N}}$. That means that $U^\infty \setminus \mathrm{FIN}_{\mathbb{N}}$ can be reduced to $\mathrm{TOTAL}_{\mathbb{N}}$ and $\mathrm{FIN}_{\mathbb{N}}$ can be reduced to $U^\infty \setminus \mathrm{TOTAL}_{\mathbb{N}}$, and in both cases the reduction is possible by means of a computable mapping with $c_{\mathcal{M}} \mapsto c_{\mathcal{N}_{\mathcal{M}}}$ and $\vec{x} \mapsto \vec{x}$ for all $\vec{x} \in U^\infty \setminus \mathrm{CODES}_{\mathcal{A}}$. Thus, $\mathrm{FIN}_{\mathbb{N}}$ is in Σ_2^0.

Second, we know, by Proposition 4.1, that $U^\infty \setminus \mathrm{TOTAL}_{\mathbb{N}}$ is Σ_2^0-complete. Therefore, we want to give a mapping $c_{\mathcal{M}} \mapsto c_{\mathcal{N}_{\mathcal{M}}}$ for reducing the Σ_2^0-complete problem $U^\infty \setminus \mathrm{TOTAL}_{\mathbb{N}}$ to $\mathrm{FIN}_{\mathbb{N}}$ which implies the Σ_2^0-completeness of $\mathrm{FIN}_{\mathbb{N}}$. We use that $c_{\mathcal{M}} \in U^\infty \setminus \mathrm{TOTAL}_{\mathbb{N}}$ if and only if there is a j with $\mathcal{M}(\vec{n}^{(j)})\uparrow$ and $\mathcal{M}(\vec{n}^{(i)})\downarrow$ for all $i < j$. Let $\mathcal{M} \in \mathsf{M}_{\mathcal{A}}$ and let $\mathcal{N}_{\mathcal{M}} \in \mathsf{M}_{\mathcal{A}}$ be the machine executing the following program.

1. Input $\vec{x} = (x_1, x_2, \ldots, x_n) \in U^\infty$;
2. $i := \bar{0}$;
3. simulate $\mathcal{M}$ on $\vec{n}^{(i)}$;
4. if $\vec{x} = \vec{n}^{(i)}$ then halt else $\{i := \mathrm{succ}(i)$; goto 3$\}$.

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

Consequently, $c_{\mathcal{N}_\mathcal{M}} \in \mathrm{TOTAL}_{\mathbb{N}}$ if $c_\mathcal{M} \in \mathrm{TOTAL}_{\mathbb{N}}$, and if there is a j with $\mathcal{M}(\vec{n}^{(i)}) \downarrow$ for all $i < j$ and $\mathcal{M}(\vec{n}^{(j)}) \uparrow$, then $\mathcal{N}_\mathcal{M}(\vec{n}^{(i)}) \downarrow$ for all $i < j$ and $\mathcal{N}_\mathcal{M}(\vec{n}^{(k)}) \uparrow$ for all $k \geqslant j$. Thus, $c_\mathcal{M} \in U^\infty \setminus \mathrm{TOTAL}_{\mathbb{N}}$ if and only if $c_{\mathcal{N}_\mathcal{M}} \in \mathrm{FIN}_{\mathbb{N}}$. □

5. The class $\mathcal{A}$-Δ_2^0 and a limit lemma for $\mathcal{A}$

The discussion in this section was inspired by the paper Spreen (1995) where, building on the Limit Lemma, the class Δ_2^0 of the arithmetical hierarchy was characterized by infinitary domains. We want to continue the study of the class $\mathcal{A}$-Δ_2^0 with particular reference to the well-known standard work (Soare 1987).

In order to characterize the class $[\mathcal{A}\text{-}]\Delta_2^0$, we transfer some notions and the Limit Lemma (cf. Soare 1987, p. 57) and describe the characteristic functions χ_P of the problems $P \in \Delta_2^0$ by means of computable total functions. If $\mathcal{A}$ is a structure over the reals including the non-negative integers in $\mathbb{N}$, then the following holds for the usual notion of limit which can be defined by means of the Euclidean metric for any subspace $\mathbb{R}^n$. If $\mathcal{A}$ is a structure for which this metric is not available, then we take the discrete metric and, for any sequence $y_0, y_1, \ldots$ of elements in U, we use the notation $\lim_{s\to\infty} y_s = y$ if $y_s = y$ holds for cofinitely many values of s.

For any sequence of total functions $f_s : U^\infty \to U^\infty$ (with $f_{\bar{s}} = f_s$ for all $s \in \mathbb{N}$) and a function $f : U^\infty \to U^\infty$, we write $\lim_{s\to\infty} f_s = f$ if $f_0, f_1, f_2, \ldots$ converges pointwise to f. Thus, for any sequence of total functions $f_s : U^\infty \to \bar{\mathbb{N}}$ and any total function f with $f = \lim_{s\to\infty} f_s$, we have the following properties.

1. $\lim_{s\to\infty} f_s(\vec{x}) = f(\vec{x})$ holds for any $\vec{x} \in U^\infty$.
2. The image $f(U^\infty)$ of the limit f is a subset of $\bar{\mathbb{N}}$.
3. For all $\vec{x} \in U^\infty$, there is an $s_0 \in \mathbb{N}$ such that $f(\vec{x}) = f_s(\vec{x})$ for all $s \geqslant s_0$.

The properties 2 and 3 hold since the range of all f_s is discrete.

A function $m : U^\infty \to \bar{\mathbb{N}}$ with $f(\vec{x}) = f_s(\vec{x})$ for all $\vec{x} \in U^\infty$ and $s \geqslant m(\vec{x})$ is called a *modulus of convergence for* $(f_s)_{s\in\mathbb{N}}$. The *least modulus of convergence for* $(f_s)_{s\in\mathbb{N}}$ is the function $m(\vec{x}) = \min\{s \in \bar{\mathbb{N}} \mid f(\vec{x}) = f_s(\vec{x})\}$. A sequence of functions $f_s : U^\infty \to U^\infty$ is *computable over* $\mathcal{A}$ if there is a machine $\mathcal{M} \in \mathsf{M}_\mathcal{A}$ that computes the mapping $(s.\vec{x}) \mapsto f_s(\vec{x})$ for all $s \in \bar{\mathbb{N}}$ and $\vec{x} \in U^\infty$. For a computable sequence of functions $f_s : U^\infty \to \{\bar{0}, \bar{1}\}$ and a problem $P \subseteq U^\infty$ with $\lim_{s\to\infty} f_s = \chi_P$, its least modulus m_0 and χ_P are computable by means of the oracle $\{(\vec{x}, m(\vec{x})) \mid \vec{x} \in U^\infty\}$ if m is any modulus of convergence. However, m is usually not computable with χ_P as oracle. Corollary 3.4 in Soare (1987, p. 57) – which implies that a problem P has a recursively enumerable degree iff χ_P is the limit of a computable sequence for which a modulus can be computed by means of P – fails for certain structures $\mathcal{A}$ and thus cannot be used in order to characterize classes as $\mathcal{A}$-Δ_2^0. However, on the one hand, some of the classical results can be used and transferred, but, on the other hand, in proofs we cannot use arguments such as the recursive enumeration of the Halting Problem that are typical of the classical theory.

For any $\mathcal{A}$-machine $\mathcal{M}$, let $S_\mathcal{M}$ denote the sequence $(f_s)_{s\in\mathbb{N}}$ of partial (or total) functions $f_s : U^\infty \to U^\infty$ computed by $\mathcal{M}$ by outputting $f_s(\vec{x})$ on the input $(s.\vec{x})$ for all $s \in \bar{\mathbb{N}}$ and $\vec{x} \in U^\infty$. Moreover, for all $\vec{x} \in U^\infty$, if $f_s(\vec{x}) \downarrow$ (which means that $f_s(\vec{x})$ is defined) for

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

all $s \in \bar{\mathbb{N}}$, then we denote the corresponding sequence $(f_s(\vec{x}))_{s\in\mathbb{N}}$ by $S_{\vec{x},\mathcal{M}}$ and we write $S_{\vec{x},\mathcal{M}} \downarrow$ if $S_{\vec{x},\mathcal{M}}$ converges. A first characterization of the convergence of total functions with the discrete range $\bar{\mathbb{N}}$ is the following.

Proposition 5.1. For

$$\mathrm{KONV}_{\mathbb{N}} = \{(\vec{x}\,.\,c_{\mathcal{M}}) \in U^{\infty} \mid \mathcal{M} \in \mathsf{M}_{\mathcal{A}} \;\&\; (\forall f_s \in S_{\mathcal{M}})(f_s(\vec{x}) \downarrow \;\&\; f_s(\vec{x}) \in \bar{\mathbb{N}}) \;\&\; S_{\vec{x},\mathcal{M}} \downarrow\},$$

we have $\mathrm{KONV}_{\mathbb{N}} \preceq_1 \mathrm{TOTAL}_{\mathbb{N}} \times \mathrm{FIN}_{\mathbb{N}}$.

Proof. Let $\mathcal{M}$ be any $\mathcal{A}$-machine and let $S_{\mathcal{M}} = (f_s)_{s\in\mathbb{N}}$ be the sequence of partial functions $f_s : U^{\infty} \to U^{\infty}$ computed by $\mathcal{M}$ on $\bar{\mathbb{N}} \times U^{\infty}$.

Moreover, let $D_{S_{\mathcal{M}}} = \{\vec{x} \in U^{\infty} \mid f_s(\vec{x}) \in \bar{\mathbb{N}} \text{ for all } s \in \mathbb{N}\}$ and let $\mathcal{L}_{\vec{x},\mathcal{M}}$ be the machine executing the following program.

1. Input $\vec{y} = (y_1, \ldots, y_m) \in U^{\infty}$;
2. simulate $\mathcal{M}$ on $(y_1\,.\,\vec{x})$ (at first without making an output) (if $\mathcal{M}(y_1\,.\,\vec{x}) \downarrow$ then goto 3);
3. check (i.e. semi-decide) whether the computed output value $\vec{z}$ is in $\bar{\mathbb{N}}$ by using the program of $\mathcal{M}_{\bar{\mathbb{N}}}$; if $\vec{z} \in \bar{\mathbb{N}}$ then output $\vec{z}$.

Then, $D_{S_{\mathcal{M}}}$ can be reduced to $\mathrm{TOTAL}_{\mathbb{N}}$ by the mappings $\vec{x} \mapsto (\vec{x}\,.\,c_{\mathcal{M}}) \mapsto c_{\mathcal{L}_{\vec{x},\mathcal{M}}}$.

Now, let $G_{S_{\mathcal{M}}} = \{\vec{x} \in U^{\infty} \mid (\exists m \in \mathbb{N})(\forall s \geqslant m)(f_m(\vec{x}) = f_s(\vec{x}))\}$ and, for any $\vec{x} \in U^{\infty}$, let $\mathcal{N}_{\vec{x},\mathcal{M}}$ be the machine executing the following program where we assume that $\mathrm{succ}(y_1) = y_1$ holds for all $y_1 \notin \bar{\mathbb{N}}$.

1. Input $\vec{y} = (y_1, \ldots, y_m) \in U^{\infty}$;
2. if $m \geqslant 2$ then goto 2;
3. simulate $\mathcal{M}$ on $(y_1\,.\,\vec{x})$ and $(\mathrm{succ}(y_1)\,.\,\vec{x})$ (without making an output);
4. if the two computed output values are not equal, then output any value else goto 4.

$G_{S_{\mathcal{M}}}$ can be reduced to $\mathrm{FIN}_{\mathbb{N}}$ by the mappings $\vec{x} \mapsto (\vec{x}\,.\,c_{\mathcal{M}}) \mapsto c_{\mathcal{N}_{\vec{x},\mathcal{M}}}$.

□

Now, let us transfer the Limit Lemma from the classical theory. To simplify matters, we use a function $u : U^{\infty} \to \bar{\mathbb{N}}$ that has properties similar to the modulus of convergence.

Lemma 5.1 (Limit Lemma for the structure $\mathcal{A}$). Let $f : U^{\infty} \to U^{\infty}$ be a total function. f is computable by a machine in $\mathsf{M}_{\mathcal{A}}(\mathbb{H}_{\mathcal{A}})$ and it holds $f : U^{\infty} \to \bar{\mathbb{N}}$ if and only if there are a computable sequence of total functions $g_s : U^{\infty} \to U^{\infty}$ and a function $u : U^{\infty} \to \bar{\mathbb{N}}$ such that $f = \lim_{s\to\infty} g_s$ and $g_s(\vec{x}) \in \bar{\mathbb{N}}$ for all $\vec{x} \in U^{\infty}$ and all $s \in \bar{\mathbb{N}}$ with $s \geqslant u(\vec{x})$.

Proof.

a. Let $\mathcal{M} \in \mathsf{M}_{\mathcal{A}}(\mathbb{H}_{\mathcal{A}})$ compute $f : U^{\infty} \to \bar{\mathbb{N}}$.
The following machine $\mathcal{N}_g \in \mathsf{M}_{\mathcal{A}}$ simulates $\mathcal{M}$ with an accuracy depending on an inputted integer s. Step-by-step each instruction of $\mathcal{M}$ will be simulated or modified and then executed. If s is great enough, then, starting with the input $(s\,.\,\vec{x}) \in U^{\infty}$ the machine $\mathcal{N}_g \in \mathsf{M}_{\mathcal{A}}$ exactly simulates the computation of $\mathcal{M}$ on input $\vec{x} \in U^{\infty}$.
   1. Input $(s\,.\,\vec{x}) \in U^{\infty}$;
   2. $j := \bar{0}$;

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

3. if $s \neq j$, then $\{j := \mathrm{succ}(j)$; goto 3$\}$;
4. simulate and evaluate the instructions of $\mathcal{M}$ as follows:
   - simulate all instructions of $\mathcal{M}$ on $\vec{x}$ which do not require the evaluation of queries;
   - for all oracle instructions of the form (1) of $\mathcal{M}$, check whether the oracle query has the form $(\vec{y}\,.\,c_{\mathcal{L}}) \in \mathbb{H}_{\mathcal{A}}?$ and in the positive case, execute the modified instruction

$$\text{if } \mathcal{L}(\vec{y})\downarrow^s \text{ then goto } l_1 \text{ else goto } l_2;$$

5. if the output of $\mathcal{M}$ is not reached within the simulation of the first s steps of $\mathcal{M}$, then output 0.

Let g be the function computed by $\mathcal{N}_g$ and $g_s(\vec{x}) = g(s\,.\,\vec{x})$ for all $\vec{x} \in U^\infty$ and all $s \in \bar{\mathbb{N}}$.

In order to show that g possesses the required features, let us once more consider the maximal sequence of all oracle queries $(\vec{y}^{(1)}\,.\,c_{\mathcal{L}_1}) \in \mathbb{H}_{\mathcal{A}}?, \ldots, (\vec{y}^{(k)}\,.\,c_{\mathcal{L}_k}) \in \mathbb{H}_{\mathcal{A}}?$ executed during the computation of $\mathcal{M}$ on input $\vec{x}$ in this order before the machine halts. Then, there is again a finite sequence $\beta_1, \beta_2, \ldots, \beta_k \in \mathbb{N}$ with $\beta_i = t > 0$ iff $\mathcal{L}_i(\vec{y}^{(i)})\uparrow^{t-1}$ and $\mathcal{L}_i(\vec{y}^{(i)})\downarrow^t$ and with $\beta_i = 0$ iff $\mathcal{L}_i(\vec{y}^{(i)})\uparrow$.

Analogously, for any $(s\,.\,\vec{x})$ with $s \in \bar{\mathbb{N}}$, there is a finite sequence $\beta_1^{(s)}, \ldots, \beta_{m_s}^{(s)} \leqslant s$ of numbers of steps representing the answers of the modified oracle queries $\mathcal{L}_{s,i}(\vec{y}^{(s,i)})\downarrow^s?$ during the computation of $\mathcal{N}_g$ on input $(s\,.\,\vec{x})$ in this order before the machine halts (i.e. $\mathcal{L}_{s,i}(\vec{y}^{(s,i)})\uparrow^{\beta_i^{(s)}-1}$ and $\mathcal{L}_{s,i}(\vec{y}^{(s,i)})\downarrow^{\beta_i^{(s)}}$ iff $\beta_i^{(s)} \neq 0$). By the following arguments. there is a sequence $0 \leqslant s_1 \leqslant s_2 \leqslant \cdots \leqslant s_{k_0}$ $(k_0 \leqslant k)$ such that

$$(i)\quad (\beta_1, \beta_2, \ldots, \beta_{j_i}) = (\beta_1^{(s_i)}, \beta_2^{(s_i)}, \ldots, \beta_{j_i}^{(s_i)}),\ \beta_{j_i+1} \neq \beta_{j_i+1}^{(s_i)} \text{ for } j_i < m_{s_i}, k_0,$$

$$(k)\quad (\beta_1, \beta_2, \ldots, \beta_k) = (\beta_1^{(s)}, \beta_2^{(s)}, \ldots, \beta_k^{(s)}) \qquad \text{for } k = m_s,\ s \geqslant s_{k_0}.$$

Let $i < k$ and let us assume that (i) holds. Then, $(\vec{y}^{(s_i,j)}\,.\,c_{\mathcal{L}_{s_i,j}}) = (\vec{y}^{(j)}\,.\,c_{\mathcal{L}_j})$ for every $j \leqslant j_i{+}1$. Moreover, the statement, that the values in the sequence $\beta_{j_i+1}^{(s_i)}, \beta_{j_i+1}^{(s_i+1)}, \beta_{j_i+1}^{(s_i+2)}, \ldots$ are not all equal, can only mean that the first members $\beta_{j_i+1}^{(s_i)}, \beta_{j_i+1}^{(s_i+1)}, \ldots, \beta_{j_i+1}^{(s_i+r)}$ are 0, and, for some r, $\beta_{j_i+1}^{(s_i+r+1)} = \beta_{j_i+1}$ will be a non-zero value. If $\beta_{j_i+1} > s_i$, then let $s_{i+1} = \beta_{j_i+1} = \beta_{j_i+1}^{(s_i+r+1)}$. If $\beta_{j_i+1} = 0$, then we can have only $\beta_{j_i+1}^{(s_i+1)} = \beta_{j_i+1}^{(s_i+2)} = \cdots = 0$. This case and the case $\beta_{j_i+1} \leqslant s_i$ are not possible by definition of (i). If $s \geqslant s_{k_0}$, then, on $(s\,.\,\vec{x})$, $\mathcal{N}_g$ simulates s steps of the computation of $\mathcal{M}$ on $\vec{x}$ exactly. Since $\mathcal{M}$ halts on $\vec{x}$, there is an $s_{\vec{x}}$ such that $\mathcal{N}_g$ halts on $(s_{\vec{x}}\,.\,\vec{x})$ and outputs the same value as $\mathcal{M}$ for $\vec{x}$.

b. Now, let $g : U^\infty \to U^\infty$ be a computable function with $g(s\,.\,\vec{x}) = g_s(\vec{x})$ for all $\vec{x} \in U^\infty$ and all $s \in \bar{\mathbb{N}}$, and let $g(s\,.\,\vec{x}) \in \bar{\mathbb{N}}$ for all $\vec{x} \in U^\infty$ and all $s \in \bar{\mathbb{N}}$ with $s \geqslant u(\vec{x})$. Then, let $\mathcal{M} \in \mathsf{M}_{\mathcal{A}}(\mathbb{H}_{\mathcal{A}})$ execute the following program.

1. Input $\vec{x} \in U^\infty$;
2. $s := \bar{0}$;

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

3. ask the oracle $\mathbb{H}_{\mathcal{A}}$ whether the following machine $\mathcal{L}$ halts on $(s.\vec{x})$;
   $\mathcal{L}$:
   1′. input $(s.\vec{x})$;
   2′. compute $g(s.\vec{x})$;
   3′. $k := \mathrm{succ}(s)$;
   4′. compute $g(k.\vec{x})$;
   5′. if $g(s.\vec{x}) = g(k.\vec{x})$ then $\{k := \mathrm{succ}(k)$; goto 4′$\}$ else halt;
   if $\mathcal{L}(s.\vec{x})\downarrow$ then $\{s := \mathrm{succ}(s)$; goto 3$\}$ else output $g(s.\vec{x})$.

If $f(\vec{x}) = \lim_{s\to\infty} g(s.\vec{x})$ and $g(s.\vec{x}) \in \bar{\mathbb{N}}$ for all $\vec{x} \in U^\infty$ and all $s \in \bar{\mathbb{N}}$ with $s \geqslant u(\vec{x})$, then $\mathcal{M}$ computes f. □

Since the latter proposition holds also for total functions $f : U^\infty \to \{\bar{0}, \bar{1}\}$, we get the following.

Theorem 5.1 (A characterization of $\mathcal{A}$-Δ_2^0). Let $P \subseteq U^\infty$ be any problem. Then, P is in $\mathcal{A}$-Δ_2^0 if and only if there are a computable sequence of total functions $g_s : U^\infty \to U^\infty$ and a function $u : U^\infty \to \bar{\mathbb{N}}$ such that $\chi_P = \lim_{s\to\infty} g_s$ and $g_s(\vec{x}) \in \bar{\mathbb{N}}$ for all $\vec{x} \in U^\infty$ and all $s \in \bar{\mathbb{N}}$ with $s \geqslant u(\vec{x})$.

6. Real BSS RAM's and algebraic problems

In this section, we want to study some special features of BSS RAM's over the reals. On the one hand, real RAM's are a very important model of computation for describing the complexity of algorithms in theoretical terms. In such models, any real number is viewed as an object that can be stored in one register and, depending on the problem, suitable operations such as the addition, the multiplication, and the k-roots are chosen to be the primitive operations that are available at unit cost. On the other hand, their features, capabilities and limitations have not yet been investigated comprehensively and, therefore, we want to continue our investigation of additive real BSS RAM's by considering fundamental properties of this model which were first studied in papers such as the following. The uniform BSS model of computation was introduced in Blum *et al.* (1989), additive real RAM's that are able to execute additions and subtractions were introduced by Dobkin and Lipton (1978), and the additive and linear BSS machines were considered, for instance, in Koiran (1994), Cucker and Koiran (1995), Gaßner (1997) and Meer and Ziegler (2008).

For structures $\mathcal{A}$ given by $\mathbb{R}^= =_{\mathrm{df}} (\mathbb{R}; \mathbb{R}; \cdot, +, -; =)$, $\mathbb{R} =_{\mathrm{df}} (\mathbb{R}; \mathbb{R}; \cdot, +, -; \geqslant)$, $\mathbb{R}^=_{\mathrm{add}} =_{\mathrm{df}} (\mathbb{R}; \mathbb{R}; +, -; =)$ and $\mathbb{R}_{\mathrm{add}} =_{\mathrm{df}} (\mathbb{R}; \mathbb{R}; +, -; \geqslant)$, we consider the classes $\mathsf{M}_{\mathcal{A}} = \mathsf{M}^{[=]}_{[\mathrm{add}]}$ of machines over $\mathbb{R}^{[=]}_{[\mathrm{add}]}$ and the halting problems

$$\mathbb{H}^{[=]}_{[\mathrm{add}]} =_{\mathrm{df}} \{(\vec{x}.c_{\mathcal{M}}) \mid \vec{x} \in \mathbb{R}^\infty \ \&\ \mathcal{M} \in \mathsf{M}^{[=]}_{[\mathrm{add}]} \ \&\ \mathcal{M}(\vec{x})\downarrow\}.$$

The following algebraic problems $\mathbb{P}_{\mathrm{sqrt}}, \mathbb{A}_{\mathrm{alg}} \subseteq \mathbb{R}$ are examples that show that problems can be in classes of different levels if we allow different sets of basic operations and relations. Let

$$\begin{aligned} \mathbb{P}_{\mathrm{sqrt}} &= \{\sqrt{p} \mid p \in \mathbb{P}\}, \\ \mathbb{A}_{\mathrm{alg}} &= \{x \in \mathbb{R} \mid (\exists p \in \mathbb{Z}[x])(p(x) = 0 \ \&\ p \not\equiv \mathbf{0})\}, \end{aligned}$$

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

where $\mathbb{P} = \{p \mid p \text{ is a prime number}\}$ and, for $p \in \mathbb{Z}[x]$, $p \not\equiv \mathbf{0}$ means that we have a non-zero polynomial p with $p(x) = z_n x^n + \cdots + z_1 x^1 + z_0$ for some $n \geqslant 0$ and $z_0, z_1, \ldots, z_n \in \mathbb{Z}$ with $z_n \neq 0$.

It is known that both problems are neither contained in $\mathbb{R}^{=}_{\text{add}}$-$\Delta^0_1$ nor in $\mathbb{R}^{=}$-Δ^0_1, since all problems $P \subseteq \mathbb{R}$ decidable over $\mathbb{R}^{=}$ are finite or cofinite which follows from the analysis of computation paths in analogy to Gaßner (2013, p. 5).

For the set $\mathbb{P}_{\text{sqrt}}$ of square roots of the prime numbers, we have

- $\mathbb{P}_{\text{sqrt}} \in \mathbb{R}$-$\Delta^0_1$ since it is possible to check $x^2 = p$ for all prime numbers $p \leqslant x^2$,
- $\mathbb{P}_{\text{sqrt}} \in \mathbb{R}^{=}$-$\Sigma^0_1$ since $x \in \mathbb{P}_{\text{sqrt}} \iff (\exists n \in \mathbb{N})(n \in \mathbb{P} \;\&\; x^2 = n)$,
- $\mathbb{P}_{\text{sqrt}} \in \mathbb{R}_{\text{add}}$-$\Pi^0_1$ since it is decidable over $\mathbb{R}_{\text{add}}$ whether two real numbers p_1, p_2 are two successive prime numbers (i.e. whether $(p_1, p_2) \in R \setminus \{(1,2)\}$ for the relation R defined below holds). Both relations $x < q$ and $x < q^2$ for any $x \in \mathbb{R}$ and any number q in the set $\mathbb{Q}$ of rational numbers as well as the relation

$$R =_{\text{df}} \{(p_1, p_2) \in \mathbb{R}^2 \mid$$
$$p_1 < p_2 \;\&\; p_1, p_2 \in \mathbb{P} \cup \{1\} \;\&\; (\forall r \in \mathbb{P})(p_1 \leqslant r \leqslant p_2 \Rightarrow r = p_1 \vee r = p_2)\}$$

  are decidable over $\mathbb{R}_{\text{add}}$ (since the decidability of $\mathbb{P}$ over $\mathbb{R}_{\text{add}}$ follows from the decidability of $\mathbb{N}$ over $\mathbb{R}_{\text{add}}$). Consequently, the set $\mathbb{R} \setminus \mathbb{P}_{\text{sqrt}}$ is semi-decidable over $\mathbb{R}_{\text{add}}$ since all its elements range between the roots of two successive prime numbers. More precisely, we have

$$x \notin \mathbb{P}_{\text{sqrt}}$$
$$\iff (x \leqslant 1 \vee (\exists p_1, p_2 \in \mathbb{N})(\exists q, r \in \mathbb{Q})(R(p_1, p_2) \;\&\; p_1 < q^2 \;\&\; q \leqslant x \leqslant r \;\&\; r^2 < p_2))$$

  and, consequently, by De Morgan's laws we get

$$x \in \mathbb{P}_{\text{sqrt}}$$
$$\iff (x > 1 \;\&\; (\forall p_1, p_2 \in \mathbb{N})(\forall q, r \in \mathbb{Q})(\neg R(p_1, p_2) \vee p_1 \geqslant q^2 \vee \neg(q \leqslant x \leqslant r) \vee r^2 \geqslant p_2)).$$

Later, we will see that $\mathbb{P}_{\text{sqrt}} \notin \mathbb{R}_{\text{add}}$-$\Delta^0_1$ holds.

For the set $\mathbb{A}_{\text{alg}}$ of all real algebraic numbers which is undecidable over the ordered ring of real numbers (as investigations in Meer and Ziegler 2008 and in Calvert *et al.* 2011 confirm) we have the following relationships.

- $\mathbb{A}_{\text{alg}} \in \mathbb{R}^{=}$-$\Sigma^0_1$ follows directly from the definition of $\mathbb{A}_{\text{alg}}$ since the set $\mathbb{Z}[x]$ of all polynomials with coefficients in $\mathbb{Z}$ is effectively enumerable over $\mathbb{R}^{=}$.
- $\mathbb{A}_{\text{alg}} \in \mathbb{R}$-$\Sigma^0_1$ follows from $\mathbb{A}_{\text{alg}} \in \mathbb{R}^{=}$-$\Sigma^0_1$.
- $\mathbb{A}_{\text{alg}} \in \mathbb{R}_{\text{add}}$-$\Sigma^0_3$ holds since we can apply Bolzano's theorem (the intermediate value theorem) to polynomial functions since these functions are continuous. Accordingly, a characterization of the zeros of a polynomial which is not the zero polynomial results from the fact that any zero is in some interval the only zero and that it satisfies one of the following three cases $k = 1, 2, 3$. In the first case $k = 1$, the graph of the polynomial crosses the x-axis at the zero. In both other cases, it only touches the x-axis at the zero. We have

$$x \in \mathbb{A}_{\text{alg}} \iff (\exists p \in \mathbb{Z}[y])(\exists a, b \in \mathbb{Q})(\exists k \in \mathbb{N})(\forall m \in \mathbb{N}^+)(\forall q_1, q_2 \in \mathbb{Q})(\exists r_1, r_2 \in \mathbb{Q})$$

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

$$
\begin{aligned}
&(a < q_1 < x < q_2 < b\\
&\Rightarrow ((k = 1 \ \&\ p(q_1) > 0 \ \&\ p(q_2) < 0)\\
&\vee (k = 1 \ \&\ p(q_1) < 0 \ \&\ p(q_2) > 0)\\
&\vee (k = 2 \ \&\ q_1 < r_1 < x < r_2 < q_2 \ \&\ p(q_1), p(q_2) < 0 \ \&\ p(r_1), p(r_2) > -\frac{1}{m})\\
&\vee (k = 3 \ \&\ q_1 < r_1 < x < r_2 < q_2 \ \&\ p(q_1), p(q_2) > 0 \ \&\ p(r_1), p(r_2) < \frac{1}{m}))).
\end{aligned}
$$

Note, that the following description, derived directly from the definition of continuity, implies the weaker relationship $\mathbb{A}_{\mathrm{alg}} \in \mathbb{IR}_{\mathrm{add}}\text{-}\Sigma_4^0$. For any real algebraic number x, there is some polynomial $p \in \mathbb{Z}[x]$ such that, for all positive fractions $\frac{1}{k}$, there is a neighbourhood $U =]x - \frac{1}{l}, x + \frac{1}{l}[$ of x such that $|p(\frac{n}{m})| < \frac{1}{k}$ holds for all rational numbers $\frac{n}{m} \in U$. Consequently, we have $\mathbb{A}_{\mathrm{alg}} \in \mathbb{IR}_{\mathrm{add}}\text{-}\Sigma_4^0$ since

$$
\begin{aligned}
&x \in \mathbb{A}_{\mathrm{alg}} \Longleftrightarrow (\exists s \in \mathbb{N})(\exists z_0, \ldots, z_s \in \mathbb{Z})(\forall k \in \mathbb{N}^+)(\exists l \in \mathbb{N}^+)(\forall n \in \mathbb{Z})(\forall m \in \mathbb{N}^+)\\
&((z_0, \ldots, z_s) \neq (0, \ldots, 0) \ \&\ (x - \frac{1}{l} < \frac{n}{m} < x + \frac{1}{l} \Rightarrow |z_s(\frac{n}{m})^s + \cdots + z_0(\frac{n}{m})^0| < \frac{1}{k})).
\end{aligned}
$$

On the other hand, we can show that $\mathbb{A}_{\mathrm{alg}} \notin \bigcup_{n \geqslant 1} \mathbb{IR}^{=}_{\mathrm{add}}\text{-}\Sigma_n^0$ and that there is no oracle $\mathcal{O} \in \bigcup_{n \geqslant 1} \mathbb{IR}^{=}_{\mathrm{add}}\text{-}\Sigma_n^0$ such that there is a machine in $\mathsf{M}_{\mathrm{add}}(\mathcal{O})$ semi-deciding the real algebraic numbers.

The following Query Lemma gives a characterization of the queries of the machines in $\mathsf{M}_{\mathrm{add}}(\mathbb{IH}_{\mathrm{add}})$ on inputs in $\mathbb{IR}$. It is a generalization of Lemma 4.6 in Gaßner (2013) that was given for characterizing the queries of the machines in $\mathsf{M}_{\mathbb{IR}^1_{\mathrm{add}}}(\mathbb{IH}^{1,=}_{\mathrm{add}})$ over $\mathbb{IR}^1_{\mathrm{add}} =_{\mathrm{df}} (\mathbb{IR}; 0, 1; +, -; \geqslant)$ with respect to the halting problem $\mathbb{IH}^{1,=}_{\mathrm{add}}$ for the machines in $\mathsf{M}^{1,=}_{\mathrm{add}}$ over $\mathbb{IR}^{1,=}_{\mathrm{add}} =_{\mathrm{df}} (\mathbb{IR}; 0, 1; +, -; =)$. We will use that, for structures such as $\mathbb{IR}_{\mathrm{add}}$ and $\mathbb{IR}^1_{\mathrm{add}}$ with or without arbitrary real constants, the corresponding halting problems $\mathbb{IH}_{\mathrm{add}}$ and $\mathbb{IH}^1_{\mathrm{add}}$ represent the same Turing degree with respect to the here considered BSS RAM's over real numbers.

Proposition 6.1. $\mathbb{IH}^1_{\mathrm{add}}$ is one-one-reducible to $\mathbb{IH}_{\mathrm{add}}$ and $\mathbb{IH}_{\mathrm{add}}$ is one-one-reducible to $\mathbb{IH}^1_{\mathrm{add}}$ and the reduction functions are computable by machines over $\mathbb{IR}^{1,=}_{\mathrm{add}}$.

For any machine in $\mathcal{M} \in \mathsf{M}_{\mathrm{add}}(\mathbb{IH}_{\mathrm{add}})$ with the machine constants $r_1, \ldots, r_i$ let $\vec{r} = \vec{r}_{\mathcal{M}} = r_1, \ldots, r_i$ and let $L_{\mathcal{M}}$ and $R_{\mathcal{M}}$ be the effectively enumerable sets of all linear combinations of $r_1, \ldots, r_i$ with scalars in $\mathbb{Z}$ and $\mathbb{Q}$, respectively, i. e. let

$$
\begin{aligned}
L_{\mathcal{M}} = L_{\vec{r}} &= \{m_1 r_1 + \cdots + m_i r_i \mid m_1, \ldots, m_i \in \mathbb{Z}\},\\
R_{\mathcal{M}} = R_{\vec{r}} &= \{q_1 r_1 + \cdots + q_i r_i \mid q_1, \ldots, q_i \in \mathbb{Q}\}.
\end{aligned}
$$

Every computation path B traversed by $\mathcal{M} \in \mathsf{M}_{\mathrm{add}}(\mathbb{IH}_{\mathrm{add}})$ can be uniquely described by a sequence $(s_1, s_2, \ldots, s_l)$ of labels of instructions executed by $\mathcal{M}$ in this order. Consequently, for all inputs $x \in \mathbb{IR}$, for which $\mathcal{M}$ traverses B, at any time the content of any register of $\mathcal{M}$ can be described by a term $kX + l$ for some $k \in \mathbb{Z}$ and $l \in L_{\mathcal{M}}$ that are independent of the input values and that result only from the application of the computation instructions on the variable X for inputs x of length 1. Moreover, for these inputs, all index registers I_j get the value 1 at the beginning of the computation of $\mathcal{M}$ and, during the computation

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

along the path B, all considered index registers obtain integers that are also independent of these inputs of length 1. That means that, for the fixed path B and inputs $x \in \mathbb{R}$, any oracle query results from the general form

$$(k_1 X + l_1, \ldots, k_m X + l_m) \in \mathbb{H}^1_{\text{add}}? \tag{5}$$

where m is fixed and all k_i are in $\mathbb{Z}$ and all l_i are in $L_{\mathcal{M}}$ $(i = 1, \ldots, m)$ and these numbers result only from the computation on the variable X that stands for input values $x \in \mathbb{R}$. Accordingly, after replacing X in the formula (5) by an input value x the oracle query has the form

$$((k_1 x + l_1, \ldots, k_j x + l_j) . \underbrace{(k_{j+1} x + l_{j+1}, \ldots, k_{j+s} x + l_{j+s})}_{=\vec{y}}) \in \mathbb{H}^1_{\text{add}}? \tag{6}$$

and if, for any $j < m$, $\vec{y}$ is not the code of some $\mathcal{N} \in \mathsf{M}^1_{\text{add}}$, then the query is always answered in the negative. In the other case, we have the following.

Lemma 6.1 (Query Lemma). Let $\mathcal{M}$ be a machine in $\mathsf{M}_{\text{add}}(\mathbb{H}^1_{\text{add}})$. If, for some input $x_0 \in \mathbb{R} \setminus R_{\mathcal{M}}$, an oracle query of the form (6) agrees with

$$((k_1 x_0 + l_1, \ldots, k_j x_0 + l_j) . c_{\mathcal{N}}) \in \mathbb{H}^1_{\text{add}}? \tag{7}$$

for some $j \in \mathbb{Z}$ and a machine $\mathcal{N} \in \mathsf{M}^1_{\text{add}}$, then this oracle query has the same form

$$((k_1 x + l_1, \ldots, k_j x + l_j) . c_{\mathcal{N}}) \in \mathbb{H}^1_{\text{add}}? \tag{8}$$

(with the same $j, k_i \in \mathbb{Z}, l_i \in L_{\mathcal{M}}$) for all inputs $x \in \mathbb{R}$ for which $\mathcal{M}$ goes through the same computation path as for x_0 until this query is executed.

Proof. Let B be the computation path traversed by $\mathcal{M}$ on some $x \in \mathbb{R}$. Then, for all inputs $x \in \mathbb{R}$, any query along the path B has the form (6) in accordance with the general form (5). Let us assume that for such an $x_0 \in \mathbb{R} \setminus R_{\mathcal{M}}$ there is a machine $\mathcal{N} \in \mathsf{M}^1_{\text{add}}$ such that $\vec{y}$ is its code $c_{\mathcal{N}} \in \{0,1\}^\infty$. Since $x_0 \in \mathbb{R} \setminus R_{\mathcal{M}}$ satisfies equations of the form $g_\lambda(X) =_{\text{df}} k_\lambda X + l_\lambda = 0$ and $h_\mu(X) =_{\text{df}} k_\mu X + l_\mu = 1$ only if $k_\lambda = k_\mu = 0$ holds, $c_{\mathcal{N}}$ is determined by constant functions $g_\lambda(X) = 0$ and $h_\mu(X) = 1$ and thus, for all $x \in \mathbb{R}$ for which $\mathcal{M}$ goes through the computation path B, the query Q has the same form (8). □

We want to emphasize the following point.

Lemma 6.2 (Computation paths and intervals). If $\mathcal{N}(k_1 x_0 + l_1, \ldots, k_j x_0 + l_j) \downarrow$ for some machine $\mathcal{N} \in \mathsf{M}^1_{\text{add}}$, some $j, k_i \in \mathbb{Z}$, $l_i \in L_{\vec{r}}$, and $x_0 \notin R_{\vec{r}}$ $(\vec{r} \in \mathbb{R}^\infty)$, then there are $a, b \in R_{\vec{r}}$ with $a < x_0 < b$ such that $\mathcal{N}$ traverses the same computation path for all inputs $(k_1 x + l_1, \ldots, k_j x + l_j)$ with $x \in]a, b[$.

Proof. Let B be a finite computation path traversed by $\mathcal{N}$. Then, by analogy with the previous lemma, for inputs $(k_1 x + l_1, \ldots, k_j x + l_j)$ with $x \in \mathbb{R}$, any order test has the form $kx + l \geqslant 0?$ for some $k \in \mathbb{Z}$ and $l \in L_{\vec{r}}$. Thus, $\mathcal{N}$ goes through B on $(k_1 x_0 + l_1, \ldots, k_j x_0 + l_j)$ if and only if x_0 satisfies the corresponding finite system S_B of inequalities of the form $k_\nu X + l_\nu \geqslant 0$ and $k_\mu X + l_\mu > 0$. Since $x_0 \notin R_{\vec{r}}$, we have $k_\nu x_0 + l_\nu \neq 0$ for any possible ν if $k_\nu \neq 0$. Therefore, there is an open neighbourhood of x_0 such that all elements of this neighbourhood satisfy the system S_B. □

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

In the same way it is possible to show the following proposition that describes special properties on which one of the most used techniques of the BSS theory is based.

Proposition 6.2 (Semi-decidability and intervals). If $\mathcal{M}(x_0)\downarrow$ for some machine $\mathcal{M} \in \mathsf{M}_{\mathrm{add}}$ and $x_0 \in \mathbb{R} \setminus R_{\mathcal{M}}$, then there are $a, b \in R_{\mathcal{M}}$ with $a < x_0 < b$ such that $\mathcal{M}$ traverses the same computation path for all inputs $x \in]a, b[$.

Furthermore, the preceding considerations (in the proof of Lemma 6.1) mean that all inputs in $x \in \mathbb{R} \setminus R_{\mathcal{M}}$ for which a machine $\mathcal{M} \in \mathsf{M}_{\mathrm{add}}$ goes through the same path are accepted or all these inputs are rejected. Consequently, we have the following since the set $\mathbb{A}_{\mathrm{alg}}$ is dense.

Corollary 6.1. $\mathbb{A}_{\mathrm{alg}} \notin \mathbb{R}_{\mathrm{add}}\text{-}\Sigma_1^0$.

The following proposition requires a more detailed analysis.

Lemma 6.3. For any $\vec{r} \in \mathbb{R}^\infty$ and any infinite set $P \subseteq \mathbb{P}_{\mathrm{sqrt}}$, there is a prime number p with $\sqrt{p} \in P \setminus R_{\vec{r}}$.

Proof. Whereas the dimension of the vector space $R_{\vec{r}}$ over $\mathbb{Q}$ is finite, the set of the square roots of the prime numbers generates – as known – an infinite dimensional vector space over $\mathbb{Q}$. The latter holds since, for any enumeration of prime numbers $p_1, p_2, \ldots$ and the fields $\mathbb{K}_0 =_{\mathrm{df}} (\mathbb{Q}; 0, 1; \cdot, +, -; =)$ and $\mathbb{K}_{n+1} =_{\mathrm{df}} \mathbb{K}_n[\sqrt{p_{n+1}}]$ $(n \geqslant 0)$, we have the dimension $[\mathbb{K}_{n+1} : \mathbb{K}_n] = 2$ for any $n \geqslant 0$ which can be shown as follows. Let us assume that $\sqrt{p_{n+1}} \in \mathbb{K}_n$. Then, there would be $a, b \in \mathbb{K}_{n-1}$ such that $a + b\sqrt{p_n} = \sqrt{p_{n+1}}$ and thus $(a + b\sqrt{p_n})^2 = p_{n+1}$. However, by induction on n, we can show that there are no different prime numbers $p', p_1'', \ldots, p_r'' \in \{p_{n+1}, p_{n+2}, \ldots\}$ such that $x^2 = \frac{p'}{p_1'' \cdots p_r''}$ for any $n \in \mathbb{N}$, $x \in \mathbb{K}_n$ and $r \geqslant 0$ (where $p_1'' \cdots p_0'' = 1$ in case that $r = 0$). The reason is that the latter equation for $x = a + b\sqrt{p_n} \in \mathbb{K}_n$ with $n \geqslant 1$, $a, b \in \mathbb{K}_{n-1}$ together with $(a + b\sqrt{p_n})^2 = a_1 + b_1\sqrt{p_n} \in K_n$ for some $a_1, b_1 \in \mathbb{K}_{n-1}$ means that $a_1 = a^2 + b^2 p_n = \frac{p'}{p_1'' \cdots p_r''} \in \mathbb{Q}$ and $b_1 = 2ab = 0$ and, consequently, $a = 0$ and $b^2 = \frac{p'}{p_1'' \cdots p_r'' \cdot p_n}$ or $b = 0$ and $a^2 = \frac{p'}{p_1'' \cdots p_r''}$ hold.

Obviously, from $[\mathbb{K}_{n+1} : \mathbb{K}_n] = 2$ it follows that $\sqrt{p_{n+1}}$ must be linearly independent of $\sqrt{p_1}, \ldots, \sqrt{p_n}$ over $\mathbb{Q}$. □

Proposition 6.3. For all infinite sets P of square roots of prime numbers, $P \notin \mathbb{R}_{\mathrm{add}}\text{-}\Sigma_1^0$ holds.

Proof. Let us assume that $\mathcal{M} \in \mathsf{M}_{\mathrm{add}}$ is a machine semi-deciding P by means of constants in $\vec{r}$. By Lemma 6.3, there are a prime number p with $\sqrt{p} \in P \setminus R_{\vec{r}}$ and, by Proposition 6.2, an open interval I including $\sqrt{p}$ such that all elements of I are accepted by $\mathcal{M}$. This is a contradiction to the assumption that $\mathcal{M}$ semi-decides P. □

Corollary 6.2. $\mathbb{P}_{\mathrm{sqrt}} \notin \mathbb{R}_{\mathrm{add}}\text{-}\Sigma_1^0$.

A further characterization of the additive oracle machines is the following where $\mathbb{R}_{\mathrm{add}}^{\vec{r}} =_{\mathrm{df}} (\mathbb{R}; r_1, \ldots, r_i; +, -; \geqslant)$.

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

Proposition 6.4. Let $\mathcal{M} \in \mathsf{M}_{\mathrm{add}}(\mathbb{IH}^1_{\mathrm{add}})$ be an oracle machine over $\mathbb{IR}^{\vec{r}}_{\mathrm{add}}$ deciding a problem $\subseteq \mathbb{IR}$ and let $[a,b] \subseteq \mathbb{IR}$ be an interval with $-\infty \leqslant a < b \leqslant \infty$. Then, there is an interval $I \subseteq [a,b]$ and a set P semi-decidable over $\mathbb{IR}^{\vec{r}}_{\mathrm{add}}$ such that the set $I \setminus (R_{\vec{r}} \cup P)$ is uncountable and $\mathcal{M}$ either rejects all inputs in $I \setminus (R_{\vec{r}} \cup P)$ or accepts all these inputs.

Proof. Let $\mathrm{instr}_1, \mathrm{instr}_2, \ldots, \mathrm{instr}_N$ be instructions over $\mathbb{IR}^{\vec{r}}_{\mathrm{add}}$ with $\vec{r} = (r_1, \ldots, r_i)$ – but, to simplify matters, without branching instructions with conditions of the form $Z_i = Z_j$ – such that

$$1 : \mathrm{instr}_1; \quad 2 : \mathrm{instr}_2; \quad \ldots; \quad N : \mathrm{instr}_N$$

is the program of $\mathcal{M} \in \mathsf{M}_{\mathrm{add}}(\mathbb{IH}^1_{\mathrm{add}})$ that decides a problem $\subseteq \mathbb{IR}$. Since $\mathcal{M}$ decides a problem, every computation path is finite and therefore, by Lemma 6.1, for any computation path B of $\mathcal{M}$, there is a finite system S_B of conditions that is satisfied by an input $x \in \mathbb{IR} \setminus R_{\vec{r}}$ if and only if this path B is traversed by $\mathcal{M}$ on x. The conditions of this system can take any of the following forms:

$$\begin{aligned} k_\nu X + l_\nu \ &\geqslant\ 0, \qquad k_\mu X + l_\mu \ >\ 0, \\ ((k_1 X + l_1, \ldots, k_j X + l_j) . c_{\mathcal{N}}) \ &\in\ \mathbb{IH}^1_{\mathrm{add}}, \\ ((k_1 X + l_1, \ldots, k_j X + l_j) . c_{\mathcal{N}}) \ &\notin\ \mathbb{IH}^1_{\mathrm{add}} \end{aligned} \tag{9}$$

($k_i \in \mathbb{Z}, l_i \in L_{\vec{r}}, \mathcal{N} \in \mathsf{M}^1_{\mathrm{add}}$).

Since the set of all finite computation paths is countable, there is a path B_0 through which $\mathcal{M}$ goes for an uncountable set $M \subseteq [a,b]$ of inputs. For this B_0, let $(s_1, s_2, \ldots, s_l)$ be the sequence of the labels of all instructions executed by $\mathcal{M}$ along B_0 in this order. Moreover, we only consider inputs in $\mathbb{IR} \setminus R_{\vec{r}}$. That means that all index registers I_j get the value 1 at the beginning of the computation of $\mathcal{M}$ and any considered branching condition that is of the form $I_{j_1} = I_{j_2}$ is true for all inputs in $\mathbb{IR}$ for which $\mathcal{M}$ goes through the same path or it is false for all such inputs.

In the following, we define a sequence of sets U_k and semi-decidable P_k recursively such that for the initial part B_k of the path B_0 defined by the labels $s_1, s_2, \ldots, s_k$, the machine traverses this part B_k for an input $x \in [a,b] \setminus R_{\vec{r}}$ if and only if $x \in U_k \setminus (R_{\vec{r}} \cup P_k)$. We will get $U_1 \supseteq U_2 \supseteq \cdots \supseteq U_l$ and $P_1 \subseteq P_2 \subseteq \cdots \subseteq P_l$ such that $\mathcal{M}$ goes through $B_0 = B_l$ for an input $x \in [a,b] \setminus R_{\vec{r}}$ if and only if $x \in U_l \setminus (R_{\vec{r}} \cup P_l)$. This means that all these sets $U_k \setminus (R_{\vec{r}} \cup P_k)$ and, in particular, the set $U_l \setminus (R_{\vec{r}} \cup P_l)$ are uncountable. Every U_k will be a countable union of intervals. Thus, the set U_l includes an open interval (with an uncountable number of elements). P_l will be a semi-decidable set.

Let $U_1 = [a,b]$ and $P_1 = \varnothing$ and, for $k = 2, \ldots, l$, let us assume that U_{k-1} and P_{k-1} are already defined and U_{k-1} is a countable union of intervals and P_{k-1} is semi-decidable over $\mathbb{IR}^{\vec{r}}_{\mathrm{add}}$. The possible types of an instruction instr_{s_k} have different influence on the result. Therefore, we distinguish the following cases depending on the type of instr_{s_k}.

1. instr_{s_k} is not a branching instruction with a condition of the form $Z_i \geqslant Z_j$ or an oracle instruction.

   Then, we take $U_k =_{\mathrm{df}} U_{k-1}$ and $P_k =_{\mathrm{df}} P_{k-1}$.
2. instr_{s_k} is a branching instruction s_k : *if* $Z_i \geqslant Z_j$ *then goto* λ_1 *else goto* λ_2.

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

Then we have $s_{k+1} = \lambda_1$ or $s_{k+1} = \lambda_2$ and the test condition is, for inputs $x \in \mathbb{R}$, of the form $k_v X + l_v \geqslant 0$. Let us consider a solution $y = \frac{l_v}{k_v} \in R_{\mathcal{F}}$ of $k_v y + l_v = 0$ in the case of $k_v \neq 0$. Then, for the interval J of solutions of $k_v X + l_v \geqslant 0$ (which satisfy the branching condition), there are two possibilities. We have $J = J_1 =_{\mathrm{df}}]-\infty, y]$ and then let $J_2 =_{\mathrm{df}}]y, \infty[$, or we have $J = J_1 =_{\mathrm{df}} [y, \infty[$, then let $J_2 =_{\mathrm{df}}]-\infty, y[$. For $k_v = 0$ and $l_v \geqslant 0$, let $J = J_1 =_{\mathrm{df}} \mathbb{R}$ and $J_2 =_{\mathrm{df}} \varnothing$. For $k_v = 0$ and $l_v < 0$, let $J = J_1 =_{\mathrm{df}} \varnothing$ and $J_2 =_{\mathrm{df}} \mathbb{R}$.
If $s_{k+1} = \lambda_1$, then $(U_{k-1} \cap J_1) \setminus R_{\mathcal{F}} \neq \varnothing$ holds and we have to put $U_k =_{\mathrm{df}} U_{k-1} \cap J_1$. If $s_{k+1} = \lambda_2$, then $(U_{k-1} \cap J_2) \setminus R_{\mathcal{F}} \neq \varnothing$ holds and we have to put $U_k =_{\mathrm{df}} U_{k-1} \cap J_2$. In both cases, we take $P_k =_{\mathrm{df}} P_{k-1}$.

3. instr_{s_k} is an oracle instruction s_k : *if* $(Z_1, \ldots, Z_{I_1}) \in \mathcal{O}$ *then goto* λ_1 *else goto* λ_2.

   a. The oracle query has not the form (8).
   Then we have $s_{k+1} = \lambda_2$. In this case, we need $U_k =_{\mathrm{df}} U_{k-1}$ and $P_k =_{\mathrm{df}} P_{k-1}$.

   b. The oracle query has the form (8). Then, we distinguish three further cases.

   i. We have $\mathcal{N}(k_1 x + l_1, \ldots, k_j x + l_j) \uparrow$ holds for all $x \in U_{k-1} \setminus R_{\mathcal{F}}$.
   Then we have $s_{k+1} = \lambda_2$.
   The reason is that, in this case, $\mathcal{N}(k_1 x + l_1, \ldots, k_j x + l_j) \downarrow$ could hold at most for a countable number of elements in U_{k-1} (since $R_{\mathcal{F}}$ is countable) and, consequently, $s_{k+1} = \lambda_1$ would imply that $U_k \subseteq U_{k-1} \cap R_{\mathcal{F}}$ would be countable. In such a case, U_l could be only countable. This contradicts the fact that B_0 is traversed for an uncountable set of inputs.
   In this case, let $U_k =_{\mathrm{df}} U_{k-1}$ and $P_k =_{\mathrm{df}} P_{k-1}$.

   ii. $\mathcal{N}(k_1 x_0 + l_1, \ldots, k_j x_0 + l_j) \downarrow$ holds for some $x_0 \in U_{k-1} \setminus R_{\mathcal{F}}$ and we have $s_{k+1} = \lambda_1$.
   Then by Lemma 6.2 there is an interval $J =]c, d[\subseteq U_{k-1}$ with $c < x_0 < d$ such that $\mathcal{N}(k_1 x + l_1, \ldots, k_j x + l_j) \downarrow$ for all $x \in J \setminus R_{\mathcal{F}}$.
   Altogether, there is a countable sequence of intervals $J_1, J_2, \ldots \subseteq \mathbb{R}$ such that we have $\mathcal{N}(k_1 x + l_1, \ldots, k_j x + l_j) \downarrow$ if and only if $x \in \bigcup_{i \geqslant 1} J_i$ holds. Hence $U_k =_{\mathrm{df}} U_{k-1} \cap \bigcup_{i \geqslant 1} J_i$ is also a countable union of intervals.
   Moreover, let $P_k =_{\mathrm{df}} P_{k-1}$.

   iii. $\mathcal{N}(k_1 x_0 + l_1, \ldots, k_j x_0 + l_j) \downarrow$ holds for some $x_0 \in U_{k-1} \setminus R_{\mathcal{F}}$ and we have $s_{k+1} = \lambda_2$.
   Then, let $U_k =_{\mathrm{df}} U_{k-1}$ and let $P_k =_{\mathrm{df}} P_{k-1} \cup H_{\mathcal{N}}$.

□

Thus, by Proposition 6.1, the following theorem is an immediate consequence of the latter proposition.

Theorem 6.1 (A property of problems in $\mathbb{R}_{\mathrm{add}}$-$\Delta_2^0$). If $P \subseteq \mathbb{R}$ belongs to $\mathbb{R}_{\mathrm{add}}$-$\Delta_2^0$, then there are $a, b, r_1, \ldots, r_i \in \mathbb{R}$ and a set P_0 that is semi-decidable over the structure $(\mathbb{R}; r_1, \ldots, r_i; +, -; \geqslant)$ such that the set $M = [a, b] \setminus (R_{\mathcal{F}} \cup P_0)$ is uncountable and we have either $M \subseteq P$ or $M \cap P = \varnothing$.

Let us transfer this result to the BSS model over the ordered ring of reals. Since, for any polynomial $p \in \mathbb{R}[x]$ that is not the zero polynomial, we have one of the three cases that either one of the inequalities $p(x) > 0$ or $p(x) < 0$ is satisfied for all $x \in \mathbb{R}$ or that

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

there is a finite number of intervals such that $p(x) > 0$ holds only for all elements of these intervals. $p(x) \neq 0$ is even satisfied by almost all real numbers. For any machine $\mathcal{M}$ with the constants in $\vec{r}$, let $N_{\vec{r}} = \{x \mid (\exists p \in \mathbb{Z}[r_1, \ldots, r_i][x])(p(x) = 0 \ \& \ p \not\equiv \mathbf{0})\}$. By means of this set, it is possible to formulate a Query Lemma for the ring of reals where, for any path of computation of $\mathcal{M}$ on inputs $x \in \mathbb{R} \setminus N_{\vec{r}}$, the critical queries are of the uniform form

$$((p_1(x), \ldots, p_j(x)) \,.\, c_{\mathcal{N}}) \in \mathbb{H}^1_{\text{add}}?$$

with $p_1, \ldots, p_j \in \mathbb{Z}[r_1, \ldots, r_i][x]$.

Proposition 6.5. Let $\mathcal{M} \in \mathsf{M}(\mathbb{H})$ be an oracle machine over $(\mathbb{R}; \mathbb{R}; \cdot, +, -; \geqslant)$ deciding a problem $\subseteq \mathbb{R}$ by means of the constants $r_1, \ldots, r_i$ and let $[a, b] \subseteq \mathbb{R}$ be an interval with $-\infty \leqslant a < b \leqslant \infty$. Then, there is an interval $I \subseteq [a, b]$ and a set P semi-decidable over $(\mathbb{R}; r_1, \ldots, r_i; \cdot, +, -; \geqslant)$ such that $I \setminus (N_{\vec{r}} \cup P)$ is uncountable and $\mathcal{M}$ either rejects all inputs in $I \setminus (N_{\vec{r}} \cup P)$ or accepts all these inputs.

Theorem 6.2 (A property of problems in $\mathbb{R}$-Δ^0_2). If $P \subseteq \mathbb{R}$ belongs to $\mathbb{R}$-Δ^0_2, then there are $a, b, r_1, \ldots, r_i \in \mathbb{R}$ and a set P_0 that is semi-decidable over the structure $(\mathbb{R}; r_1, \ldots, r_i; \cdot, +, -; \geqslant)$ such that $M = [a, b] \setminus (N_{\vec{r}} \cup P_0)$ is uncountable and $M \subseteq P$ or $M \cap P = \varnothing$.

Because it is not easy to find simple application examples for Theorem 6.1 and Theorem 6.2, we want to focus here on oracle sets such as the halting problems $(\mathbb{H}^{1,=}_{\text{add}})^{(n)}$ over the structure $\mathbb{R}^{1,=}_{\text{add}}$. It is not difficult to see that every problem $P \subseteq \mathbb{R}$ that is semi-decidable by a machine $\mathcal{M}$ over $\mathbb{R}^{=}_{\text{add}}$, is a subset of $R_{\mathcal{M}}$ or includes $\mathbb{R} \setminus R_{\mathcal{M}}$. We will show the more general statement given in Proposition 6.6 by using the following lemma.

Lemma 6.4 (Computation paths and co-enumerability). Let $\vec{r} \in \mathbb{R}^\infty$.

a. If $\mathcal{N}(k_1 x_0 + l_1, \ldots, k_j x_0 + l_j) \downarrow$ for some $j, k_i \in \mathbb{Z}$, $l_i \in L_{\vec{r}}$, $x_0 \in \mathbb{R} \setminus R_{\vec{r}}$, and $\mathcal{N} \in \mathsf{M}^{1,=}_{\text{add}}$, then $\mathcal{N}(k_1 x + l_1, \ldots, k_j x + l_j) \downarrow$ holds for all $x \in \mathbb{R} \setminus R_{\vec{r}}$.
b. If $\mathcal{N}(k_1 x_0 + l_1, \ldots, k_j x_0 + l_j) \uparrow$ for some $j, k_i \in \mathbb{Z}$, $l_i \in L_{\vec{r}}$, $x_0 \in \mathbb{R} \setminus R_{\vec{r}}$, and $\mathcal{N} \in \mathsf{M}^{1,=}_{\text{add}}$, then $\mathcal{N}(k_1 x + l_1, \ldots, k_j x + l_j) \uparrow$ holds for all $x \in \mathbb{R} \setminus R_{\vec{r}}$.

Proposition 6.6 (Semi-decidability and co-enumerability). If $\mathcal{M} \in \mathsf{M}^{=}_{\text{add}}(\mathbb{H}^{=}_{\text{add}})$ is a machine semi-deciding a problem $\subseteq \mathbb{R}$, then $\mathcal{M}$ accepts all inputs in $\mathbb{R} \setminus R_{\vec{r}}$ or $\mathcal{M}$ halts for none of these inputs.

Proof. As in the proof of Proposition 6.4, we can consider a finite or an infinite path B_0 through which $\mathcal{M} \in \mathsf{M}^{=}_{\text{add}}(\mathbb{H}^{=}_{\text{add}})$ goes for an uncountable set $\subseteq \mathbb{R}$ of inputs. Then, there is a corresponding finite or infinite system S_{B_0} of conditions of the forms

$$\begin{aligned} k_\mu X + l_\mu &\neq 0, \\ ((k_1 X + l_1, \ldots, k_j X + l_j) \,.\, c_{\mathcal{N}}) &\in \mathbb{H}^{1,=}_{\text{add}}, \\ ((k_1 X + l_1, \ldots, k_j X + l_j) \,.\, c_{\mathcal{N}}) &\notin \mathbb{H}^{1,=}_{\text{add}} \end{aligned} \tag{10}$$

($k_i \in \mathbb{Z}, l_i \in L_{\vec{r}}$, $\mathcal{N} \in \mathsf{M}^{1,=}_{\text{add}}$) that is satisfied by an input $x \in \mathbb{R} \setminus R_{\vec{r}}$ if and only if this path B_0 is traversed by $\mathcal{M}$ on x. Since, in the system S_{B_0}, we have the conditions of the form (10) instead of (9) and we can use Lemma 6.4, we can take $I = \mathbb{R}$ and P remains

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

the empty set $\varnothing$ (since the case b.iii is here not possible). Thus, we get the assumption in analogy with the proof of Proposition 6.4. □

Based on these considerations, we can continue the characterization of the classes $\mathbb{IR}^{=}_{\text{add}}$-$\Sigma^0_n$.

Proposition 6.7. Let $n \geqslant 1$. If $\mathcal{M} \in \mathsf{M}^{=}_{\text{add}}((\mathbb{IH}^{=}_{\text{add}})^{(n-1)})$ is a machine semi-deciding a problem $\subseteq \mathbb{IR}$, then $\mathcal{M}$ accepts all inputs in $\mathbb{IR} \setminus R_{\vec{r}}$ or $\mathcal{M}$ halts for none of these inputs.

Theorem 6.3 (A property of problems in $\mathbb{IR}^{=}_{\text{add}}$-$\Sigma^0_n$). Let $n \geqslant 1$. If $P \subseteq \mathbb{IR}$ belongs to $\mathbb{IR}^{=}_{\text{add}}$-$\Sigma^0_n$, then we have $\mathbb{IR} \setminus R_{\vec{r}} \subseteq P$ or $(\mathbb{IR} \setminus R_{\vec{r}}) \cap P = \varnothing$.

Corollary 6.3. If $P \subseteq \mathbb{IR}$ belongs to the hierarchy $\bigcup_{n\geqslant 1} \Sigma^0_n$ over $\mathbb{IR}^{=}_{\text{add}}$, then we have $\mathbb{IR} \setminus R_{\vec{r}} \subseteq P$ or $(\mathbb{IR} \setminus R_{\vec{r}}) \cap P = \varnothing$.

We consider once more the problems $\mathbb{P}_{\text{sqrt}} \subseteq \mathbb{A}_{\text{alg}}$. By Lemma 6.3, we know that there are prime numbers p with $\sqrt{p} \notin R_{\vec{r}}$ for any $\vec{r} \in \mathbb{IR}^{\infty}$. Because of $\mathbb{P}_{\text{sqrt}} \nsubseteq R_{\vec{r}}$ and $\mathbb{A}_{\text{alg}} \nsubseteq R_{\vec{r}}$, we get the following by Corollary 6.3.

Corollary 6.4. The problems $\mathbb{P}_{\text{sqrt}}$ and $\mathbb{A}_{\text{alg}}$ do not belong to the hierarchy $\bigcup_{n\geqslant 1} \Sigma^0_n$ over $\mathbb{IR}^{=}_{\text{add}}$.

Moreover, analogously to the proof of Proposition 6.4 we can show the following.

Proposition 6.8. Let $\mathcal{M} \in \mathsf{M}_{\text{add}}(\mathcal{O})$ with $\mathcal{O} \in \bigcup_{n\geqslant 1} \mathbb{IR}^{=}_{\text{add}}$-$\Sigma^0_n$ be a machine semi-deciding a problem $P \subseteq \mathbb{IR}$ with $P \setminus R_{\mathcal{M}} \neq \varnothing$. Then, there is an interval $I \subseteq \mathbb{IR}$ such that $\mathcal{M}$ accepts all inputs in $I \setminus R_{\mathcal{M}}$.

Proof. Let us assume that P is semi-decidable by $\mathcal{M} \in \mathsf{M}_{\text{add}}(\mathcal{O})$ and let $x \in P \setminus R_{\mathcal{M}}$. Similar to the proof of Proposition 6.4, we can show that there is an interval I with $x \in I$ such that all elements of $I \setminus R_{\mathcal{M}}$ have to be accepted by $\mathcal{M}$. Here, we consider the computation path $s_1, s_2, \ldots, s_l$ which is traversed by $\mathcal{M}$ on x. In order to determine $U_l \setminus (R_{\mathcal{M}} \cup P_l)$, we can take $P_l = \varnothing$ and for defining a suitable sequence $U_1 \supseteq U_2 \supseteq \cdots \supseteq U_l$ we work as in the proof of Proposition 6.4 if instr_{s_k} is not a branching instruction and, in the case that instr_{s_k} is a branching instruction s_k : *if* $Z_i \geqslant Z_j$ *then goto* λ_1 *else goto* λ_2, we have $s_{k+1} = \lambda_m$ for some $m \in \{1, 2\}$ if $x \in J_m$ and thus in this case we take $U_k =_{\text{df}} U_{k-1} \cap J_m$ (for the definitions of J_1 and J_2 see above). □

Thus, by Lemma 6.3, we have the following two statements.

Corollary 6.5. The set $\mathbb{P}_{\text{sqrt}}$ is not semi-decidable by a machine in $\mathsf{M}_{\text{add}}(\mathcal{O})$ with an oracle set $\mathcal{O} \in \bigcup_{n\geqslant 1} \mathbb{IR}^{=}_{\text{add}}$-$\Sigma^0_n$.

Corollary 6.6. The set $\mathbb{A}_{\text{alg}}$ is not semi-decidable by a machine in $\mathsf{M}_{\text{add}}(\mathcal{O})$ with an oracle set $\mathcal{O} \in \bigcup_{n\geqslant 1} \mathbb{IR}^{=}_{\text{add}}$-$\Sigma^0_n$.

7. Summary and outlook

For structures $\mathcal{A}$ with a finite number of operations and relations and with all individuals as constants for which there is a suitable effectively enumerable infinite set, we gave

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

complete problems for $\mathcal{A}$-Σ_2^0 and $\mathcal{A}$-Π_2^0 and we presented some common features of the classes $\mathcal{A}$-Δ_2^0. Moreover, we considered the algebraic decision problems $\mathbb{P}_{\text{sqrt}}$ and $\mathbb{A}_{\text{alg}}$ and assigned them to classes of undecidable problems with respect to several structures over the reals:

- $\mathbb{P}_{\text{sqrt}}$ is in $\mathbb{R}$-Δ_1^0, in $\mathbb{R}^{=}$-Σ_1^0, and in $\mathbb{R}_{\text{add}}$-Π_1^0.
- $\mathbb{P}_{\text{sqrt}}$ is not in $\mathbb{R}_{\text{add}}$-Σ_1^0 and not in $\mathbb{R}_{\text{add}}^{=}$-$\Sigma_n^0$.
- $\mathbb{P}_{\text{sqrt}}$ is not semi-decidable by a machine in $\mathsf{M}_{\text{add}}((\mathbb{H}_{\text{add}}^{=})^{(n)})$.
- $\mathbb{A}_{\text{alg}}$ is in $\mathbb{R}^{=}$-Σ_1^0 and in $\mathbb{R}_{\text{add}}$-Σ_3^0.
- $\mathbb{A}_{\text{alg}}$ is not in $\mathbb{R}_{\text{add}}$-Σ_1^0 and not in $\mathbb{R}_{\text{add}}^{=}$-$\Sigma_n^0$.
- $\mathbb{A}_{\text{alg}}$ is not semi-decidable by a machine in $\mathsf{M}_{\text{add}}((\mathbb{H}_{\text{add}}^{=})^{(n)})$.

In the first part of the paper, we used that there is an effectively enumerable set denoted by $\bar{\mathbb{N}}$. However, it would also be possible to use a set such as $\{(\bar{0},\bar{1}),(\bar{0},\bar{0},\bar{1}),\ldots\}$ for two constants $\bar{0}$ and $\bar{1}$ in $\mathcal{A}$ instead of $\bar{\mathbb{N}}$ in order to define complete problems similar to the problems presented here. In such a case, we would take $((\underbrace{\bar{0},\ldots,\bar{0}}_{n},\bar{0},\bar{1})\,.\,\vec{x})$ instead of $(\bar{n}\,.\,\vec{x})$ and so on.

I am very grateful to the anonymous referees for the detailed feedback and their valuable suggestions and comments which helped to improve the quality of this article. My thanks also go to Pedro Francisco Valencia Vizcaíno for proofreading the final manuscript.

References

Asveld, P.R.J. and Tucker, J.V. (1982). Complexity theory and the operational structure of algebraic programming systems. *Acta Informatica* **17** 451–476. http://dx.doi.org/10.1007/BF00264163.

Blum, L., Cucker, F., Shub, M. and Smale, S. (1998). *Complexity and Real Computation*, Springer.

Blum, L., Shub, M. and Smale, S. (1989). On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society* **21** 1–46. http://projecteuclid.org/euclid.bams/1183555121.

Börger, E. (1992). *Berechenbarkeit, Komplexität, Logik*, Vieweg.

Bournez, O., Cucker, F., de Naurois, P.J. and Marion, J.-Y. (2003). Computability over an arbitrary structure: Sequential and parallel polynomial time. In: *Proceeding FOSSACS'03/ETAPS'03* 185–199. http://dx.doi.org/10.1007/3-540-36576-1_12.

Bournez, O., Cucker, F., de Naurois, P.J. and Marion, J.-Y. (2006). Implicit complexity over an arbitrary structure: Quantifier alternations. *Information and Computation* **202** (2) 210–230. http://dx.doi.org/10.1016/j.ic.2005.07.005

Calvert, W., Kramer, K. and Miller, R. (2011). Noncomputable functions in the Blum-Shub-Smale model. Special Issue for the *"Seventh International Conference on Computability and Complexity in Analysis (CCA 2010)"*; Logical Methods in Computer Science **7** (2:15) (2011), 1–20. http://dx.doi.org/10.2168/LMCS-7(2:15)2011.

Cucker, F. (1992). The arithmetical hierarchy over the reals. *Journal of Logic and Computation* **2**(3) 375–395. http://dx.doi.org/10.1093/logcom/2.3.375.

Cucker, F. and Koiran, P. (1995). Computing over the reals with addition and order: Higher complexity classes. *Journal of Complexity* **11** 358–376. http://dx.doi.org/10.1006/jcom.1995.1018.

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press

Dobkin, D. and Lipton, R.J. (1978). A lower bound of $\frac{1}{2}n^2$ on linear search programs for the knapsack problem. *Journal of Computer and System Sciences* **16** (3) 413–417. http://dx.doi.org/10.1016/0022-0000(78)90026-0.

Friedman, H. and Mansfield, R. (1992). Algorithmic procedure. *Transactions of the AMS* **332** 297–312.

Gaßner, C. (1997). On NP-completeness for linear machines. *Journal of Complexity* **13** 259–271. http://dx.doi.org/10.1006/jcom.1997.0444.

Gaßner, C. (2009). Oracles and relativizations of the P =? NP question for several structures. *Journal of Universal Computer Science* **15** (6) 1186–1205. http://dx.doi.org/10.3217/jucs-015-06-1186.

Gaßner, C. (2013). Strong Turing degrees for additive BSS RAM's. *Logical Methods in Computer Science* **9** (4:25) 1–18. http://dx.doi.org/10.2168/LMCS-9(4:25)2013.

Grädel, E. (2007). Finite model theory and descriptive complexity. In: Grädel, E., Kolaitis, P.G., Libkin, L., Marx, M., Spencer, J., Vardi, M.Y., Venema, Y. and Weinstein, S. (eds.) *Finite Model Theory and Its Applications*, Springer, 125–230.

Hemmerling, A. (1998). Computability of string functions over algebraic structures. *Mathematical Logic Quarterly* **44** 1–44. http://dx.doi.org/10.1002/malq.19980440102.

Kleene, S.C. (1952). *Introduction to Metamathematics*, North-Holland Publ. Co.

Koiran, P. (1995). Computing over the reals with addition and order. *Theoretical Computer Science* **133** 35–47. http://dx.doi.org/10.1016/0304-3975(93)00063-B.

Meer, K. and Ziegler, M. (2008). An explicit solution to post's problem over the reals. *Journal of Complexity* **24** 3–15. http://dx.doi.org/10.1016/j.jco.2006.09.004.

Moschovakis, Y.N. (1969). Abstract first order computability. I. *Transactions of the American Mathematical Society* **138** 427–464. http://dx.doi.org/10.2307/1994926.

Moschovakis, Y.N. (1980). *Descriptive set theory. Studies in Logic and the Foundations of Mathematics* 100, North-Holland.

Poizat, B. (1995). *Les Petits Cailloux*. Aléas.

Preparata, F.P. and Shamos, M.I. (1985). *Computational Geometry: An Introduction*, Springer.

Scott, D. (1967). Some definitional suggestions for automata theory. *Journal of Computer and System Sciences* **1** (2) 187–212. http://dx.doi.org/10.1016/S0022-0000(67)80014-X.

Soare, R.I. (1987). *Recursively Enumerable Sets and Degrees: A Study of Computable Functions and Computably Generated Sets*, Springer.

Spreen, D. (1995). On some decision problems in programming. *Information and Computation* **122** 120–139. http://dx.doi.org/10.1006/inco.1995.1143. Corrigendum ibid. **148** (1999) 241–244. http://dx.doi.org/10.1006/inco.1998.2758

Spreen, D. (2010). Every Δ_0^2-set is natural, up to Turing equivalence. In: Ferreira F. et al., (eds.) CiE2010, Lecture Notes in Computer Science 6158, Springer (2010), 386–393. http://dx.doi.org/10.1007/978-3-642-13962-8_43

Tavana, N. and Weihrauch, K. (2011). Turing machines on represented sets, a model of computation for analysis. *Logical Methods in Computer Science* **7** (2) 1–21. http://dx.doi.org/10.2168/LMCS-7(2:19)2011.

Tucker, J.V. and Zucker, J.I. (2001). Computable functions and semicomputable sets on many-sorted algebras. *Handbook of Logic in Computer Science* **5** 397–525.

https://doi.org/10.1017/S0960129516000116 Published online by Cambridge University Press