

Electronic Notes in Theoretical Computer Science 221 (2008) 71–83

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

On Relativizations of the P =? NP Question for Several Structures

Christine Gaßner[1](#_bookmark0)*,*[2](#_bookmark0)

*Institut fu¨r Mathematik und Informatik, Ernst-Moritz-Arndt-Universita¨t, F.-L.-Jahn-Straße 15 a, 17487 Greifswald, Germany*

**Abstract**

We consider the uniform model of computation over arbitrary structures with two constants. For several structures, including structures over the reals, we construct oracles which imply that the relativized versions of P and NP are equal or are not equal. Moreover we discuss some special features of these oracles resulting from the undecidability of halting problems in order to explain the difficulties to define structures of finite signature which satisfy P = NP. We show that there are oracles which lose their non-deterministic self- reducibility which is sufficient for a recursive definition if their elements are compressed to tuples of fixed length.

*Keywords:* BSS machines, oracle machines, relativizations, P-NP problem, Halting Problem

# Introduction

The uniform model of computation over arbitrary algebraic structures K can be defined in analogy to the BSS model over the real numbers introduced by L. Blum,

M. Shub, and S. Smale [5, 4]. For the structure K 0*,*1 =df ( 0*,* 1 ; 0*,* 1; ; =) which

*{ } { }*

is also the basic structure for Turing machines (compare [2]) and for structures like the ordered ring of reals used in case of the BSS model, questions like P =? NP are open. For the classical setting, T. Baker, J. Gill, and R. Solovay [1] constructed

relativized versions of P and NP which imply different relationships between these classes. There are oracles such that the classes P*O* and NP*O* are equal and other oracles such that they are not equal. T. Emerson [10] transferred these results to the ring of reals and other ordered rings. 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 [5]. In this way the authors showed that, in both settings, for Turing machines as well as for

*O*

1 Email: [gassnerc@uni-greifswald.de](mailto:%20gassnerc@uni-greifswald.de)

2 I thank Robert Bialowons and Rainer Schimming for helpful hints and I thank the referee for his comments.

1571-0661 © 2008 Elsevier B.V. Open access under [CC BY-NC-ND license.](http://creativecommons.org/licenses/by-nc-nd/3.0/)

doi:10.1016/j.entcs.2008.12.008

BSS machines, the extension of the machines by oracles is not very useful for solving the central problems like P =? NP. This implies questions like the following for any structures K: Which relationships between the relativized versions of PK and NPK will we obtain if we permit oracles for machines over K? Can we provide evidence that the construction of new oracles is not really helpful for solving the P =? NP

problem, by defining oracles *0* and *Q* satisfying PK*O* = NPK*O* and P*Q*K */*= NK*Q*P for

structures for which the relation between PK and NPK is known? Is it possible to derive new relations of fixed arity from these oracles in order to get PM = NPM for new structures M?

# The Model of Computation

Let struc(*U* ) be the class of structures K = (*U* ; (*dj*)*j∈J*0 ; (*fj*)*j∈J*1 ; (*Rj*)*j∈J*2 *,* =) with the constants *dj ∈ U* , the operations *fj*, and the relations *Rj*. Any of these oper- ations, *fj*, has some fixed arity *nfj ≥* 1 and any relation *Rj* has some fixed arity *nRj* . For any K struc(*U* ), we define the K-machines in analogy to [5, 24, 11] such that we get a natural format of abstract computers for this kind of structures, on the one hand, and such that one has to consider only a small number of kinds of instructions, on the other hand.

*∈*

Every K-machine *ł* is equipped with registers *Z*1*, Z*2*,...* for the elements of

*U* and with a fixed number of registers *I*1*, I*2*,..., Ik* for indices in N+ = N *\ {*0*}*.

For an input (*x ,...,x* ) *∈ U∞* = *∞ Ui*, the se*M*quence *x ,...,x ,x ,x ,...* is

1

*n*

df

*i*=1

1

*n*

*n*

*n*

assigned to the registers *Z*1*, Z*2*,.. .*. 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 *Zj* := *fk*(*Zj*1 *,..., Zjn*

*fk*

)*, Zj* := *dk*, *ZIj* := *ZIk* ,

and *if cond then goto l*1 *else goto l*2 where *cond* can be of the form *Zj* = *Zk*

or *Rk*(*Zj*1 *,..., Zjn*

*R*

*k*

). The K-machines perform these instructions as a computer.

Each function and each relation of K is processed within a fixed time. The index registers are used in the copy instructions. For useful copying, we also allow *Ij* := 1, *Ij* := *Ij* + 1, and *if Ij* = *Ik then goto l*1 *else goto l*2. Moreover, *oracle machines* can execute *if* (*Z*1*,..., ZI*1 ) *then goto l*1 *else goto l*2 for some oracle *U∞*. The *non-deterministic* machines are able to guess an arbitrary number of arbitrary elements *y*1*,..., ym ∈ U* in one step after the input and to assign the guesses

*∈0 0 ⊆*

to *ZI*1+1*,..., ZI*1+*m*. Note, that we do not restrict the domain for *m* to simplify matters. *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*,..., xn*) is, by definition, its length *n*. If the *output* instruction is reached, then (*Z*1*,..., ZI*1 ) is the output and the machine halts.

Let MK and MN be the sets of deterministic and non-deterministic K-machines, respectively. Let, moreover, the machines in MK(*0*) and MN(*0*) be able to use the

K

K

oracle *0*.

Let us assume in the following that the considered structures contain two con- stants *a* = *d*1 and *b* = *d*2. We denote the class of these structures by struc*a,b*(*U* ).

Then we say that a deterministic K-machine *accepts* (or *rejects*, respectively) a tu- ple *→x U∞* if the machine outputs *a* (or *b*, respectively) on input *→x*. A K-machine *accepts* an input (*x*1*,..., xn*) *U∞ non-deterministically* if there is some finite sequence of guesses (*y*1*,..., ym*) *U∞* such that outputs *a* on input (*x*1*,..., xn*) for the guesses *y*1*,..., ym*. The execution of one instruction is one step of the com- putation process. That means that each step can be executed in a fixed time unit and that the cost of an instruction is 1. A K-machine will come to a halt *in poly- nomial time* if there is a polynomial *p* such that, on every input (*x*1*,..., xn*) *U∞* (and for any guesses), the machine performs at most *p*(*n*) instructions before the output is generated. The *decidability* and the *recognition* (or *semi-decidability*) of a problem *P over* K results from the computability of its (partial) characteristic

*∈*

*∈ ł*

*ł ∈*

*∈*

function *fP* : *U∞ → {a, b}* by some K-machine.

For any structure K, let PK and NPK denote the usual complexity classes of de- cision problems *U∞* decided or non-deterministically recognized by a machine in MK or in MN in polynomial time (where an input is only accepted if and only if it is in *P*). DECK contains all problems decided by a machine in MK. For any oracle *0*, PK*O*, NPK*O*, and DECK*O* denote the classes extended to machines which can also use *0*.

K

*P ⊆*

Let strucfin (*U* ) be the class of structures of finite signature of the form (*U* ; *a, b, d*3*,..., dk*0 ; *f*1*,..., fk*1 ; *R*1*,..., Rk*2 *,* =) for some *k*0 2 and *k*1*, k*2 0.

*a,b*

*≥ ≥*

For any structure K strucfin (*U* ), we can define *universal* deterministic and non-

*a,b*

*∈*

deterministic K-machines which are able to simulate the machines MK and MN, respectively, on any input *→x* if they get *→x* and a suitable code of as input. In order to encode the programs of these machines by strings which can be transformed into tuples in *U∞*, we consider strings over any alphabet *U* where *U* can also be infinite. The concatenation of any strings *s*1*, s*2 *∈ U∗* is denoted by *s*1*s*2, and for *r ∈ U∗* and *S, S*1*, S*2 *⊆ U∗*, we have *S*1*S*2 = *{s*1*s*2 *| s*1 *∈ S*1 & *s*2 *∈ S*2*}*,

K

*ł ∈ ł*

*ł ∈*

*rS* = *{r}S*, and *Sr* = *S{r}*.

**Definition 2.1** *Let S*code =df *b*2(*{a, b}∗\*(*{a, b}∗b*2*{a, b}∗*)) *be a set of strings which are suitable to be codes and which contain the sub-string b*2 *as preﬁx only. Let Code∗*K *be an injective mapping of the set of all deterministic and non-deterministic oracle*

K*-machines into* code *such that every character of the program is unambiguously translated into a string by this mapping where the oracle queries are encoded in- dependent of the used oracle by taking the same characters as codes for all oracle queries.*

*S*

Note that we omit the index 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*,..., ck*) *∈ U∞* can be stored in *k* registers.

**Definition 2.2** *For every non-empty string s* = *c*1 *··· ck ∈ U∗ where |s|* = *k ≥* 1*, let [s| be the* representation of *s* in the form of a tuple (*c*1*,..., ck*) *∈ Uk ⊂ U∞, that means that [c*1 *··· ck|* = (*c*1*,.* *, ck*)*.*

To simplify matters, we use the vector notation for the tuples and for the parts

of tuples. (*→x, [c*1 *··· ck|*) stands for (*x*1*,..., xn, c*1*,..., ck*), and the like. Moreover, for any *t ≥* 1, *t*˜ stands for *[bta|* and *Code* is defined by *Code*(*ł*) = *[Code∗*(*ł*)*|* for any machine *ł*.

**Definition 2.3** *Let the* Universal NPK-Problem*, the* Halting Problem*, and a* spe- cial halting problem with respect to K *∈* strucfin (*U* ) *be given by*

*a,b*

UNIK = *{*(*t*˜*, →x, Code*(*ł*))*| →x ∈ U∞* & *ł∈* MN & *ł* accepts *→x* within *t* steps*},* HK = *{*(*→x, Code*(*ł*)) *| →x ∈ U∞* & *ł∈* MK & *ł* halts on *→x},*

K

Hspec = *{Code*(*ł*) *|ł∈* MK & *ł* halts on *Code*(*ł*)*}.*

K

The first problem can be recognized by a universal non-deterministic machine in polynomial time. We can generalize some known results.

**Proposition 2.4** *For each structure* K *∈* strucfin (*U* )*,* UNIK *is* NPK*-complete.*

*a,b*

**Corollary 2.5** *For each structure* K *∈* strucfin (*U* )*, we have*

*a,b*

1. PK = NPK *if and only if* UNIK *∈* PK*,*
2. PK */*= NKP*if* UNIK */*D*∈*ECK*.*

Let us mention that the finite signature of the structure is a sufficient but not a necessary assumption for the definition of NPK-complete problems. For example, for linear Rlin-machines over the reals and for scalar Zsc-machines over the integers which can only execute the multiplication by constants, we can encode the constant factors by themselves, but there is not a universal machine (see [24, 13]). How- ever, although there is not any NPZsc -complete problem, there are NPRlin -complete problems (see [13]).

The undecidability of the Halting Problem is known for Turing machines, for BSS machines, for *While* programs on standard algebras [28], 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 [5, 4, 28], respectively. For BSS machines and restricted classes of BSS machines, further halting problems were considered, for instance, in [25] and in [12].

**Proposition 2.6** *For each* K *∈* struc*a,b*(*U* )*,* HK *∈* DECK *implies* Hspec *∈* DECK*.*

K

**Proposition 2.7** *For each* K *∈* struc*a,b*(*U* )*,* Hspec */*D*∈*ECK*.*

K

**Proof.** Assume that there is a K-machine *ł*0 which decides Hspec. Let *ł*1 be the following machine. *ł*1 works as *ł*0 until the output instruction of *ł*0 is reached, *ł*1 does not halt if the output of *ł*0 is *a*, and *ł*1 halts if the output of *ł*0 is *b*.

K

That means, that *ł*1 executes instructions like

*l* : *Z*2 := *a*; if *Z*1 = *Z*2 then goto *l*; output *Z*1

iff *ł*0 executes an output instruction of the form

*l* : output *Z*1*.*

Therefore, *ł*1 halts on *Code*(*ł*1) iff the output of *ł*0 on *Code*(*ł*1) is *b*, and consequently, iff *Code*(*ł*1) is not in Hspec, and thus, iff *ł*1 does not halt on *Code*(*ł*1). This is a contradiction.

K

**Corollary 2.8** *For each* K *∈* struc*a,b*(*U* )*,* HK *is not decidable by a* K*-machine.*

# The Equality of Relativized Versions of P and NP

We shall define a universal oracle *0* with PK*O* = NPK*O* for any structure K which

permits to compute the codes of the programs of machines over K. The first con- struction is restricted to structures of finite signature with two constants. Then, we can explicitly encode the programs of machines character-by-character similarly as in [10]. We transfer and modify the definitions given in [1] and [10]. 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 [1].) The tuples which can occur in the oracles

(K) (K)

1(= 1 ) and 2(= 2 ) (for a given K struc(*U* ), we omit the index K) have

*0 0 0 0 ∈*

the same form as the elements of a universal problem.

**Definition 3.1** *For any* K *∈* strucfin (*U* )*, let*

*a,b*

UNI(K)(*0*) = *{*(*t*˜*, →x, Code*(*ł*)) *| →x ∈ U∞* & *ł∈* MN(*0*) & *ł*(*→x*) *↓t}*

1

K

*be the* Universal NPK*O*-Problem *where ł*(*→x*) *↓t means that ł accepts →x for some guesses within t steps. Let 0*1(= *0*(K)) *be a* universal oracle *deﬁned by 0*1 =

1

*i≥*0 *Wi where W*0 = *∅ and*

*Wi* = *{*(*t*˜*, →x, Code*(*ł*)) *∈ Ui |ł∈* MN( *Wj*) & *ł*(*→x*) *↓t}.*

K

*j<i*

For any oracle *0*, UNI(K)(*0*) is NP*O*-complete since the codes of machines allow

1

K

to simulate the single steps of the oracle machines using the oracle *0* by only one

universal oracle machine in polynomial time. Moreover, for any *i ≥* 0, we have

UNI(K)(*0*1) *∩ Ui* = UNI(K)( *Wj*) *∩ Ui* = *Wi* since the length of a tuple in an

1

1

*j<i*

oracle query, executed within the first *t* steps, is less than *t* + *n < i* for any input

(*x*1*,..., xn*). This implies UNI(K)(*0*1) = *0*1. Because of *0*1 *∈* P*O*1

we get the

1 K

following.

**Proposition 3.2** *For any* K *∈* strucfin (*U* )*, there is some 0 such that* PK*O* = NPK*O.*

*a,b*

**Remark 3.3** A further characterization of the power of the universal oracle *0*1

is possible by comparison of the classes PK*O*1 and NP*O*K1 with the classes of the

polynomial hierarchy PHK and the class PATK containing the problems recognized in polynomial alternating time (for the definitions of these classes see [2, 7–9]). For any K *∈* strucfin (*U* ), we know that PHK *⊆* PATK [9] and PATK *⊆* P*O*1 [18].

*a,b*

K

The mentioned NP-completeness of UNI(K)( ) is not a necessary assumption for the construction. Proposition 3.2 can be generalized to any structure K if every oracle machine can be encoded by a computable tuple *→u ∈V* =df *{[v|∈ U∞ | v ∈*

1

*0*

*b*2(*U∗* (*U∗b*2*U∗*)) . For structures of enumerable signature, the possible codes are the indices of a list of all programs as in the definition in [1], 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 *0*,

*\ }*

UNI(K)(*0*) = *{*(*t*˜*, →x, →u*) *∈ U∞ |*

2

*→u ∈V* & (*∃ł ∈* MN(*0*))(*→u* is the code of *ł* & *ł*(*→x*) *↓t*)*}*

K

be a universal problem restricted to non-deterministic K-machines which can use

*0*. UNI(K)(*0*) is NP*O*-hard if every code of a machine *ł* can be computed by a

2

K

deterministic K-machine *NM* on any input *→x*. For *0*2(= *0*(K)) = *Wi* defined

by *W*0 = *∅* and

*Wi* = *{*(*t*˜*, →x, →u*) *∈ Ui |*

2 *i≥*0

*→u* *∈V* & (*∃ł ∈* MN( *j<i Wj*))(*→u* is the code of *ł* & *ł*(*→x*) *↓t*)*},*

K

there holds UNI(K)( 2) *Ui Wi* for any *i >* 0 if the codes (including the oracle queries) are independent of the used oracle. This implies the following.

2

*0 ∩ ⊆*

**Proposition 3.4** *For any* K struc*a,b*(*U* )*, for which the oracle machines can be encoded by computable tuples in U∞ independently of the used oracle, there is some oracle 0 such that* PK*O* = NPK*O.*

*∈*

# The Inequality of Relativized Versions of P and NP

We shall present three kinds of oracles *Q*1(= *Q*(K)), *Q*2(= *Q*(K)), and *Q*3(= *Q*(K))

1

2

3

for several structures K, in order to get the inequality between the corresponding relativized classes. The first two oracles are defined recursively by means of diago- nalization techniques. These techniques were also used by Gill, Baker, Solovay, and

R. Ladner (for details see [1]) and Emerson [10]. We simplify and generalize the construction for Archimedean rings given by Emerson and for special groups in [19].

* 1. *The Classical Way to Deﬁne the First Kind of Oracles*

If K is in the class strucenum(*U* ) of structures of enumerable signature, then the wished oracle can be defined recursively on the numbers of programs as in [1]. We take positive integers in order to

*a,b*

* + - enumerate all programs of oracle machines whose form (including the oracle queries) is independent of the used oracle,
    - encode all polynomials which can be used to define time bounds for the compu- tation processes,
    - encode all couples of polynomials and programs.

Let *i ∈* N+ be the code of a pair (*pi, Pi*) which determines a class of deterministic oracle K-machines *{NiB |B ⊆ U∞}* by the following.

* + - 1. The machine *NiB* performs the instructions of the program *Pi*.
      2. If *NiB* queries an oracle, then *NiB* uses the oracle *B*.
      3. The number of the instructions of *Pi* carried out by *NiB*

counted by *NiB* by means of an additional index register.

is simultaneously

* + - 1. For any input in *Un*, the machine *iB* halts after at most *pi*(*n*) steps of the execution of *Pi*. (The bound *pi*(*n*) can be computed by using index registers.)

*N*

* + - 1. If the output of *Pi* is reached in this time, then *iB* outputs the value deter- mined by *Pi*. If the output instruction of *Pi* is not reached in this time, then *NiB* rejects the input.

*N*

Then, for any oracle *B* and any problem in P*B*K there is an *i ≥* 1 such that the machine *NiB* decides this problem.

## The Construction of *Q*1.

*Let V*0 = *∅ and m*0 = 0*. We construct the set Q*1 *in stages.*

*Stage i* 1*: Let ni be any integer such that ni > mi−*1 *and pi*(*ni*) + *ni <* 2*ni .*

*≥*

*Moreover, let*

*Wi* = *j<i Vj,*

*Vi* = *{→x ∈ {a, b}ni | NWi rejects* (*a,..., a*) *∈ Uni*

*i*

& *→x is not queried by NWi on input* (*a,..., a*) *∈ Uni },*

*i*

*mi* = 2*ni .*

*Finally, let Q*1 = *i≥*1 *Wi and L*1 = *{→y |* (*∃i ∈* N+)(*→y ∈ Uni* & *Vi /*=*∅*)*}.*

**Lemma 4.1** *L*1 *∈* NPK*Q*1 *\* PK*Q*1 *.*

**Proposition 4.2** *For any structure* K *∈* strucenum(*U* ) *there is an oracle Q such*

*a,b*

*that* PK*Q /*= NK*Q*P*.*

* 1. *The Second Kind of Oracles*

Now, we want to consider mainly structures K whose signature and, consequently, the programs of oracle machines over K are not countable. Let us assume that, for any oracle *B*, all machines in MK(*B*) can be encoded by tuples in a set *U ⊆ U∞* independently of the used oracle such that each *→u ∈ U* represents a pair (*p→u, P→u*) which determines a class of deterministic oracle K-machines *→uB U∞* sat- isfying the properties analogously to (a), (b), (c), (d), and (e). Again this implies that, for any problem in P*B*K, there is some *→u* such that *→uB* decides this problem in polynomial time.

*∈U N*

*{N | B ⊆ }*

In order to get P*Q*

K

R

*/*= NK*Q*P for the structure KR = (R; R; +*, −, ·* ; *≤,* =) (where

any real number can be a machine constant) Emerson constructed a new kind of oracles. For any program *P→u* and any polynomial *p→u*, he considered the greatest absolute value of all numbers used in a query by one of the oracle machines in

R

*→uB U∞* if these machines get their own code *→u* as input. In order to define some oracle recursively, for any natural number *n >* 0, he summarized all codes of

*{N |B ⊆ }*

(*p→u, P→u*) for which this greatest value is in the interval ]*n* 1*, n*]. 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 are assumed. We can extend his investigation in two directions.

*—*

1. We permit any structure K with an infinite universe *U* which allows to define the necessary codes by tuples in *U∞*.
2. Since *U* is infinite, we shall assume that there is an element *α*0 and an injective mapping *σ* : *U U* satisfying *σ*(*αi*) = *αi*+1 and *αi*+1 =*α*0 for all *i* N. The mapping does not need to belong to the structure and it is not necessary that this mapping can be defined or computed over K. We denote the infinite sequence of images by ¯1*,* ¯2*,...* where *n*¯(= *n*¯K) =df *σ*(*αn−*1) for any *n ∈* N+.

*→ / ∈*

**Remark 4.3** In this way we also answer the three questions posed by Emerson in the last section of [10]. 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 α*0*, α*1*,.. .* 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 [3] and which is used instead of the general AC in the Analytical Topology can be sufficient. 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 (*Gi*)*i≥*0 whose members include their predecessors properly. Moreover, this implies the existence of an injective mapping *σ* by DC where *σ*(*αi*) *∈ Gi*+1 *\ Gi*.

## The Construction of *Q*2. ¯ ¯

*Let us assume that U contains an inﬁnite sequence* 1*,* 2*,... given by an injective mapping σ described above. Let V*0 = *∅. We construct the set Q*2 *in stages.*

*Stage i ≥* 1*: Let*

*Ki* = *{→u ∈U |* (*∀j > i*)(*∀B ⊆ U∞*)

(*N→uB does not compute or use the value* ¯*j on input →u*)*},*

*Wi* =

*k<i Vk,*

*Vi* = *{*(*i* + 1*, →u*) *| →u ∈ Ki* & *NWi rejects →u}.*

*Finally, let Q*2 = *i≥*1 *Wi and L*2 = *{→y |* (*∃n ∈* N )((*n*¯*, →y*) *∈ Q*2)*}.*

*→u*

+

**Lemma 4.4** *L*2 *∈* NPK*Q*2 *\* PK*Q*2 *.*

**Proposition 4.5** *For any structure* K struc*a,b*(*U* ) *with an inﬁnite universe U which allows to encode the* K*-machines by means of tuples in U∞ independently of the used oracle, there is an oracle Q such that* PK*Q /*= NK*Q*P*.*

*∈*

**Remark 4.6** Simpler constructions are possible if there is an element which is not computable from the codes in *U* . For instance, the deterministic oracle machines

over KQ*,√*2 = (Q*√*2+Q; 0*,* 1; +*, —, ·* ; =) can be encoded by integers *i ∈* N. Then, the

inequalities DECK*Q √ /*= NK*Q*P *√* and thus P*Q*K *√ /*= NK*Q*P *√* hold if *Q* = *{*(*√*2*, i*) *|*

*Ui∅* rejects *i}*.

Q*,* 2

Q*,* 2

Q*,* 2

Q*,* 2

**Remark 4.7** The construction given by Emerson was simplified and generalized especially in order to show Proposition 4.5 for any structure of non-enumerable signature. However, for structures K like the ordered ring over the reals we can

KR

prove P*Q /*= N*Q*Pfor further oracles *Q*. We will show that Q *∈* NPZ

K

K

KR

*\* PZ

. Note

that the proofs are the same for the unordered ring.

1. *Proof for* Q *∈* NPZ

KR

*.* Q can be non-deterministically recognized by a machine

in MN

K

R

(Z) which queries the oracle whether the guesses *y*1 and *y*2 are integers and

which checks *y*1 */*= 0 an*y*d1*x* = *y*2 for any input *x*.

1. *Proof for* Q */*P*∈*Z *.* Assume that there is a machine *U* in MK

KR

R

(Z) which decides

Q in polynomial time. The decidability of a set of reals in polynomial time means that there is a number *t*0 1 such that any input *x* R is accepted or rejected within *t*0 steps. Consequently, the number of computation paths of traverse by the inputs *x* R is finite. Thus, there is a finite set *M* = *p*1*,..., pm* containing polynomial functions of arity 1 and degree *d* 1, such that each of these paths, *P* ,

*≥*

*∈ { }*

*U*

*≥ ∈*

can be described by a system *SP* consisting of conditions of the form *pk*(*x*) 0, *pk*(*x*) *>* 0, *pk*(*x*) Z, and *pk*(*x*) Z where *k m*. An input *x* traverses a path *P* if and only if it satisfies *SP* (for more details, compare also [12]). Moreover, *X* = *{x |* (*Ek ≤ m*)(*pk*(*x*) *∈* Z)*}* is countable. Therefore, the set R *\* (Q *∪ X*) is non-empty and it contains a real number *r* which is rejected by *U* . Let *Pr* be the computation path of *U* traversed by *r*. Because of *r /∈X*, *SPr* does not contain conditions of the form *pk*(*x*) *∈* Z. If a condition of the form *pk*(*x*) *≤* 0 belongs to *SPr* , then *pk*(*r*) *<* 0 holds. For any sequence of rational numbers (*qi*)*i∈*N with limit *r* there is an *i*0 N such that, for all *i i*0, *SPr* is also satisfied by *qi*. This is a contradiction to *qi* Q since we suppose that any computation path is either an accepting path or a rejecting path.

*∈*

*∈ ≥*

*∈ /∈ ≤*

*≤*

* 1. *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

strucN¯ (*U* ) containing all structures K *∈* strucfin (*U* ) for which *U* includes an infinite

*a,b*

*a,b*

set N¯ = *{*¯0*,* ¯1*,* ¯2*,.. .}* with the following properties.

* + - N¯ is defined by some injective mapping *σ* of *U* into *U* where *i* +1 = *σ*(¯*i*) */*=¯0.
    - N¯ is decidable by a deterministic K-machine.
    - N¯ is enumerable by a deterministic K-machine which can compute ¯0 independently of the input and which can compute *i* + 1 from ¯*i*.

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. We can build, for instance, some oracle *Q* such that EXPK*Q /*= NK*Q*Pholds if we use the exponential

functions as time bounds. The next oracle implies the corresponding inequalities for each class of time bounds.

## The Definition of *Q*3.

*For* K *∈* strucN¯ (*U* )*, let*

*a,b*

*Q*3 = *{*(*t*¯*, →x, Code*(*ł*)) *∈ U∞ |ł∈* MK & *t ∈* N+ & *ł*(*→x*) *↓t}.*

**Lemma 4.8** *For any* K *∈* strucN¯ (*U* )*,* HK *∈* NP*Q*3 *and* P*Q*3 *⊆* DECK*.*

*a,b*

K

K

By Corollary 2.8 we can conclude the following.

**Proposition 4.9** *For any* K *∈* strucN¯ (*U* ) *there is some oracle Q such that* P*Q /*=

*a,b*

K

NPK*Q.*

**Remark 4.10** The results can be transferred to structures of infinite signature if they contain only finitely many relations and operations, for instance, to the structure KR = (R; R; +*, —, ·* ; *≤,* =).

# Relations Instead of Oracles?

Since we do not know the answer for the classical problem P =? NP, we should study the properties of all known structures K and the relationships between the classes PK and NPK (like, for instance, in [23, 24, 21, 26, 6, 11]) and we should investigate several possibilities to construct structures K with PK = NPK (compare [26, 22, 14– 18, 20, 27]). Inspired by a construction of a structure K of infinite signature with PK = NPK given by G. Mainhardt [22] where an infinite number of relations was derived from a universal NPK-problem, we want to discuss the following question. Is it possible to replace the oracle (K) for some K by one additional relation of fixed arity in order to get a structure M of finite signature with PM = NPM?

1

*0*

If we want to derive a new relation *R* (which can be satisfied only by tuples of a fixed length *nR*) from the oracle *0*1 such that any oracle query (*Z*1*,..., ZI*1 ) *∈0* can be replaced by a condition of the form *R*(*Z*1*,..., ZnR* ), then we have to compress the tuples in 1 to tuples of length *nR*. Since, for many structures, it is not possible

*0*

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 5.1** *For an arbitrary universe U, let A* = *U∗ such that the elements of U are the characters of the strings in A, and let* struc*∗*(*A*) *be the class of structures of the form* ( ; 0; *f*1*,..., fk*1 *,* add*,* subl*,* subr; *R*1*,..., Rk*2 *,* =) *where* 0 *is a ﬁnite set of constants and a, b, ε* 0*.* add *is a binary operation for adding a character to a string.* subr *and* subl *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 , r U, and c U by* add(*s, c*) = *sc,* subl(*sc*) = *s,* subr(*sc*) = *c,* add(*s, r*) = *ε,* subl(*ε*) = *ε, and* subr(*ε*) = *ε. Each fi is an operation on A. Each Ri is a relation on A.*

*∈ A*

*A A A ⊆A*

*∈ A ∈A \ ∈*

**Lemma 5.2** *{bi | i ∈* N*} is decidable and enumerable over* K *∈* struc*∗*(*A*)*.*

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

**Definition 5.3** *For every string s ∈ A, let the value ⟨s⟩ be recursively deﬁned by*

*⟨ε⟩* = *a and ⟨rc⟩* = *⟨r⟩ca for all strings r ∈ A and all character c ∈ U. For every integer n >* 1 *and every tuple →s* = (*s*1*,..., sn*) *∈ An, let ⟨s*1*,..., sn⟩ be the string*

*⟨s*1*⟩b*2 *··· ⟨sn−*1*⟩b*2*⟨sn⟩.*

Although the elements of the oracles *0* = *0*(K) and *Q* = *Q*(K) (for any

1

3

K struc*∗*( )) have a similar form, we have different relationships between the

*∈ A*

relativized versions of PK and NPK. That implies, on the one hand, the conjec-

ture that it could be easy to define oracles *0*¯*, Q*¯ *⊆ A* or new unary relations *R*

by compressing the sequences of strings in *0, Q ⊆ A∞* to single strings in order

to get P*O*¯ = NP*O*¯ and P*Q*¯ */*= N*Q*¯Pand PK = NPK or PK */*= NKP for new

K

K

K

K

*R*

*R R R*

structures K*R*. On the other hand it implies the conjecture that it is not possible to define oracles *0*¯ *⊆ A* with P*O*¯ = NP*O*¯ since the different relationships between the

K

K

complexity classes, relativized by using the oracles and , respectively, mainly are the result of the different representation of the number of steps: In case of , the number of possible steps, *t*, is determined by the length of the tuple *t*˜. In case

*0*

*0 Q*

of , the number of steps is given by only one element of the structure. To use only single strings as codes of the elements of in defining a new oracle ¯ could be easier said than done. The following results bear out that. They follow from the

*0 0*

*Q*

undecidability of Hspec and

K

Hspec(*0*¯) = *{Code*(*ł*) *∈ A∞ | ł∈* MK(*0*¯) & *ł* halts on *Code*(*ł*)*}.*

K

**Theorem 5.4** *For any* K *∈* struc*∗*(*A*)*, the oracle*

*Q*¯ = *{bt⟨→x⟩Code∗*(*ł*) *∈A|ł∈* MK & *t ∈* N+ & *ł*(*→x*) *↓t}*

*implies* P*Q*¯ */*= N*Q*¯P*.*

K

K

Whereas it is easy to transfer the construction of oracles in order to again obtain inequalities between the relativized polynomial time complexity classes for struc- tures 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 5.5** *For any* K *∈* struc*∗*(*A*)*, there is not any oracle satisfying*

*bt⟨→x⟩Code∗*(*ł*) *∈ 0*¯ *⇔ ł ∈* MN(*0*¯) & *t ∈* N+ & *ł*(*→x*) *↓t .*

K

Each deterministic machine over K *a,b ∗* = ( *a, b ∗*; *a, b, ε*; add*,* subl*,* subr; =) 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 5.6** *The set*

*Q*¯TM = *{bt⟨→x⟩Code∗*(*ł*) *∈ {a, b}∗ |ł∈* TM & *t ∈* N+ & *ł*(*→x*) *↓t}*

*implies* DEC*Q*¯TM

K*{a,b}∗*

= N*Q*¯PTM

K*{a,b}∗*

*/*

*and, hence,* P*Q*¯TM

K*{a,b}∗*

= N*Q*¯PTM *.*

K*{a,b}∗*

*/*

**Remark 5.7** The last result remains true if we consider machines over the structure ( *a, b ∗*; *a, b, ε*; add*,* subl*,* subr;= *a,* = *b*) where any test has the form *Zj* = *a* or *Zj* = *b*.

*{ }*

**Remark 5.8** A possibility to define new relations *R* of arity 1 (or oracles containing

only single elements of the universe) derived from *0*(K) such that there holds PK =

1

*R*

NPK*R* for the new structures K*R*, is presented in [14, 15, 18]. The crucial idea is to define new relations *R* satisfied by padded codes of the elements of an NPK*R* - complete problems. (For more details see [16, 17], too.) The subject of [14] is the construction of a new structure of 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 PK =? NPK problem for any structure K. On the one hand, we know structures K with PK */*= NKPand we can define an oracle *0* which implies PK*O* = NPK*O*. On the other hand, we know structures M with PM = NPM and we can define an oracle *Q* implying P*Q*M */*= N*Q*MP.

# References

1. Baker, T., J. Gill, and R. Solovay (1975). Relativizations of the P =? NP question. *SIAM J. Comput.*

4, 431–442.

1. J. Balc´azar, J. D´ıaz, and J. Gabarr´o (1988/90). *Structural Complexity I* and *Structural Complexity*

*II*. Springer-Verlag.

1. Bernays, P. (1942). A system of axiomatic set theory. *Journal of Symbolic Logic* 7, 65–89.
2. Blum, L., F. Cucker, M. Shub, and S. Smale (1998). *Complexity and Real Computation.* Springer- Verlag.
3. Blum, L., M. Shub, and S. Smale (1989). 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, 1–46.
4. Cucker, F., M. Shub, and S. Smale (1994). Separation of complexity classes in Koiran’s weak model.

*Theoretical Computer Science* 133, 3–14.

1. Cucker, F. (1993) On the complexity of quantifier elimination: the structural approach. *The Computer Journal* 36, 399–408.
2. Bournez, O., F. Cucker, P. J. de Naurois, and J.-Y. Marion (2003). Safe recursion over an arbitrary structure. *ENTCS* 90, 3–14.
3. Bournez, O., F. Cucker, P. J. de Naurois, and J.-Y. Marion (2006). Implicit complexity over an arbitrary structure: Quantifier alternations. *Information and Computation* 202, 2, 210–230.
4. Emerson, T. (1994). Relativizations of the P =? NP question over the reals (and other ordered rings).

*Theoretical Computer Science* 133, 15–22.

1. Gaßner, C. (2001). The P-DNP problem for infinite abelian groups. *Journal of Complexity* 17, 574–583.
2. Gaßner, C. (2007). A Hierarchy below the Halting Problem for Additive Machines. *Theory of Computer Systems* DOI 10.1007/s00224-007-9020-y.
3. Gaßner, C. (1997). On NP-Completeness for Linear Machines. *Journal of Complexity* 13, 259–271.
4. Gaßner, C. (2004). U¨ ber die Konstruktion von Strukturen endlicher Signatur mit P = NP. *Preprint*

1/2004. [3](#_bookmark1)

3 *Preprint-Reihe Mathematik, E.-M.-Arndt-Universita¨t Greifswald*

1. Gaßner, C. (2004). NP DEC und P = NP fu¨r Expansionen von Erweiterungen von Strukturen endlicher Signatur mit Identit¨atsrelation. *Preprint* 13/2004.3

*@*

1. Gaßner, C. (2006). A structure with P = NP. *CiE 2006. Computer Science Report Series of the University of Wales Swansea* CSR 7, 85–94.
2. Gaßner, C. (2006). Expansions of structures with P = NP. *CiE 2006. Computer Science Report Series of the University of Wales Swansea* CSR 7, 95–104.
3. Gaßner, C. (2007). P = NP for Expansions Derived from Some Oracles. *CiE 2007. Technical report*

no. 487, June 2007, 161–169.

1. Gaßner, C. (2008). Computation over a group. *CiE 2008. Technical report*, University of Athens, June 2007, 147–156.
2. Hemmerling, A. (2005). P = NP for some structures over the binary words. *Journal of Complexity* 21, 557–578.
3. Koiran, P. (1994). Computing over the reals with addition and order. *Theoretical Computer Science*

133, 35–47.

1. Mainhardt, G. (2004). P versus NP and computability theoretic constructions in complexity theory over algebraic structures. *Journal of Symbolic Logic* 69, 39–64.
2. Meer, K. (1992). A note on a P = NP result for a restricted class of real machines. *Journal of Complexity*

*/*

8, 451–453.

1. Meer, K. (1993). Real number models under various sets of operations. *Journal of Complexity* 9, 366– 372.
2. Meer, K. and M. Ziegler (2005). An explicit solution to Post’s problem over the reals. *15th International Symposium on Fundamentals of Computation Theory* LNCS 3623, 456–467.
3. Poizat, B. (1995). *Les Petits Cailloux.* Al´eas.
4. Prunescu, M. (2006). Structure with fast elimination of quantifiers. *Journal of Symbolic Logic* 71, 321–328.
5. Tucker, J. V. and J. I. Zucker (2000). 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, 317–523.