Journal of Universal Computer Science, vol. 15, no. 6 (2009), 1186-1205
submitted: 3/11/08, accepted: 4/2/09, appeared: 28/3/09 © *J.UCS*

Oracles and Relativizations of the P =? NP Question for Several Structures

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

Abstract: We consider the uniform model of computation over any structure with two constants. For several structures, we construct oracles which imply that the relativized versions of P and NP are equal or are not equal. We construct universal oracles which imply the equality of the relativized versions of P and NP and we show that we lose the possibility to define these oracles recursively if we try to compress their elements to tuples of fixed length. Moreover we give new oracles for the BSS model in order to separate the classes P and NP relative to these oracles.
Key Words: BSS machines, oracle machines, relativizations, P-NP problem, Halting Problem
Category: F.1, F.1.1, F.1.2, F.1.3

1 Introduction

The uniform model of computation over arbitrary algebraic structures $\mathbb{K}$ can be defined in analogy to the BSS model over the real numbers introduced by L. Blum, M. Shub, and S. Smale in [Blum et al. 1989] (see also [Blum et al. 1998]). For the structure $\mathbb{K}_{\{0,1\}} =_{\mathrm{df}} (\{0,1\};0,1;;=)$ which is also the basic structure for Turing machines (cf. [Balcázar et al. 1988/90]) and for structures like the ordered ring of reals used in case of the BSS model, questions like $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ are open. For the classical setting, T. Baker, J. Gill, and R. Solovay constructed relativized versions of P and NP which imply different relationships between these classes (cf. [Baker et al. 1975]). There are oracles $\mathcal{O}$ such that the classes $\mathrm{P}^{\mathcal{O}}$ and $\mathrm{NP}^{\mathcal{O}}$ are equal and other oracles such that they are not equal. T. Emerson transferred these results to the ring of reals and other ordered rings in [Emerson 1994]. In the classical setting, the proofs rely on the enumerability of the programs of oracle machines. Emerson introduced oracles of a new kind where he used the codes of BSS machines as specified in [Blum et al. 1989]. In this way the authors showed that, in both settings, for Turing machines as well as for BSS machines, the extension of the machines by oracles is not very helpful for solving the central problems like $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$. This implies questions like the following for any structures $\mathbb{K}$: Which relationships between the relativized versions of $\mathrm{P}_{\mathbb{K}}$ and $\mathrm{NP}_{\mathbb{K}}$ will we obtain if we permit oracles for machines over $\mathbb{K}$? Can we provide evidence that the construction of new oracles is not really helpful for solving the $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ problem, by defining oracles $\mathcal{O}$ and $\mathcal{Q}$ satisfying $\mathrm{P}^{\mathcal{O}}_{\mathbb{K}} = \mathrm{NP}^{\mathcal{O}}_{\mathbb{K}}$

and $\mathrm{P}_{\mathbb{K}}^{\mathcal{Q}} \neq \mathrm{NP}_{\mathbb{K}}^{\mathcal{Q}}$ for structures for which the relation between $\mathrm{P}_{\mathbb{K}}$ and $\mathrm{NP}_{\mathbb{K}}$ is known? Is it possible to derive new relations from these oracles in order to get $\mathrm{P}_{\mathbb{M}} = \mathrm{NP}_{\mathbb{M}}$ for new structures $\mathbb{M}$?

2 The Model of Computation

Let $\mathsf{struc}(U)$ be the class of structures $\mathbb{K} = (U; (d_j)_{j\in J_0}; (f_j)_{j\in J_1}; (R_j)_{j\in J_2}, =)$ 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 $n_{f_j} \geq 1$. For $j \in J_2$, R_j is a relation of arity n_{R_j}. For any $\mathbb{K} \in \mathrm{struc}(U)$, we define the machines over $\mathbb{K}$, the $\mathbb{K}$-machines, in analogy to [Blum et al. 1989] such that we get a natural format of abstract computers over $\mathbb{K}$, on the one hand, and such that one has to consider only a small number of kinds of instructions, on the other hand.

Every $\mathbb{K}$-*machine* $\mathcal{M}$ is equipped with registers $Z_1, Z_2, \ldots$ for the elements of U and with a fixed number of registers $I_1, I_2, \ldots, I_{k_{\mathcal{M}}}$ for indices in $\mathbb{N}^+ = \mathbb{N}\backslash\{0\}$. For an input $(x_1, \ldots, x_n) \in U^\infty =_{\mathrm{df}} \bigcup_{i=1}^{\infty} U^i$, the sequence $x_1, \ldots, x_n, x_n, x_n, \ldots$ is assigned to the registers $Z_1, Z_2, \ldots$. The index registers get the content n. After the input the machine executes its *program* defined by a finite sequence of labelled instructions until an output instruction is reached. 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 $Z_j = Z_k$ or $R_k(Z_{j_1}, \ldots, Z_{j_{n_{R_k}}})$. The $\mathbb{K}$-machines perform these instructions as a computer where we assume that each function and each relation of $\mathbb{K}$ can be processed within a fixed time unit. The index registers are used in the copy instructions. For useful copying the value from Z_{I_k} into Z_{I_j}, we also allow $I_j := 1$, $I_j := I_j + 1$, and *if* $I_j = I_k$ *then goto* l_1 *else goto* l_2. Moreover, *oracle machines* can execute *if* $(Z_1, \ldots, Z_{I_1}) \in \mathcal{O}$ *then goto* l_1 *else goto* l_2 for some oracle $\mathcal{O} \subseteq U^\infty$. The *non-deterministic* machines are able to guess an arbitrary number of arbitrary elements $y_1, \ldots, y_m \in U$ in one step after the input and to assign the guesses to $Z_{I_1+1}, \ldots, Z_{I_1+m}$. We do not restrict the domain for m to simplify matters. That means that m is independent of n. However, a machine can use at most t guesses within t steps. In any case, the *size* of an input $(x_1, \ldots, x_n)$ is, by definition, its length n. If the *output* instruction is reached, then $(Z_1, \ldots, Z_{I_1})$ is the output and the machine halts.

Let $\mathsf{M}_{\mathbb{K}}$ and $\mathsf{M}_{\mathbb{K}}^{\mathrm{N}}$ be the classes of deterministic and non-deterministic $\mathbb{K}$-machines, respectively. Let, moreover, the machines in $\mathsf{M}_{\mathbb{K}}(\mathcal{O})$ and $\mathsf{M}_{\mathbb{K}}^{\mathrm{N}}(\mathcal{O})$ be able to use the oracle $\mathcal{O}$. Then, we have $\mathsf{M}_{\mathbb{K}} \subseteq \mathsf{M}_{\mathbb{K}}^{\mathrm{N}}$ and $\mathsf{M}_{\mathbb{K}}(\mathcal{O}) \subseteq \mathsf{M}_{\mathbb{K}}^{\mathrm{N}}(\mathcal{O})$ since we assume that non-deterministic machines need not guess any elements.

In the following we only consider structures containing two constants. We denote the class of these structures by $\mathsf{struc}_{a,b}(U)$ where we assume that a, b are two constants belonging to U. Then we say that a deterministic $\mathbb{K}$-machine *accepts* (or *rejects*, respectively) a tuple $\mathbf{x} \in U^\infty$ if the machine outputs a (or b, respectively) on input $\mathbf{x}$. For any *problem* $\mathcal{P} \subseteq U^\infty$, its *decidability* and its *recognition* (or *semi-decidability*) *over* $\mathbb{K}$ result from the computability of the characteristic function of $\mathcal{P}$, $f_\mathcal{P} : U^\infty \to \{a, b\}$, by some $\mathbb{K}$-machine and from the computability of the partial characteristic function of $\mathcal{P}$, $f_\mathcal{P} : U^\infty \to \{a\}$, respectively. A non-deterministic $\mathbb{K}$-machine $\mathcal{M}$ *accepts* an input $(x_1, \ldots, x_n) \in U^\infty$ if there is some finite sequence of guesses $(y_1, \ldots, y_m) \in U^\infty$ such that $\mathcal{M}$ outputs a on input $(x_1, \ldots, x_n)$ for the guesses $y_1, \ldots, y_m$. The execution of one instruction is one step of the computation process. Each step can be executed in a fixed time unit. A $\mathbb{K}$-machine will come to a halt *in polynomial time* if there is a polynomial function p such that, on every input $(x_1, \ldots, x_n) \in U^\infty$ (and for any guesses), the machine performs at most $p(n)$ instructions before the output is generated.

For any structure $\mathbb{K}$, let $\mathrm{P}_\mathbb{K}$ and $\mathrm{NP}_\mathbb{K}$ denote the usual complexity classes of decision problems $\mathcal{P} \subseteq U^\infty$ decided or non-deterministically recognized by a machine in $\mathsf{M}_\mathbb{K}$ or in $\mathsf{M}^\mathrm{N}_\mathbb{K}$ in polynomial time (where an input is only accepted if and only if it is in $\mathcal{P}$). $\mathrm{DEC}_\mathbb{K}$ contains all problems decided by a machine in $\mathsf{M}_\mathbb{K}$. Moreover, let $\mathrm{FP}_\mathbb{K}$ be the class of all functions $f : U^\infty \to U^\infty$ which can be computed by $\mathbb{K}$-machines in polynomial time. For any oracle $\mathcal{O}$, $\mathrm{P}^\mathcal{O}_\mathbb{K}$, $\mathrm{NP}^\mathcal{O}_\mathbb{K}$, and $\mathrm{DEC}^\mathcal{O}_\mathbb{K}$ denote the classes extended to machines which can also use $\mathcal{O}$.

Let $\mathsf{struc}^{\mathsf{fin}}_{a,b}(U)$ be the class of structures of finite signature which have the form $(U; a, b, d_3, \ldots, d_{k_0}; f_1, \ldots, f_{k_1}; R_1, \ldots, R_{k_2}, =)$ for some $k_0 \geq 2$ and $k_1, k_2 \geq 0$. For any structure $\mathbb{K} \in \mathsf{struc}^{\mathsf{fin}}_{a,b}(U)$, we can define *universal* deterministic and non-deterministic $\mathbb{K}$-machines which are able to simulate each machine $\mathcal{M} \in \mathsf{M}_\mathbb{K}$ and $\mathcal{M} \in \mathsf{M}^\mathrm{N}_\mathbb{K}$, respectively, on any input $\mathbf{x}$ if they get $\mathbf{x}$ and a suitable code of $\mathcal{M}$ as input. In order to encode the programs of machines by strings we use U or a subset of U as alphabet where this alphabet can also be infinite, e.g., if $U = \mathbb{R}$, then real numbers can be the symbols of strings. The concatenation of any strings $s_1, s_2 \in U^*$ is denoted by $s_1 s_2$, and for $r \in U^*$ and $\mathcal{S}, \mathcal{S}_1, \mathcal{S}_2 \subseteq U^*$, we have $\mathcal{S}_1\mathcal{S}_2 = \{s_1 s_2 \mid s_1 \in \mathcal{S}_1 \;\&\; s_2 \in \mathcal{S}_2\}$, $r\mathcal{S} = \{r\}\mathcal{S}$, and $\mathcal{S}r = \mathcal{S}\{r\}$. In any case, we want to distinguish between the product $U^{(=2)} =_\mathrm{df} UU = \{uv \mid u, v \in U\}$ defined by the concatenation of strings and the Cartesian product $U^2 = U \times U$. For this reason, we also distinguish between U^* and U^∞. $U^* = \bigcup_{i=0}^\infty U^{(=i)}$ is the set of all strings over U, U^∞ is the set of all tuples over U, and $(U^*)^\infty = \bigcup_{i=1}^\infty (U^*)^i$ is the set of all tuples whose components are strings over U.

Definition 1. *Let* $\mathcal{S}_\mathrm{code} =_\mathrm{df} b^2(\{a, b\}^* \setminus (\{a, b\}^* b^2 \{a, b\}^*))$ *be the set of strings in* $\{a, b\}^*$ *which contain the sub-string* b^2 *as prefix only. For* $\mathbb{K} \in \mathsf{struc}^{\mathsf{fin}}_{a,b}(U)$, *let* $Code^*_\mathbb{K}$ *be an injective mapping of the set of all deterministic and non-deterministic oracle* $\mathbb{K}$*-machines into* $\mathcal{S}_\mathrm{code}$ *such that every character of the programs is unambiguously translated into a string by this mapping where the oracle queries are encoded independent of the used oracle by taking the same* *sequence of characters as code for all oracle queries.*

The oracle queries can be encoded independently of the used oracle since we consider only classes of oracle machines using the same oracle. Note that we omit the index $\mathbb{K}$ since confusion is not to be expected. Since, in general, the strings over U are not elements of U, we use tuples as codes. Any $(c_1, \ldots, c_k) \in U^\infty$ can be stored in k registers.

Definition 2. *For every non-empty string $s = c_1 \cdots c_k \in U^*$ where $|s| = k \geq 1$, let $\lceil s \rceil_{\text{tuple}}$ and $\lceil s \rceil$ (we will omit the subscript) be the* representation of s in the form of a tuple $(c_1, \ldots, c_k) \in U^k \subset U^\infty$, *that means that* $\lceil c_1 \cdots c_k \rceil_{\text{tuple}} = \lceil c_1 \cdots c_k \rceil = (c_1, \ldots, c_k)$.

To simplify matters, we use the vector notation for tuples and for parts of tuples. $(\mathbf{x}, \lceil c_1 \cdots c_k \rceil)$ stands for $(x_1, \ldots, x_n, c_1, \ldots, c_k)$, and so on. Moreover, for any $t \geq 1$, $\tilde{t}$ stands for $\lceil b^t a \rceil$ and *Code* is defined by $Code(\mathcal{M}) = \lceil Code^*(\mathcal{M}) \rceil$ for any machine $\mathcal{M} \in \mathsf{M}^{\mathrm{N}}_{\mathbb{K}}(\mathcal{O})$ where $\mathbb{K} \in \mathsf{struc}^{\mathsf{fin}}_{a,b}(U)$.

Definition 3. *Let the* Universal $\mathrm{NP}_{\mathbb{K}}$-Problem, *the* Halting Problem, *and a* special halting problem with respect to $\mathbb{K} \in \mathsf{struc}^{\mathsf{fin}}_{a,b}(U)$ *be given by*

$$\mathrm{UNI}_{\mathbb{K}} = \{(\tilde{t}, \mathbf{x}, Code(\mathcal{M})) |\ \mathbf{x} \in U^\infty \ \&\ \mathcal{M} \in \mathsf{M}^{\mathrm{N}}_{\mathbb{K}} \ \&\ \mathcal{M} \text{ accepts } \mathbf{x} \text{ within } t \text{ steps}\},$$

$$\mathrm{H}_{\mathbb{K}} \ = \{(\mathbf{x}, Code(\mathcal{M})) \mid \mathbf{x} \in U^\infty \ \&\ \mathcal{M} \in \mathsf{M}_{\mathbb{K}} \ \&\ \mathcal{M} \text{ halts on } \mathbf{x}\},$$

$$\mathrm{H}^{\mathrm{spec}}_{\mathbb{K}} = \{Code(\mathcal{M}) \mid \mathcal{M} \in \mathsf{M}_{\mathbb{K}} \ \&\ \mathcal{M} \text{ halts on } Code(\mathcal{M})\}.$$

The first problem can be recognized by a universal non-deterministic machine in polynomial time since the size of an input

$$(\tilde{t}, \mathbf{x}, Code(\mathcal{M})) = (\underbrace{b, \ldots, b}_{t \times}, a, x_1, \ldots, x_n, \underbrace{b, b, c_3, \ldots, c_s}_{= Code(\mathcal{M})}) \in U^{t+1+n+s}$$

is $t + 1 + n + s$. We can generalize some known results.

Proposition 4. *For each structure* $\mathbb{K} \in \mathsf{struc}^{\mathsf{fin}}_{a,b}(U)$, $\mathrm{UNI}_{\mathbb{K}}$ *is* $\mathrm{NP}_{\mathbb{K}}$*-complete.*

Corollary 5. *For each structure* $\mathbb{K} \in \mathsf{struc}^{\mathsf{fin}}_{a,b}(U)$, *we have*

$$\mathrm{P}_{\mathbb{K}} = \mathrm{NP}_{\mathbb{K}} \ \textit{ if and only if } \ \mathrm{UNI}_{\mathbb{K}} \in \mathrm{P}_{\mathbb{K}},$$

$$\mathrm{P}_{\mathbb{K}} \neq \mathrm{NP}_{\mathbb{K}} \ \textit{ if } \ \mathrm{UNI}_{\mathbb{K}} \notin \mathrm{DEC}_{\mathbb{K}}.$$

Let us mention that the finite signature of a structure $\mathbb{K}$ is a sufficient but not a necessary assumption for the definition of $\mathrm{NP}_{\mathbb{K}}$-complete problems. For example, the linear $\mathbb{R}_{\mathrm{lin}}$-machines over the reals as well as the scalar $\mathbb{Z}_{\mathrm{sc}}$-machines over the integers, which can only execute the multiplication by constants, can be encoded if the constant factors stand for themselves in the codes. There is not a universal $\mathbb{R}_{\mathrm{lin}}$-machine and there is not any $\mathrm{NP}_{\mathbb{Z}_{\mathrm{sc}}}$-complete problem, but there are $\mathrm{NP}_{\mathbb{R}_{\mathrm{lin}}}$-complete problems (see [Gaßner 1997]).

The undecidability of the Halting Problem is known for Turing machines, for the BSS model over the real numbers, for *While* programs on standard algebras [Tucker and Zucker 2000], and so on. For these problems, the undecidability results from the enumerability of the codes of machines and the undecidability of halting sets investigated in [Blum et al. 1989, Blum et al. 1998], respectively. For BSS machines and restricted classes of BSS machines, further halting problems were considered, e.g., in [Meer and Ziegler 2005, Meer and Ziegler 2006] and in [Gaßner (1) 2008].

Proposition 6. *For any* $\mathbb{K} \in \mathsf{struc}^{\mathsf{fin}}_{a,b}(U)$*,* $\mathrm{H}_{\mathbb{K}} \in \mathrm{DEC}_{\mathbb{K}}$ *implies* $\mathrm{H}^{\mathrm{spec}}_{\mathbb{K}} \in \mathrm{DEC}_{\mathbb{K}}$*. If all elements of* $\mathbb{K}$ *are also constants of* $\mathbb{K}$*, then* $\mathrm{H}^{\mathrm{spec}}_{\mathbb{K}} \in \mathrm{DEC}_{\mathbb{K}}$ *implies* $\mathrm{H}_{\mathbb{K}} \in \mathrm{DEC}_{\mathbb{K}}$*.*

Proof. The first part of the proposition is obvious. For the proof of the second part, let us consider any deterministic $\mathbb{K}$-machine $\mathcal{N}$ and any input $\mathbf{x} = (x_1, \ldots, x_n)$. Let $\mathcal{M}_{\mathcal{N},\mathbf{x}}$ be the machine which works by means of its $k_{\mathcal{N}} + 2$ index registers $I_1, \ldots, I_{k_{\mathcal{N}}+2}$ as follows. First, $\mathcal{M}_{\mathcal{N},\mathbf{x}}$ replaces its own input values in the first n registers $Z_1, \ldots, Z_n$ by $x_1, \ldots, x_n$ and $\mathcal{M}_{\mathcal{N},\mathbf{x}}$ assigns the length n of $\mathbf{x}$ to its registers $I_1, \ldots, I_{k_{\mathcal{N}}+2}$. Then, this machine follows the program of $\mathcal{N}$ where the maximal value of the registers $I_1, \ldots, I_{k_{\mathcal{N}}}$ is also stored in $I_{k_{\mathcal{N}}+1}$ and $I_{k_{\mathcal{N}}+2}$ and $\mathcal{M}_{\mathcal{N},\mathbf{x}}$ writes x_n in each further register $Z_{n+1}, Z_{n+2}, \ldots$ by $I_{k_{\mathcal{N}}+2} := I_{k_{\mathcal{N}}+2} + 1;\ Z_{I_{k_{\mathcal{N}}+2}} := Z_{I_{k_{\mathcal{N}}+1}};\ I_{k_{\mathcal{N}}+1} := I_{k_{\mathcal{N}}+1} + 1$ before this register is used. Consequently, the output behaviour of $\mathcal{M}_{\mathcal{N},\mathbf{x}}$ on any input (also on input $Code(\mathcal{M}_{\mathcal{N},\mathbf{x}})$) is the same as one of $\mathcal{N}$ on $\mathbf{x}$. That means that $Code(\mathcal{M}_{\mathcal{N},\mathbf{x}}) \in \mathrm{H}^{\mathrm{spec}}_{\mathbb{K}}$ if and only if $(\mathbf{x}, Code(\mathcal{N})) \in \mathrm{H}_{\mathbb{K}}$. By definition of $Code$ there is a $\mathbb{K}$-machine which can compute $Code(\mathcal{M}_{\mathcal{N},\mathbf{x}})$ for any $\mathbf{x}$ and any $\mathcal{N}$ if it gets $(\mathbf{x}, Code(\mathcal{N}))$ as input. Thus, $\mathrm{H}_{\mathbb{K}}$ can be reduced to $\mathrm{H}^{\mathrm{spec}}_{\mathbb{K}}$, and, thus, $\mathrm{H}_{\mathbb{K}}$ is decidable by a $\mathbb{K}$-machine if $\mathrm{H}^{\mathrm{spec}}_{\mathbb{K}}$ is decidable by some $\mathbb{K}$-machine. □

Proposition 7. *For any* $\mathbb{K} \in \mathsf{struc}_{a,b}(U)$*, which allows to define a special halting problem analogously to* $\mathrm{H}^{\mathrm{spec}}_{\mathbb{K}}$ *by any encoding method, this special problem is not in* $\mathrm{DEC}_{\mathbb{K}}$*.*

Proof. We give the proof for $\mathrm{H}^{\mathrm{spec}}_{\mathbb{K}}$. Assume that there is a $\mathbb{K}$-machine $\mathcal{M}_0$ which decides $\mathrm{H}^{\mathrm{spec}}_{\mathbb{K}}$. Let $\mathcal{M}_1$ be the following machine. $\mathcal{M}_1$ works as $\mathcal{M}_0$ until the output instruction of $\mathcal{M}_0$ is reached, $\mathcal{M}_1$ does not halt if the output of $\mathcal{M}_0$ is a, and $\mathcal{M}_1$ halts if the output of $\mathcal{M}_0$ is b. That means, that $\mathcal{M}_1$ executes instructions like

$$l\colon \quad Z_2 := a;\ \text{if}\ Z_1 = Z_2\ \text{then goto}\ l;\ \text{output}\ Z_1$$

if $\mathcal{M}_0$ executes an output instruction of the form

$$l\colon \quad \text{output}\ Z_1.$$

Therefore, $\mathcal{M}_1$ halts on $Code(\mathcal{M}_1)$ if and only if the output of $\mathcal{M}_0$ on $Code(\mathcal{M}_1)$ is b, and consequently, if and only if $Code(\mathcal{M}_1)$ is not in $\mathrm{H}_{\mathbb{K}}^{\mathrm{spec}}$, and thus, if and only if $\mathcal{M}_1$ does not halt on $Code(\mathcal{M}_1)$. This is a contradiction.

The proof is the same one for any system for encoding machines. □

Remark. In the proof of Proposition 7 the existence of the constants a and b is a necessary assumption only if we want to use our definition of acceptance and rejection of inputs by means of the output of constants. The decidability of a problem could also be defined by the halting properties of two machines recognizing the problem and its complement, respectively. Then, we can prove Proposition 7 since the complement of $\mathrm{H}_{\mathbb{K}}^{\mathrm{spec}}$ cannot be recognized by any $\mathbb{K}$-machine.

Corollary 8. *For each* $\mathbb{K} \in \mathsf{struc}_{a,b}^{\mathsf{fin}}(U)$, $\mathrm{H}_{\mathbb{K}}$ *is not decidable by a* $\mathbb{K}$*-machine.*

3 The Equality of Relativized Versions of P and NP

We shall define a universal oracle $\mathcal{O}$ with $\mathrm{P}_{\mathbb{K}}^{\mathcal{O}} = \mathrm{NP}_{\mathbb{K}}^{\mathcal{O}}$ for any structure $\mathbb{K} \in \mathsf{struc}_{a,b}(U)$ which permits to compute the codes of the programs of machines over $\mathbb{K}$. The first construction is restricted to structures of finite signature with two constants. We transfer and modify the definitions given in [Baker et al. 1975] and [Emerson 1994]. The ideas for the definitions go also back to S. A. Cook, R. Karp, A. Meyer, M. Fischer, and H. B. Hunt. (For more details see [Baker et al. 1975].) The tuples which can occur in the oracles $\mathcal{O}_1(=\mathcal{O}_1^{(\mathbb{K})})$ and $\mathcal{O}_2(=\mathcal{O}_2^{(\mathbb{K})})$ (for a given $\mathbb{K} \in \mathsf{struc}_{a,b}(U)$, we omit the index $\mathbb{K}$) can be derived from a universal problem.

Definition 9. *For any* $\mathbb{K} \in \mathsf{struc}_{a,b}^{\mathsf{fin}}(U)$*, let*

$$\mathrm{UNI}_1^{(\mathbb{K})}(\mathcal{O}) = \{(\tilde{t}, \mathbf{x}, Code(\mathcal{M})) \mid \mathbf{x} \in U^{\infty}\ \&\ \mathcal{M} \in \mathsf{M}_{\mathbb{K}}^{\mathrm{N}}(\mathcal{O})\ \&\ \mathcal{M}(\mathbf{x}) \downarrow^t\}$$

be the Universal $\mathrm{NP}_{\mathbb{K}}^{\mathcal{O}}$-Problem *where* $\mathcal{M}(\mathbf{x}) \downarrow^t$ *means that* $\mathcal{M}$ *accepts* $\mathbf{x}$ *within* t *steps. Let* $\mathcal{O}_1(=\mathcal{O}_1^{(\mathbb{K})})$ *be a* universal oracle *defined by* $\mathcal{O}_1 = \bigcup_{i\geq 0} W_i$ *where* $W_0 = \emptyset$ *and*

$$W_i = \{(\tilde{t}, \mathbf{x}, Code(\mathcal{M})) \in U^i \mid \mathcal{M} \in \mathsf{M}_{\mathbb{K}}^{\mathrm{N}}(\bigcup_{j<i} W_j)\ \&\ \mathcal{M}(\mathbf{x}) \downarrow^t\}.$$

For any oracle $\mathcal{O}$, $\mathrm{UNI}_1^{(\mathbb{K})}(\mathcal{O})$ is $\mathrm{NP}_{\mathbb{K}}^{\mathcal{O}}$-complete since the codes of machines allow to simulate the single steps of the oracle machines in $\mathsf{M}_{\mathbb{K}}^{\mathrm{N}}(\mathcal{O})$ by only one universal oracle machine using the oracle $\mathcal{O}$ in polynomial time. Moreover, for any $i \geq 0$, we have $\mathrm{UNI}_1^{(\mathbb{K})}(\mathcal{O}_1) \cap U^i = \mathrm{UNI}_1^{(\mathbb{K})}(\bigcup_{j<i} W_j) \cap U^i = W_i$ since the length of a tuple in an oracle query, executed within the first t steps, is less than $t + n < i$ for any input $(x_1, \ldots, x_n)$. This implies $\mathrm{UNI}_1^{(\mathbb{K})}(\mathcal{O}_1) = \mathcal{O}_1$. Because of $\mathcal{O}_1 \in \mathrm{P}_{\mathbb{K}}^{\mathcal{O}_1}$ we get the following.

Proposition 10. *For any* $\mathbb{K} \in \mathsf{struc}_{a,b}^{\mathsf{fin}}(U)$*, there is an oracle* $\mathcal{O}$ *such that* $\mathrm{P}_{\mathbb{K}}^{\mathcal{O}} = \mathrm{NP}_{\mathbb{K}}^{\mathcal{O}}$.

A further characterization of the power of the universal oracle $\mathcal{O}_1$ is possible by comparison of the classes $\mathrm{P}_{\mathbb{K}}^{\mathcal{O}_1}$ and $\mathrm{NP}_{\mathbb{K}}^{\mathcal{O}_1}$ with the classes of the polynomial hierarchy $\mathrm{PH}_{\mathbb{K}}$ and the class $\mathrm{PAT}_{\mathbb{K}}$ containing the problems recognized in polynomial alternating time. Let us define the higher alternation levels of the polynomial hierarchy for any $\mathbb{K} \in \mathsf{struc}_{a,b}^{\mathsf{fin}}(U)$. A problem $\mathcal{P} \subseteq U^\infty$ is in $\Sigma_{\mathbb{K}}^k$ $(k \geq 0)$ iff there exist a problem $\mathcal{P}_0 \in \mathrm{P}_{\mathbb{K}}$ and polynomial functions $p_1, \ldots, p_k$ such that, for any $\mathbf{x} \in U^n$, the condition

$$\mathbf{x} \in \mathcal{P} \Leftrightarrow (Q_1 \mathbf{y}^{(1)} \in U^{p_1(n)}) \cdots (Q_k \mathbf{y}^{(k)} \in U^{p_k(n)})((\mathbf{x}, \mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(k)}) \in \mathcal{P}_0)$$

holds where Q_i stands for the existential quantifier $\exists$ if i is odd and for the universal quantifier $\forall$ if i is even. $\Pi_{\mathbb{K}}^k$ denotes the class obtained if the alternating quantifiers start with the universal one. The *polynomial hierarchy* is defined by $\mathrm{PH}_{\mathbb{K}} = \bigcup_{k \geq 0} \Sigma_{\mathbb{K}}^k = \bigcup_{k \geq 0} \Pi_{\mathbb{K}}^k$. A problem $\mathcal{P} \subseteq U^\infty$ belongs to *the class of the problems recognized in polynomial alternating time*, $\mathrm{PAT}_{\mathbb{K}}$, for $\mathbb{K} \in \mathsf{struc}_{a,b}^{\mathsf{fin}}(U)$ iff there exist a set $\mathcal{P}_0 \in \mathrm{P}_{\mathbb{K}}$ and some polynomial function q such that, for any $\mathbf{x} \in U^n$,

$$\begin{aligned} \mathbf{x} \in \mathcal{P} \Leftrightarrow (\exists y_1 \in U)(\forall z_1 \in U) \cdots (\exists y_{q(n)} \in U)(\forall z_{q(n)} \in U) \\ ((\mathbf{x}, y_1, z_1, \ldots, y_{q(n)}, z_{q(n)}) \in \mathcal{P}_0) \end{aligned} \tag{1}$$

holds.

For any $\mathbb{K} \in \mathsf{struc}_{a,b}^{\mathsf{fin}}(U)$, we know that $\mathrm{PH}_{\mathbb{K}} \subseteq \mathrm{PAT}_{\mathbb{K}}$ (for details, cf. [Cucker 1993, Bournez et al. 2006]). In the classical setting, we have $\mathrm{PAT} = \mathrm{PAT}_{\mathbb{K}_{\{0,1\}}} = \mathrm{PSPACE}$ (cf. [Balcázar et al. 1988/90, Bournez et al. 2006]).

Proposition 11. *For any structure* $\mathbb{K} \in \mathsf{struc}_{a,b}^{\mathsf{fin}}(U)$*, there holds*

$$\mathrm{PH}_{\mathbb{K}} \subseteq \mathrm{PAT}_{\mathbb{K}} \subseteq \mathrm{P}_{\mathbb{K}}^{\mathcal{O}_1}$$

for the universal oracle $\mathcal{O}_1$.

Proof. It is enough to show that any problem in $\mathrm{PAT}_{\mathbb{K}}$ can be reduced to $\mathcal{O}_1$ by some function in $\mathrm{FP}_{\mathbb{K}}$. We want to use the logical equivalence $\exists y_i \forall z_i H \leftrightarrow \exists y_i \neg(\exists z_i \neg H)$. Let $\mathcal{P}$ be any problem defined by some problem $\mathcal{P}_0 \in \mathrm{P}_{\mathbb{K}}$ and some polynomial function q such that (1) is satisfied for any $\mathbf{x} \in U^n$. Let $\mathcal{M}_0$ decide the problem $\mathcal{P}_0$ without using guesses in a time bounded by some polynomial function p. Then, $\mathcal{M}_1$ recognizes $\mathcal{P}$ non-deterministically if, for $i < q(n)$, the machines $\mathcal{M}_i$ and $\mathcal{N}_i$ and the machine $\mathcal{M}_{q(n)}$ use the programs given below and if $\mathcal{N}_{q(n)}$ outputs a on input $\mathbf{y}^{(q(n))}$ for any guess $z_{q(n)}$ if and only if $(\lceil b^{p(n+2q(n))}a\rceil, \mathbf{y}^{(q(n))}, z_{q(n)}, Code(\mathcal{M}_0)) \notin \mathcal{O}_1$. $\mathcal{N}_{q(n)}$ can simulate $\mathcal{M}_0$ or it can query the oracle. In the latter setting $\mathcal{N}_{q(n)}$ computes the code $Code(\mathcal{M}_0)$ and it assigns the constant b to the first $p(n+2q(n))$ registers before it queries the oracle. The number of steps, $s_{q(n)}$, executed by $\mathcal{N}_{q(n)}$ is bounded by some linear combination of $|Code^*(\mathcal{M}_0)|$ and $p(n+2q(n))$. Thus, the program of $\mathcal{N}_{q(n)}$ has the time complexity $O(p(k))$ and we can specify the program of $\mathcal{N}_{q(n)}$ so that the complexity of its length is only dependent on the degree and the coefficients of p. The behaviour of $\mathcal{N}_{q(n)}$ on input $\mathbf{y}^{(q(n))}$ is queried by $\mathcal{M}_{q(n)}$, and so on. For $i \leq q(n)$, the number of steps t_i which are done by $\mathcal{M}_i$ in order to execute its program is bounded by a linear combination of $|Code^*(\mathcal{N}_i)|$ and s_i. For $i < q(n)$, s_i is bounded by a linear combination of $|Code^*(\mathcal{M}_{i+1})|$ and t_{i+1}. Therefore, there is a reduction function in $\mathrm{FP}_{\mathbb{K}}$ which assigns the code of the non-deterministic oracle machine $\mathcal{M}_1$ where this code implicitly determines the code of the next oracle machine $\mathcal{N}_1$, and so on, to any input $\mathbf{x} \in U^n \subset U^\infty$.

The program of $\mathcal{M}_i$.

1: Input $\mathbf{z}^{(i-1)} =_{\mathrm{df}} (\mathbf{x}, y_1, z_1, \ldots, y_{i-1}, z_{i-1})$; guess y_i;
2: compute $(\tilde{s}_i, Code(\mathcal{N}_i))$;
3: if $(\tilde{s}_i, \mathbf{z}^{(i-1)}, y_i, Code(\mathcal{N}_i)) \in \mathcal{O}_1$ then goto 4 else goto 5;
4: output b;
5: output a.

The program of $\mathcal{N}_i$.

1: Input $\mathbf{y}^{(i)} =_{\mathrm{df}} (\mathbf{x}, y_1, z_1, y_2, \ldots, z_{i-1}, y_i)$; guess z_i;
2: compute $(\tilde{t}_{i+1}, Code(\mathcal{M}_{i+1}))$;
3: if $(\tilde{t}_{i+1}, \mathbf{y}^{(i)}, z_i, Code(\mathcal{M}_{i+1})) \in \mathcal{O}_1$ then goto 4 else goto 5;
4: output b;
5: output a. □

A statement like Proposition 10 can be formulated for any structure $\mathbb{K} \in \mathsf{struc}_{a,b}(U)$ if every oracle machine can be encoded by a computable tuple $\mathbf{u} \in \mathcal{V} =_{\mathrm{df}} \{\lceil v\rceil \in U^\infty \mid v \in b^2(U^* \setminus (U^* b^2 U^*))\}$. For structures of enumerable signature, the possible codes are the indices of a list of all programs like in the definition in [Baker et al. 1975], or they can have a form like the codes of the linear or scalar real machines, where the operations are encoded by real numbers, and so like. In this way we get the wished oracles also for many structures of infinite signature. Let, for any oracle $\mathcal{O}$,

$$\begin{aligned}\mathrm{UNI}_2^{(\mathbb{K})}(\mathcal{O}) = \{(\tilde{t}, \mathbf{x}, \mathbf{u}) \in U^\infty \mid \\ \mathbf{u} \in \mathcal{V} \;\&\; (\exists \mathcal{M} \in \mathsf{M}_{\mathbb{K}}^{\mathrm{N}}(\mathcal{O}))(\mathbf{u} \text{ is the code of } \mathcal{M} \;\&\; \mathcal{M}(\mathbf{x}) \downarrow^t)\}\end{aligned}$$

be a universal problem restricted to non-deterministic $\mathbb{K}$-machines which can use $\mathcal{O}$. $\mathrm{UNI}_2^{(\mathbb{K})}(\mathcal{O})$ is $\mathrm{NP}_{\mathbb{K}}^{\mathcal{O}}$-hard if every code of a machine $\mathcal{M}$ can be computed by a deterministic $\mathbb{K}$-machine $\mathcal{N}_{\mathcal{M}}$ on any input $\mathbf{x}$. For $\mathcal{O}_2(= \mathcal{O}_2^{(\mathbb{K})}) = \bigcup_{i \geq 0} W_i$ defined by $W_0 = \emptyset$ and

$$\begin{aligned}W_i = \{(\tilde{t}, \mathbf{x}, \mathbf{u}) \in U^i \mid \\ \mathbf{u} \in \mathcal{V} \;\&\; (\exists \mathcal{M} \in \mathsf{M}_{\mathbb{K}}^{\mathrm{N}}(\textstyle\bigcup_{j<i} W_j))(\mathbf{u} \text{ is the code of } \mathcal{M} \;\&\; \mathcal{M}(\mathbf{x}) \downarrow^t)\},\end{aligned}$$

there holds $\mathrm{UNI}_2^{(\mathbb{K})}(\mathcal{O}_2) \cap U^i \subseteq W_i$ for any $i > 0$. This implies the following.

Proposition 12. *For any* $\mathbb{K} \in \mathsf{struc}_{a,b}(U)$*, for which all oracle machines can be encoded by suitable computable tuples in* U^∞*, there is some oracle* $\mathcal{O}$ *such that* $\mathrm{P}_{\mathbb{K}}^{\mathcal{O}} = \mathrm{NP}_{\mathbb{K}}^{\mathcal{O}}$.

Remark. We can transfer the proof of Proposition 11 in order to get $\mathrm{PH}_{\mathbb{K}} \subseteq \mathrm{PAT}_{\mathbb{K}} \subseteq \mathrm{P}_{\mathbb{K}}^{\mathcal{O}_2}$ if the code of the machine which works as $\mathcal{M}_1$ is computable by some $\mathbb{K}$-machine in polynomial time.

4 The Inequality of Relativized Versions of P and NP

We shall present three kinds of oracles $\mathcal{Q}_1(= \mathcal{Q}_1^{(\mathbb{K})})$, $\mathcal{Q}_2(= \mathcal{Q}_2^{(\mathbb{K})})$, and $\mathcal{Q}_3(= \mathcal{Q}_3^{(\mathbb{K})})$ for several classes of structures $\mathbb{K}$, in order to get the inequality between the corresponding relativized versions of the classes $\mathrm{P}_{\mathbb{K}}$ and $\mathrm{NP}_{\mathbb{K}}$. The first two oracles are defined recursively by means of diagonalization techniques. These techniques were also used by Gill, Baker, Solovay, and R. Ladner (for details see [Baker et al. 1975]) and Emerson (see [Emerson 1994]).

4.1 The Classical Way to Define the First Kind of Oracles

If $\mathbb{K}$ is in the class $\mathsf{struc}_{a,b}^{\mathsf{enum}}(U)$ of structures of enumerable signature, then the wished oracle can be defined recursively on the number of programs as in [Baker et al. 1975]. We take the positive integers in order to

- enumerate all programs of oracle machines whose form (including the oracle queries) is independent of the used oracle,

– encode all polynomial functions which can be used to define time bounds for the computation processes,

– encode all couples of polynomial functions and programs.

Let $i \in \mathbb{N}^+$ be the code of a pair (p_i, P_i) which determines a class of deterministic oracle $\mathbb{K}$-machines $\{\mathcal{N}_i^{\mathcal{B}} \mid \mathcal{B} \subseteq U^\infty\}$ by the following.

(a) The machine $\mathcal{N}_i^{\mathcal{B}}$ performs the instructions of the program P_i.

(b) If $\mathcal{N}_i^{\mathcal{B}}$ queries an oracle, then $\mathcal{N}_i^{\mathcal{B}}$ uses the oracle $\mathcal{B}$.

(c) The number of the instructions of P_i carried out by $\mathcal{N}_i^{\mathcal{B}}$ is simultaneously counted by $\mathcal{N}_i^{\mathcal{B}}$ by means of an additional index register.

(d) For any input in U^n, the machine $\mathcal{N}_i^{\mathcal{B}}$ halts after at most $p_i(n)$ steps of the execution of P_i. (The bound $p_i(n)$ can be computed by using index registers.)

(e) If the output of P_i is reached in this time, then $\mathcal{N}_i^{\mathcal{B}}$ outputs the value determined by P_i. If the output instruction of P_i is not reached in this time, then $\mathcal{N}_i^{\mathcal{B}}$ rejects the input.

Then, for any oracle $\mathcal{B}$ and any problem $\mathcal{P} \in \mathrm{P}_{\mathbb{K}}^{\mathcal{B}}$ there is an $i \geq 1$ such that the machine $\mathcal{N}_i^{\mathcal{B}}$ decides $\mathcal{P}$.

The Construction of $\mathcal{Q}_1$. *Let $V_0 = \emptyset$ and $m_0 = 0$. We construct the set $\mathcal{Q}_1$ in stages.*

Stage $i \geq 1$: Let n_i be any integer such that $n_i > m_{i-1}$ and $p_i(n_i) + n_i < 2^{n_i}$. Moreover, let

$$\begin{aligned} W_i &= \textstyle\bigcup_{j<i} V_j, \\ V_i &= \{\mathbf{x} \in \{a,b\}^{n_i} \mid \mathcal{N}_i^{W_i} \text{ rejects } (a,\ldots,a) \in U^{n_i} \\ &\qquad \& \ \mathbf{x} \text{ is not queried by } \mathcal{N}_i^{W_i} \text{ on input } (a,\ldots,a) \in U^{n_i}\}, \\ m_i &= 2^{n_i}. \end{aligned}$$

Finally, let $\mathcal{Q}_1 = \bigcup_{i\geq 1} W_i$ and $L_1 = \{\mathbf{y} \mid (\exists i \in \mathbb{N}^+)(\mathbf{y} \in U^{n_i} \ \& \ V_i \neq \emptyset)\}$.

Lemma 13. $L_1 \in \mathrm{NP}_{\mathbb{K}}^{\mathcal{Q}_1} \setminus \mathrm{P}_{\mathbb{K}}^{\mathcal{Q}_1}$.

Proof. $p_i(n_i) < 2^{n_i}$ implies that $V_i \neq \emptyset$ if $\mathcal{N}_i^{W_i}$ rejects $(a,\ldots,a) \in U^{n_i}$. The computation of the machines $\mathcal{N}_i^{W_i}$ and $\mathcal{N}_i^{\mathcal{Q}_1}$ is the same on all inputs in U^{n_i} since the length of the tuples in the oracle queries is bounded by $p_i(n_i) + n_i$ and the queries of $\mathcal{N}_i^{W_i}$ and $\mathcal{N}_i^{W_{i+1}}$ with respect to V_i are the same. Let us assume that there is an i_0 such that $\mathcal{N}_{i_0}^{\mathcal{Q}_1}$ decides L_1. However, the machine $\mathcal{N}_{i_0}^{W_{i_0}}$, and hence $\mathcal{N}_{i_0}^{\mathcal{Q}_1}$, accepts $(a,\ldots,a) \in U^{n_{i_0}}$ if and only if $L_1 \cap U^{n_{i_0}} = \emptyset$ and it rejects $(a,\ldots,a) \in U^{n_{i_0}}$ if and only if $L_1 \cap U^{n_{i_0}} = U^{n_{i_0}}$. Thus, $L_1 \notin \mathrm{P}_{\mathbb{K}}^{\mathcal{Q}_1}$. □

Proposition 14. *For any structure* $\mathbb{K} \in \mathsf{struc}^{\mathsf{enum}}_{a,b}(U)$ *there is an oracle* $\mathcal{Q}$ *such that* $\mathrm{P}^{\mathcal{Q}}_{\mathbb{K}} \neq \mathrm{NP}^{\mathcal{Q}}_{\mathbb{K}}$.

Example 1. Structures with $\mathrm{P}^{\mathcal{O}_1}_{\mathbb{K}} = \mathrm{NP}^{\mathcal{O}_1}_{\mathbb{K}}$ and $\mathrm{P}^{\mathcal{Q}_1}_{\mathbb{K}} \neq \mathrm{NP}^{\mathcal{Q}_1}_{\mathbb{K}}$

Structures of finite signature with two constants:

$\mathbb{K} = (\{0,1\}; 0, 1; ; =)$,

$\mathbb{K} = (\mathbb{N}; 0, 1; s; =)$ where $s(n) = n + 1$,

$\mathbb{K} = (\mathbb{Z}; 0, 1; +, -; =)$, $\mathbb{K} = (\mathbb{Z}; 0, 1; +, -; \leq, =)$,

$\mathbb{K} = (\mathbb{Q}; 0, 1; +, -, \cdot; =)$, $\mathbb{K} = (\mathbb{Q}; 0, 1; +, -, \cdot; \leq, =)$,

$\mathbb{K} = (\mathbb{R}; 0, 1; +, -, \cdot; =)$, $\mathbb{K} = (\mathbb{R}; 0, 1; +, -, \cdot; \leq, =)$.

Example 2. Structures with $\mathrm{P}^{\mathcal{O}_2}_{\mathbb{K}} = \mathrm{NP}^{\mathcal{O}_2}_{\mathbb{K}}$ and $\mathrm{P}^{\mathcal{Q}_1}_{\mathbb{K}} \neq \mathrm{NP}^{\mathcal{Q}_1}_{\mathbb{K}}$

Structures of enumerable signature:

$\mathbb{K} = (\mathbb{Z}; \mathbb{Z}; +, -; =)$, $\mathbb{K} = (\mathbb{Z}; \mathbb{Z}; +, -; \leq, =)$,

$\mathbb{K} = (\mathbb{Q}; \mathbb{Q}; +, -, \cdot; =)$, $\mathbb{K} = (\mathbb{Q}; \mathbb{Q}; +, -, \cdot; \leq, =)$.

A structure of enumerable signature without an NP-complete problem:

$\mathbb{K} = (\mathbb{Z}; 0, 1; (\mathrm{sc}_c : c \in \mathbb{Z}); =)$ where $\mathrm{sc}_c(z) = cz$ for all $z \in \mathbb{Z}$.

4.2 The Second Kind of Oracles

Now, we want to consider mainly structures $\mathbb{K} \in \mathsf{struc}_{a,b}(U)$ whose signature and, consequently, the programs of oracle machines over $\mathbb{K}$ are not countable. We simplify and generalize the construction given by Emerson for Archimedean rings in [Emerson 1994] and for special groups in [Gaßner (2) 2008].

Let us assume that, for any oracle $\mathcal{B}$, all machines in $\mathsf{M}_{\mathbb{K}}(\mathcal{B})$ can be encoded by tuples in a set $\mathcal{U} \subseteq U^\infty$ independently of the used oracle such that each $\mathbf{u} \in \mathcal{U}$ represents a pair $(p_{\mathbf{u}}, P_{\mathbf{u}})$ which determines a class of deterministic oracle $\mathbb{K}$-machines $\{\mathcal{N}^{\mathcal{B}}_{\mathbf{u}} \mid \mathcal{B} \subseteq U^\infty\}$ satisfying the properties in analogy with (a), (b), (c), (d), and (e). Again this implies that, for any problem $\mathcal{P} \in \mathrm{P}^{\mathcal{B}}_{\mathbb{K}}$, there is some $\mathbf{u} \in \mathcal{U}$ such that $\mathcal{N}^{\mathcal{B}}_{\mathbf{u}}$ decides $\mathcal{P}$ in polynomial time. In the following construction we use the properties (i) and (ii).

(i) We permit any structure $\mathbb{K}$ with an infinite universe U which allows to define the necessary codes by tuples in U^∞.

(ii) Since U is infinite, we shall assume that there is an element α_0 and an injective mapping $\sigma : U \to U$ satisfying $\sigma(\alpha_i) = \alpha_{i+1}$ and $\alpha_{i+1} \neq \alpha_0$ for all $i \in \mathbb{N}$. The mapping need not belong to the structure and it is not necessary that this mapping can be defined or computed over $\mathbb{K}$. We denote the infinite sequence of images by $\bar{1}, \bar{2}, \ldots$ where $\bar{n}(= \bar{n}_{\mathbb{K}}) =_{\mathrm{df}} \sigma(\alpha_{n-1})$ for any $n \in \mathbb{N}^+$.

The Construction of $\mathcal{Q}_2$. *Let us assume that U contains an infinite sequence $\bar{1}, \bar{2}, \ldots$ given by an injective mapping σ described above. Let $V_0 = \emptyset$. We construct the set $\mathcal{Q}_2$ in stages.*
Stage $i \geq 1$: Let

$$K_i = \{\mathbf{u} \in \mathcal{U} \mid (\forall j > i)(\forall \mathcal{B} \subseteq U^\infty)$$
$$(\mathcal{N}_{\mathbf{u}}^{\mathcal{B}} \text{ does not compute or use the value } \bar{j} \text{ on input } \mathbf{u})\},$$
$$W_i = \bigcup_{k<i} V_k,$$
$$V_i = \{(\overline{i+1}, \mathbf{u}) \mid \mathbf{u} \in K_i \ \&\ \mathcal{N}_{\mathbf{u}}^{W_i} \text{ rejects } \mathbf{u}\}.$$

Finally, let $\mathcal{Q}_2 = \bigcup_{i \geq 1} W_i$ and $L_2 = \{\mathbf{y} \mid (\exists n \in \mathbb{N}^+)((\bar{n}, \mathbf{y}) \in \mathcal{Q}_2)\}$.

Lemma 15. $L_2 \in \mathrm{NP}_{\mathbb{K}}^{\mathcal{Q}_2} \setminus \mathrm{P}_{\mathbb{K}}^{\mathcal{Q}_2}$.

Proof. Let us assume that some machine

$$\mathcal{N}_{\mathbf{u}}^{\mathcal{Q}_2} \text{ decides } L_2. \tag{2}$$

We consider the cases where $\mathbf{u}$ is in L_2 and where $\mathbf{u}$ is not in L_2. If $\mathbf{u} \in L_2$, then there is some i such that $(\overline{i+1}, \mathbf{u}) \in V_i$. Thus, by the definition of K_i the machines $\mathcal{N}_{\mathbf{u}}^{\mathcal{Q}_2}$ and $\mathcal{N}_{\mathbf{u}}^{W_i}$ do not use any value $\bar{j}$ in any query on input $\mathbf{u}$ if $j > i$. Hence, $\mathcal{N}_{\mathbf{u}}^{\mathcal{Q}_2}$ works as $\mathcal{N}_{\mathbf{u}}^{W_i}$ and, consequently, it rejects $\mathbf{u}$. Thus, $\mathbf{u} \notin L_2$ holds because of (2). If $\mathbf{u} \notin L_2$, then because of (2), $\mathcal{N}_{\mathbf{u}}^{\mathcal{Q}_2}$ rejects $\mathbf{u}$. Since by definition of the sets $K_1, K_2, \ldots$ there is some i_0 such that $\mathbf{u} \in K_i$ for all $i \geq i_0$, that means that $(\overline{i_0 + 1}, \mathbf{u}) \in \mathcal{Q}_2$ and hence $\mathbf{u} \in L_2$. $\square$

Proposition 16. *For any structure $\mathbb{K} \in \mathsf{struc}_{a,b}(U)$ with an infinite universe U which allows to encode the $\mathbb{K}$-machines by means of tuples in U^∞, there is an oracle $\mathcal{Q}$ such that $\mathrm{P}_{\mathbb{K}}^{\mathcal{Q}} \neq \mathrm{NP}_{\mathbb{K}}^{\mathcal{Q}}$.*

Example 3. Structures with $\mathrm{P}_{\mathbb{K}}^{\mathcal{O}_2} = \mathrm{NP}_{\mathbb{K}}^{\mathcal{O}_2}$ and $\mathrm{P}_{\mathbb{K}}^{\mathcal{Q}_2} \neq \mathrm{NP}_{\mathbb{K}}^{\mathcal{Q}_2}$

Structures of non-enumerable signature:

$\mathbb{K} = (\mathbb{R}; \mathbb{R}; +, -, \cdot; =)$, $\mathbb{K} = (\mathbb{R}; \mathbb{R}; +, -, \cdot; \leq, =)$,

$\mathbb{K} = (\mathbb{R}; \mathbb{R}; +, -; =)$, $\mathbb{K} = (\mathbb{R}; \mathbb{R}; +, -; \leq, =)$,

$\mathbb{K} = (\mathbb{R}; 0, 1; (\mathrm{sc}_c : c \in \mathbb{R}); =)$ where $\mathrm{sc}_c(r) = cr$ for all $r \in \mathbb{R}$,

$\mathbb{K} = (\mathbb{R}; 0, 1; (\mathrm{sc}_c : c \in \mathbb{R}), +, -; =)$, $\mathbb{K} = (\mathbb{R}; 0, 1; (\mathrm{sc}_c : c \in \mathbb{R}), +, -; \leq, =)$.

Remark. 1. Simpler constructions are possible if there is an element which is not computable from the codes in $\mathcal{U}$. For instance, the deterministic oracle machines over $\mathbb{Q}[\sqrt{2}] = (\mathbb{Q}\sqrt{2} + \mathbb{Q}; 0, 1; +, -, \cdot; =)$ can be encoded by integers $i \in \mathbb{N}$. Then, the inequality $\mathrm{NP}_{\mathbb{Q}[\sqrt{2}]}^{\mathcal{Q}} \not\subseteq \mathrm{DEC}_{\mathbb{Q}[\sqrt{2}]}^{\mathcal{Q}}$, and thus, $\mathrm{P}_{\mathbb{Q}[\sqrt{2}]}^{\mathcal{Q}} \neq \mathrm{NP}_{\mathbb{Q}[\sqrt{2}]}^{\mathcal{Q}}$ holds for $\mathcal{Q} = \{(\sqrt{2}, i) \mid \mathcal{N}_i^{\emptyset} \text{ rejects } i\}$ since, for any $i \in \mathbb{N}$, the behaviour of $\mathcal{N}_i^{\mathcal{Q}}$ and $\mathcal{N}_i^{\emptyset}$ is the same on all inputs in $\mathbb{N}$.

2. In order to get $\mathrm{P}_{\mathbb{R}}^{\mathcal{Q}} \neq \mathrm{NP}_{\mathbb{R}}^{\mathcal{Q}}$ for $\mathbb{R} = (\mathbb{R}; \mathbb{R}; +, -, \cdot; \leq, =)$ (where any real number can be a machine constant), Emerson constructed a special oracle $\mathcal{Q}$ in [Emerson 1994]. For any program $P_{\mathbf{u}}$ and any polynomial function $p_{\mathbf{u}}$, he considered the greatest absolute value of all numbers used in a query by one of the oracle machines in $\{\mathcal{N}_{\mathbf{u}}^{\mathcal{B}} \mid \mathcal{B} \subseteq U^{\infty}\}$ if these machines get their own code $\mathbf{u}$ as input. In order to define some oracle recursively, for any natural number $i > 0$, he used a set K_i formed by taking all codes of $(p_{\mathbf{u}}, P_{\mathbf{u}})$ for which this greatest value is in the interval $]i-1, i]$. Emerson restricted his proofs to an Archimedean ring and he mentioned the possibility to transfer his results to other ordered rings if the Axiom of Choice (AC) and, consequently, the Well-Ordering Axiom (WO) are assumed. We extended his investigation by using only the properties (i) and (ii), and in this way we also answer the three questions posed by Emerson in the last section of [Emerson 1994]. Our assumption is not equivalent to AC. If σ is computable, then neither any restrictions for the operations and the relations of the structure nor for the domain $U \setminus \{\alpha_0, \alpha_1, \ldots\}$ are necessary. The cardinality of the infinite universe U is not important for the construction.

For some other structures, the weaker Axiom of Depend Choice (DC) which was introduced by P. Bernays in his paper [Bernays 1942] and which is used instead of the general AC in the Analytical Topology can be sufficient. If we assume DC, then we can use that, for any binary relation R, there is a sequence $(x_i)_{i \geq 0}$ such that $(x_i, x_{i+1}) \in R$ for all $i \geq 0$. Let us consider an infinite abelian group which does not contain an element of infinite order. Then we can consider the inclusion relation on the set of all non-trivial subgroups. By DC there exists, for instance, an infinite sequence of subgroups $(G_i)_{i \geq 0}$ whose members include their predecessors properly. Moreover, this implies the existence of an injective mapping σ by DC where $\sigma(\alpha_i) \in G_{i+1} \setminus G_i$. (See also [Gaßner (2) 2008].)

4.3 The Third Kind of Oracles

The following oracle is not recursively defined and we can use the undecidability of the corresponding Halting Problem in the proof. We consider only the class $\mathsf{struc}_{\bar{\mathbb{N}}}^{\mathsf{fin}}(U)$ containing all structures $\mathbb{K} \in \mathsf{struc}_{a,b}^{\mathsf{fin}}(U)$ for which U includes an infinite set $\bar{\mathbb{N}} = \{\bar{0}, \bar{1}, \bar{2}, \ldots\}$ with the following properties.

- $\bar{\mathbb{N}}$ is defined by some injective mapping σ of U into U where $\overline{i+1} = \sigma(\bar{i}) \neq \bar{0}$.
- $\bar{\mathbb{N}}$ is decidable by a deterministic $\mathbb{K}$-machine.
- $\bar{\mathbb{N}}$ is enumerable by a deterministic $\mathbb{K}$-machine which can compute $\bar{0}$ independently of the input and which can compute $\overline{i+1}$ from $\bar{i}$.
- a and b are elements of $\bar{\mathbb{N}}$.

The constructions given in Sections 4.1 and 4.2 are possible for any classes of time bounds limiting the work of the deterministic oracle machines if these bounds are computable by means of index registers. We can build, for instance, some oracle $\mathcal{Q}$ such that $\mathrm{EXP}_{\mathbb{K}}^{\mathcal{Q}} \neq \mathrm{NP}_{\mathbb{K}}^{\mathcal{Q}}$ holds if we use the exponential functions instead of the polynomial functions as time bounds. The next oracle implies the corresponding inequalities for each class of time bounds.

The Definition of $\mathcal{Q}_3$. *For* $\mathbb{K} \in \mathsf{struc}_{\mathbb{N}}^{\mathsf{fin}}(U)$*, let*

$$\mathcal{Q}_3 = \{(\bar{t}, \mathbf{x}, Code(\mathcal{M})) \in U^{\infty} \mid \mathcal{M} \in \mathsf{M}_{\mathbb{K}} \ \&\ t \in \mathbb{N}^+ \ \&\ \mathcal{M}(\mathbf{x}) \downarrow^t\}.$$

Lemma 17. *For any* $\mathbb{K} \in \mathsf{struc}_{\mathbb{N}}^{\mathsf{fin}}(U)$*,* $\mathrm{H}_{\mathbb{K}} \in \mathrm{NP}_{\mathbb{K}}^{\mathcal{Q}_3}$ *and* $\mathrm{P}_{\mathbb{K}}^{\mathcal{Q}_3} \subseteq \mathrm{DEC}_{\mathbb{K}}$*.*

Proof. Let us consider the following program: Input any $(\mathbf{x}, Code(\mathcal{M}))$. Guess $y \in U$. Output a if $(y, \mathbf{x}, Code(\mathcal{M})) \in \mathcal{Q}_3$. The acceptance is possible only if there is an $n \in \mathbb{N}^+$ such that $y = \bar{n}$. Therefore, $\mathrm{H}_{\mathbb{K}} \in \mathrm{NP}_{\mathbb{K}}^{\mathcal{Q}_3}$. Now let us assume that $\{\bar{0}, \bar{1}, \ldots\}$ is decidable and enumerable by some $\mathbb{K}$-machine. For any query described by $\mathbf{z} \in \mathcal{Q}_3?$, there is a deterministic $\mathbb{K}$-machine $\mathcal{M}_0$ which can decide whether z has the form $(\bar{t}, \mathbf{x}, Code(\mathcal{M}))$ for some $t \in \mathbb{N}^+$ and some $\mathcal{M} \in \mathsf{M}_{\mathbb{K}}$ and which can simulate each step of $\mathcal{M}$. Because of the computability of $\bar{t}$ by computing $\bar{0}, \bar{1}, \ldots, \bar{t}$ it is possible to count the number of steps. That means that $\mathrm{P}_{\mathbb{K}}^{\mathcal{Q}_3} \subseteq \mathrm{DEC}_{\mathbb{K}}$. □

By Corollary 8 we can conclude the following.

Proposition 18. *For any* $\mathbb{K} \in \mathsf{struc}_{\mathbb{N}}^{\mathsf{fin}}(U)$ *there is some oracle* $\mathcal{Q}$ *such that* $\mathrm{P}_{\mathbb{K}}^{\mathcal{Q}} \neq \mathrm{NP}_{\mathbb{K}}^{\mathcal{Q}}$*.*

Example 4. Structures with $\mathrm{P}_{\mathbb{K}}^{\mathcal{O}_1} = \mathrm{NP}_{\mathbb{K}}^{\mathcal{O}_1}$ and $\mathrm{P}_{\mathbb{K}}^{\mathcal{Q}_3} \neq \mathrm{NP}_{\mathbb{K}}^{\mathcal{Q}_3}$

Structures of finite signature for which $\mathbb{N}$ is decidable:

$\mathbb{K} = (\mathbb{N}; 0, 1; s; =)$, $\mathbb{K} = (\mathbb{N}; 0, 1; +, -; =)$, $\mathbb{K} = (\mathbb{Z}; 0, 1; +, -; =)$,

$\mathbb{K} = (\mathbb{Q}; 0, 1; +, -, \cdot; \leq, =)$, $\mathbb{K} = (\mathbb{R}; 0, 1; +, -, \cdot; \leq, =)$.

Structures with a finite number of operations and relations:

If we extend the codes so that the constants are represented by themselves, then we get the same results for the following structures.

$\mathbb{K} = (\mathbb{R}; \mathbb{R}; +, -, \cdot; \leq, =)$, $\mathbb{K} = (\mathbb{R}; \mathbb{R}; +, -; \leq, =)$.

4.4 Further Oracles for the Ordered Ring of Reals

In Section 4.2 we could simplify and generalize the construction given by Emerson in order to given suitable oracles for structures of non-enumerable signature since the main arguments are of logical nature. However, for the ring

$\mathbb{R} =_{\rm df} (\mathbb{R};\mathbb{R};+,-,\cdot;\leq,=)$ considered by Emerson we can give other simple oracles $\mathcal{Q}$ satisfying $\mathrm{P}_{\mathbb{R}}^{\mathcal{Q}} \neq \mathrm{NP}_{\mathbb{R}}^{\mathcal{Q}}$.

Since, for any input $x \in \mathbb{R}$, we can find a $k \in \mathbb{N}$ with $-2^k < x < 2^k$, it is possible to decide $\mathbb{Z}$ by binary searching. In order to accept or reject an input, an $\mathbb{R}$-machine needs $O(\log(\lceil |x| \rceil))$ many steps (where $\lceil x \rceil$ is the smallest integer greater than x). If we allow to decide $\mathbb{Z}$ within one step, then $\mathbb{Q}$ can be non-deterministically recognized by a machine in $\mathsf{M}_{\mathbb{R}}^{\mathrm{N}}(\mathbb{Z})$ which queries the oracle whether the guesses y_1 and y_2 are integers and which checks $y_1 \neq 0$ and $y_1 x = y_2$ for any input x. On the other hand, we know from the original BSS paper that $\mathbb{Q}$ is not decidable over $\mathbb{R}$. Thus, we also have $\mathbb{Q} \notin \mathrm{P}_{\mathbb{R}}^{\mathbb{Z}}$ since the oracle queries can be simulated by a machine in $\mathsf{M}_{\mathbb{R}}$. Hence, we have $\mathrm{P}_{\mathbb{R}}^{\mathbb{Z}} \neq \mathrm{NP}_{\mathbb{R}}^{\mathbb{Z}}$.

In order to show $\mathrm{P}_{\mathbb{R}}^{\mathbb{Q}} \neq \mathrm{NP}_{\mathbb{R}}^{\mathbb{Q}}$ we consider the set

$$\mathbb{Q}_{\rm sq} = \{q^2 \mid q \in \mathbb{Q}\}.$$

$\mathbb{Q}_{\rm sq} \in \mathrm{DEC}_{\mathbb{R}}^{\mathbb{Q}}$ holds (cf. [Meer and Ziegler 2006]), but we can show $\mathbb{Q}_{\rm sq} \notin \mathrm{P}_{\mathbb{R}}^{\mathbb{Q}}$ by the following lemma. In the proof of $\mathbb{Q}_{\rm sq} \notin \mathrm{P}_{\mathbb{R}}^{\mathbb{Q}}$ it is important that we have to consider only a finite number of computation paths.

Lemma 19. *For any polynomial function $p \in \mathbb{R}[x] \setminus \mathbb{Q}[x]$, there is only a finite number of rational numbers $q \in \mathbb{Q}$ satisfying $p(q) \in \mathbb{Q}$.*

Proof. For any $p \in \mathbb{R}[x]$ given by $p(x) = a_0 + a_1 x + \cdots + a_n x^n$ for some $n \geq 0$ and some $a_0, \ldots, a_n \in \mathbb{R}$, there are polynomial functions $q_0, \ldots, q_v \in \mathbb{Q}[x]$ and linearly independent coefficients $a_{i_1}, a_{i_2}, \ldots, a_{i_v}$ $(v \leq n+1)$ which satisfy

$$p(x) = q_0(x) + a_{i_1} q_1(x) + \cdots + a_{i_v} q_v(x)$$

where $\gamma + \alpha_1 a_{i_1} + \cdots + \alpha_v a_{i_v} = 0$ implies $\gamma = \alpha_1 = \cdots = \alpha_v = 0$ if $\gamma, \alpha_1, \ldots, \alpha_v \in \mathbb{Q}$. Thus, for any $x \in \mathbb{Q}$, the condition $p(x) \in \mathbb{Q}$ means that $q_1(x) = \cdots = q_v(x) = 0$ and $p(x) = q_0(x)$. These equations are satisfied either by all rational numbers, and then $p = q_0$ and all coefficients of p are rational numbers, which are determined by Lagrange's interpolation formula, or only by a finite number of rational numbers. $\square$

Proposition 20. *For the ordered ring $\mathbb{R}$ of real numbers, $\mathbb{Q}_{\rm sq} \in \mathrm{NP}_{\mathbb{R}}^{\mathbb{Q}} \setminus \mathrm{P}_{\mathbb{R}}^{\mathbb{Q}}$ holds.*

Proof. $\mathbb{Q}_{\rm sq}$ can be non-deterministically recognized by a machine in $\mathsf{M}_{\mathbb{R}}^{\mathrm{N}}(\mathbb{Q})$ which queries the oracle whether a guess y is a rational number and which checks $x = y^2$ for any input x.

Now, assume that there is a machine $\mathcal{N}$ in $\mathsf{M}_{\mathbb{R}}(\mathbb{Q})$ which decides $\mathbb{Q}_{\rm sq}$ in polynomial time. Let $M = \{p_1, \ldots, p_m\}$ contain all polynomial functions of arity 1 and degree $d \geq 1$, such that each of the computation paths of $\mathcal{N}$ on inputs of size 1, P, can be described by a system S_P consisting of conditions of the form $p_k(x) \leq 0$, $p_k(x) < 0$, $p_k(x) \notin \mathbb{Q}$, and $p_k(x) \in \mathbb{Q}$ $(k \leq m)$.

Now, let r be a prime number satisfying (3) and (4).

$$r > \max\{s \mid (\exists k \leq m)(p_k(s) = 0)\}, \tag{3}$$

$$r > \max\{s \in \mathbb{Q} \mid (\exists k \leq m)(p_k \notin \mathbb{Q}[x] \;\&\; p_k(s) \in Q)\}. \tag{4}$$

The number r exists since both sets are finite. The second set is finite by Lemma 19. Because of (3), we have $p_k(r) \neq 0$ and $p_k(r^2) \neq 0$ for all $k \leq m$ since $r^2 > r$. Between r and r^2, there is not a zero of p_k. Thus $p_k(r) < 0$ holds if and only if $p_k(r^2) < 0$. Moreover, by (4) we have $p_k(r) \notin \mathbb{Q}$ and $p_k(r^2) \notin \mathbb{Q}$ for $k \leq m$ in case that $p_k \notin \mathbb{Q}[x]$.

Then, $r \in \mathbb{Q} \setminus \mathbb{Q}_{\mathsf{sq}}$ and r^2 traverses the same computation path of $\mathcal{N}$. That contradicts the assertion. □

5 Relations Instead of Oracles?

Since, for many structures $\mathbb{K}$, we do not know the relationship between $\mathrm{P}_{\mathbb{K}}$ and $\mathrm{NP}_{\mathbb{K}}$, we want to discuss the following question. Is it possible to replace the oracle $\mathcal{O}_1^{(\mathbb{K})}$ for some $\mathbb{K} \in \mathsf{struc}_{a,b}^{\mathsf{fin}}(U)$ by one additional relation in order to get a structure $\mathbb{M}$ of finite signature with $\mathrm{P}_{\mathbb{M}} = \mathrm{NP}_{\mathbb{M}}$?

If we want to derive a new relation R (which can be satisfied only by tuples of a fixed length n_R) from the oracle $\mathcal{O}_1$ such that any oracle query $(Z_1, \ldots, Z_{I_1}) \in \mathcal{O}$ can be replaced by a condition of the form $R(Z_1, \ldots, Z_{n_R})$, then we have to compress the tuples in $\mathcal{O}_1$ to tuples of length n_R. Since, for many structures, it is not possible to compute a bijection of the set of the finite sequences of elements into a set of tuples of a fixed length, here we want to consider a class of structures over strings which allow to encode finite sequences of elements by single elements.

Definition 21. *For any universe U, let $\mathcal{A} = U^*$ such that the elements of U are the characters of the strings in $\mathcal{A}$, and let $\mathsf{struc}_*^{\mathsf{fin}}(\mathcal{A})$ be the class of structures of the form $(\mathcal{A}; \mathcal{A}_0; f_1, \ldots, f_{k_1}, \mathrm{add}, \mathrm{sub_l}, \mathrm{sub_r}; R_1, \ldots, R_{k_2}, =)$ where $\mathcal{A}_0 \subseteq \mathcal{A}$ is a finite set of constants and $a, b, \varepsilon \in \mathcal{A}_0$. add is a binary operation for adding a character to a string. $\mathrm{sub_r}$ and $\mathrm{sub_l}$ are unary operations for computing the last character and the remainder of a string, respectively. That means that these functions are defined for the strings $s \in \mathcal{A}$, $r \in \mathcal{A} \setminus U$, and $c \in U$ by $\mathrm{add}(s, c) = sc$, $\mathrm{sub_l}(sc) = s$, $\mathrm{sub_r}(sc) = c$, $\mathrm{add}(s, r) = \varepsilon$, $\mathrm{sub_l}(\varepsilon) = \varepsilon$, and $\mathrm{sub_r}(\varepsilon) = \varepsilon$. Each f_i is an operation on $\mathcal{A}$. Each R_i is a relation on $\mathcal{A}$.*

In encoding the elements of oracles we can use that the tuples of strings can be encoded by strings in the following way.

Definition 22. *For every string $s \in \mathcal{A}$, let the value $\langle s \rangle$ be recursively defined by $\langle \varepsilon \rangle = a$ and $\langle rc \rangle = \langle r \rangle ca$ for all strings $r \in \mathcal{A}$ and all character $c \in U$. For every integer $n > 1$ and every tuple $\mathbf{s} = (s_1, \ldots, s_n) \in \mathcal{A}^n$, let $\langle s_1, \ldots, s_n \rangle$ be the string $\langle s_1 \rangle b^2 \cdots \langle s_{n-1} \rangle b^2 \langle s_n \rangle$.*

Note that each deterministic machine over $(U^*; a, b, \varepsilon; \mathrm{add}, \mathrm{sub_l}, \mathrm{sub_r}; =)$ can be simulated by some machine over $(U; a, b; ; =)$ in polynomial time if the inputs are given in a suitable form. If a machine over $(U^*; a, b, \varepsilon; \mathrm{add}, \mathrm{sub_l}, \mathrm{sub_r}; =)$ processes guessed strings, then, in general, the simulation is not possible in polynomial time.

Since we can take $\bar{t} = b^t$ for any $t \in \mathbb{N}$, we have $\mathsf{struc}_*^{\mathsf{fin}}(\mathcal{A}) \subset \mathsf{struc}_{\mathbb{N}}^{\mathsf{fin}}(\mathcal{A})$ because of the following lemma.

Lemma 23. $\{b^i \in \mathcal{A} \mid i \in \mathbb{N}\}$ *is decidable and enumerable over* $\mathbb{K} \in \mathsf{struc}_*^{\mathsf{fin}}(\mathcal{A})$.

Although, for any $\mathbb{K} \in \mathsf{struc}_*^{\mathsf{fin}}(\mathcal{A})$, the elements of the oracles $\mathcal{O} = \mathcal{O}_1^{(\mathbb{K})}$ and $\mathcal{Q} = \mathcal{Q}_3^{(\mathbb{K})}$ have a similar form, we have different relationships between the relativized versions of $\mathrm{P}_{\mathbb{K}}$ and $\mathrm{NP}_{\mathbb{K}}$. That implies, on the one hand, the conjecture that it could be easy to define oracles $\bar{\mathcal{O}}, \bar{\mathcal{Q}} \subseteq \mathcal{A}$ or new unary relations R by compressing the sequences of strings in $\mathcal{O}, \mathcal{Q} \subseteq \mathcal{A}^\infty$ to single strings in order to get $\mathrm{P}_{\mathbb{K}}^{\bar{\mathcal{O}}} = \mathrm{NP}_{\mathbb{K}}^{\bar{\mathcal{O}}}$ and $\mathrm{P}_{\mathbb{K}}^{\bar{\mathcal{Q}}} \neq \mathrm{NP}_{\mathbb{K}}^{\bar{\mathcal{Q}}}$ and $\mathrm{P}_{\mathbb{K}_R} = \mathrm{NP}_{\mathbb{K}_R}$ or $\mathrm{P}_{\mathbb{K}_R} \neq \mathrm{NP}_{\mathbb{K}_R}$ for new structures $\mathbb{K}_R$. On the other hand it implies the conjecture that it is not possible to define oracles $\bar{\mathcal{O}} \subseteq \mathcal{A}$ with $\mathrm{P}_{\mathbb{K}}^{\bar{\mathcal{O}}} = \mathrm{NP}_{\mathbb{K}}^{\bar{\mathcal{O}}}$ since the different relationships between the complexity classes, relativized by using the oracles $\mathcal{O}$ and $\mathcal{Q}$, respectively, mainly are the result of the different representation of the number of steps: In case of $\mathcal{O}$, the number of possible steps, t, is determined by the length of the tuple $\tilde{t}$. In case of $\mathcal{Q}$, the number of steps is given by only one element of the structure, $\bar{t}$. To use only single strings as codes of the elements of $\mathcal{O}$ in defining a new oracle $\bar{\mathcal{O}}$ could be easier said than done. The following results bear that out.

Theorem 24. *For any* $\mathbb{K} \in \mathsf{struc}_*^{\mathsf{fin}}(\mathcal{A})$, *the oracle*

$$\bar{\mathcal{Q}} = \{b^t \langle \mathbf{x} \rangle Code^*(\mathcal{M}) \in \mathcal{A} \mid \mathcal{M} \in \mathsf{M}_{\mathbb{K}} \ \&\ t \in \mathbb{N}^+ \ \&\ \mathcal{M}(\mathbf{x}) \downarrow^t\}$$

implies $\mathrm{P}_{\mathbb{K}}^{\bar{\mathcal{Q}}} \neq \mathrm{NP}_{\mathbb{K}}^{\bar{\mathcal{Q}}}$.

Proof. First we want to show that $\mathrm{P}_{\mathbb{K}}^{\bar{\mathcal{Q}}} \subset \mathrm{DEC}_{\mathbb{K}}$. Let $\mathcal{M}_0$ be any machine in $\mathsf{M}_{\mathbb{K}}(\bar{\mathcal{Q}})$. A query of the form $s \in \bar{\mathcal{Q}}$? carried out by $\mathcal{M}_0$ can be simulated by a deterministic machine which decides whether s has the form $b^t \langle \mathbf{x} \rangle Code^*(\mathcal{M})$ for some $t \geq 1$ and for some $\mathcal{M}$ and which simulates the first t steps of $\mathcal{M}$ on input $\mathbf{x}$ if s has a suitable form. By Lemma 23 it is possible to count the steps. Thus, there is a machine in $\mathsf{M}_{\mathbb{K}}$ which can simulate any instruction of $\mathcal{M}_0$ on every input.

$\mathrm{H}_{\mathbb{K}}^{\mathrm{spec}} \in \mathrm{NP}_{\mathbb{K}}^{\bar{\mathcal{Q}}}$ holds since there is some machine in $\mathsf{M}_{\mathbb{K}}^{\mathrm{N}}(\bar{\mathcal{Q}})$ such that, for any input $Code(\mathcal{M})$, this machine is able to guess a string $b^t \in \mathcal{A}$ in one step, to compute $s = b^t\langle Code(\mathcal{M})\rangle Code^*(\mathcal{M})$ in polynomial time, and to check $s \in \bar{\mathcal{Q}}$. Therefore, the assertion follows from Proposition 7. □

Whereas it is easy to transfer the construction of oracles in order to again obtain inequalities between the relativized polynomial time complexity classes for structures over strings, the method does not work if we want to again get equations for the relativized classes as it is shown by the following theorem.

Theorem 25. *For any $\mathbb{K} \in \mathsf{struc}_*^{\mathsf{fin}}(\mathcal{A})$, there is not any oracle satisfying*

$$b^t\langle \mathbf{x}\rangle Code^*(\mathcal{M}) \in \bar{\mathcal{O}} \Leftrightarrow \mathcal{M} \in \mathsf{M}_{\mathbb{K}}^{\mathrm{N}}(\bar{\mathcal{O}}) \;\&\; t \in \mathbb{N}^+ \;\&\; \mathcal{M}(\mathbf{x}) \downarrow^t . \tag{5}$$

Proof. Let us assume that such an oracle $\bar{\mathcal{O}}$ satisfying (5) is given. Then, there holds $\mathrm{NP}_{\mathbb{K}}^{\bar{\mathcal{O}}} \subseteq \mathrm{DEC}_{\mathbb{K}}^{\bar{\mathcal{O}}}$ by (5). Since we suppose that the set of the codes of the deterministic oracle machines in $\mathsf{M}_{\mathbb{K}}(\bar{\mathcal{O}})$ is decidable by some $\mathbb{K}$-machine and that we can extend the code of any machine such that any output b is avoided by an infinite loop, the Halting Problem for deterministic oracle machines can be non-deterministically recognized by guessing strings of the form b^t. Consequently,

$$\mathrm{H}_{\mathbb{K}}^{\mathrm{spec}}(\bar{\mathcal{O}}) = \{Code(\mathcal{M}) \in \mathcal{A}^\infty \mid \mathcal{M} \in \mathsf{M}_{\mathbb{K}}(\bar{\mathcal{O}}) \;\&\; \mathcal{M} \text{ halts on } Code(\mathcal{M})\}$$

is also in $\mathrm{NP}_{\mathbb{K}}^{\bar{\mathcal{O}}}$, and consequently in $\mathrm{DEC}_{\mathbb{K}}^{\bar{\mathcal{O}}}$, too. That means, that there exists some deterministic oracle machine which decides the special halting problem $\mathrm{H}_{\mathbb{K}}^{\mathrm{spec}}(\bar{\mathcal{O}})$. Since in analogy to Proposition 7 $\mathrm{H}_{\mathbb{K}}^{\mathrm{spec}}(\bar{\mathcal{O}})$ cannot be in $\mathrm{DEC}_{\mathbb{K}}^{\bar{\mathcal{O}}}$, we get a contradiction. □

Each deterministic machine over $\mathbb{K}_{\{a,b\}^*} = (\{a,b\}^*; a, b, \varepsilon; \mathrm{add}, \mathrm{sub}_{\mathrm{l}}, \mathrm{sub}_{\mathrm{r}}; =)$ can be simulated by some Turing machine. Thus, the following statement follows from the undecidability of the Halting Problem for the set TM of Turing machines.

Proposition 26. *The set*

$$\bar{\mathcal{Q}}_{\mathrm{TM}} = \{b^t\langle \mathbf{x}\rangle Code^*(\mathcal{M}) \in \{a,b\}^* \mid \mathcal{M} \in \mathsf{TM} \;\&\; t \in \mathbb{N}^+ \;\&\; \mathcal{M}(\mathbf{x}) \downarrow^t\}$$

implies $\mathrm{NP}_{\mathbb{K}_{\{a,b\}^*}}^{\bar{\mathcal{Q}}_{\mathrm{TM}}} \not\subseteq \mathrm{DEC}_{\mathbb{K}_{\{a,b\}^*}}^{\bar{\mathcal{Q}}_{\mathrm{TM}}}$ *and, hence,* $\mathrm{P}_{\mathbb{K}_{\{a,b\}^*}}^{\bar{\mathcal{Q}}_{\mathrm{TM}}} \neq \mathrm{NP}_{\mathbb{K}_{\{a,b\}^*}}^{\bar{\mathcal{Q}}_{\mathrm{TM}}}$.

Remark. The last result remains true if we consider machines over the structure $(\{a,b\}^*; a, b, \varepsilon; \mathrm{add}, \mathrm{sub}_{\mathrm{l}}, \mathrm{sub}_{\mathrm{r}}; = a, = b)$ where any test has the form $Z_j = a$ or $Z_j = b$.

A possibility to define new relations R (or oracles containing only single elements of the universe) derived from $\mathcal{O}_1^{(\mathbb{K})}$ such that, for the new structures $\mathbb{K}_R$, $\mathrm{P}_{\mathbb{K}_R} = \mathrm{NP}_{\mathbb{K}_R}$ holds is presented in [Gaßner (1) 2004, Gaßner (2) 2004, Gaßner 2007]. The crucial idea used to overcome the barriers resulting from Theorem 25 is to define new relations R satisfied by *padded* codes of the elements of an $\mathrm{NP}_{\mathbb{K}_R}$-complete problem. (For more details see [Gaßner (1) 2006, Gaßner (2) 2006], too.) The subject of [Gaßner (1) 2004] is the construction of a new structure over binary trees for which the equality of trees cannot be decided in one step.

In this way we can once more substantiate the thesis that additional oracles are not very helpful for solving the $\mathrm{P}_{\mathbb{K}} \stackrel{?}{=} \mathrm{NP}_{\mathbb{K}}$ problem for any structure $\mathbb{K}$. On the one hand, we know structures $\mathbb{K}$ with $\mathrm{P}_{\mathbb{K}} \neq \mathrm{NP}_{\mathbb{K}}$ (cf., e.g., [Gaßner 2001, Gaßner 1997]) and we can define an oracle $\mathcal{O}$ which implies $\mathrm{P}^{\mathcal{O}}_{\mathbb{K}} = \mathrm{NP}^{\mathcal{O}}_{\mathbb{K}}$. On the other hand, we know structures $\mathbb{M}$ with $\mathrm{P}_{\mathbb{M}} = \mathrm{NP}_{\mathbb{M}}$ and we can define an oracle $\mathcal{Q}$ implying $\mathrm{P}^{\mathcal{Q}}_{\mathbb{M}} \neq \mathrm{NP}^{\mathcal{Q}}_{\mathbb{M}}$.

Acknowledgments

I would like to thank the anonymous referees for careful reading of the paper and helpful comments.

References

[Baker et al. 1975] Baker, T., J. Gill, and R. Solovay: “Relativizations of the $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ question”; SIAM J. Comput. 4 (1975), 431–442.

[Balcázar et al. 1988/90] J. Balcázar, J. Díaz, and J. Gabarró: “Structural Complexity I” and “Structural Complexity II”; Springer-Verlag (1988/90).

[Bernays 1942] Bernays, P.: “A system of axiomatic set theory”; Journal of Symbolic Logic 7(1942), 65–89.

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

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

[Cucker 1993] Cucker, F.: “On the complexity of quantifier elimination: the structural approach”; The Computer Journal 36 (1993), 399-408.

[Bournez et al. 2006] Bournez, O., F. Cucker, P. J. de Naurois, and J.-Y. Marion: “Implicit complexity over an arbitrary structure: Quantifier alternations”; Information and Computation 202, 2 (2006), 210–230.

[Emerson 1994] Emerson, T.: “Relativizations of the $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ question over the reals (and other ordered rings)”; Theoretical Computer Science 133 (1994), 15–22.

[Gaßner 1997] Gaßner, C.: “On NP-completeness for linear machines”; Journal of Complexity 13 (1997), 259–271.

[Gaßner 2001] Gaßner, C.: “The P-DNP problem for infinite abelian groups”; Journal of Complexity 17 (2001), 574–583.

[Gaßner (1) 2004] Gaßner, C.: “Über die Konstruktion von Strukturen endlicher Signatur mit P = NP”; Preprint-Reihe Mathematik, E.-M.-Arndt-Universität Greifswald 1 (2004).

[Gaßner (2) 2004] Gaßner, C.: "NP $\subset$ DEC und P = NP für Expansionen von Erweiterungen von Strukturen endlicher Signatur mit Identitätsrelation"; Preprint-Reihe Mathematik, E.-M.-Arndt-Universität Greifswald 13 (2004).

[Gaßner (1) 2006] Gaßner, C.: "A structure with P = NP"; CiE 2006. Computer Science Report Series of the University of Wales Swansea CSR 7 (2006), 85–94.

[Gaßner (2) 2006] Gaßner, C.: "Expansions of structures with P = NP"; CiE 2006. Computer Science Report Series of the University of Wales Swansea CSR 7 (2006), 95–104.

[Gaßner 2007] Gaßner, C.: "P = NP for expansions derived from some oracles"; CiE 2007. Technical report no. 487, (June 2007), 161-169.

[Gaßner (1) 2008] Gaßner, C.: "A hierarchy below the Halting Problem for additive machines"; Theory of Computer Systems (2008) 43 (3), 464–470.

[Gaßner (2) 2008] Gaßner, C.: "On the power of relativized versions of P, DNP, and NP for groups"; (2008) Submitted.

[Meer and Ziegler 2005] Meer, K. and M. Ziegler: "An explicit solution to Post's problem over the reals"; 15th International Symposium on Fundamentals of Computation Theory LNCS 3623 (2005), 456–467.

[Meer and Ziegler 2006] Meer, K. and M. Ziegler: "Uncomputability below the real Halting Problem"; CiE2006, LNCS 3988 (2006), 368–377.

[Tucker and Zucker 2000] Tucker, J. V. and J. I. Zucker: "Computable functions and semicomputable sets on many-sorted algebras"; Handbook of Logic in Computer Science 5 (S. Abramskz, D. Gabbay, T. Maibaum Eds.), Oxford University Press (2000), 317–523.