T U M

I N S T I T U T F Ü R I N F O R M A T I K

Generic Separations and Leaf Languages

Matthias Galota, Sven Kosub, Heribert Vollmer

TUM-I0104
September 01

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

Generic Separations and Leaf Languages

Matthias Galota[1], Sven Kosub[2], and Heribert Vollmer[1]

[1] Theoretische Informatik, Julius-Maximilians-Universität Würzburg,
Am Hubland, D-97074 Würzburg, Germany
[2] Institut für Informatik, Technische Universität München,
D-80290 München, Germany

Abstract. In the early nineties of the previous century, leaf languages were introduced as a means for the uniform characterization of many complexity classes, mainly in the range between P (polynomial time) and PSPACE (polynomial space). It was shown that the separability of two complexity classes can be reduced to a combinatorial property of the corresponding defining leaf languages. In the present paper, it is shown that every separation obtained in this way holds for every generic oracle in the sense of Blum and Impagliazzo. We obtain several consequences of this result, regarding, e.g., simultaneous separations and universal oracles, resource-bounded genericity, and type-2 complexity.

Keywords: computational and structural complexity, leaf language, oracle separation, generic oracle, type-2 complexity theory.

1 Introduction

In 1975 Baker, Gill, and Solovay constructed an oracle B that separates P from NP: $\mathrm{P}^B \neq \mathrm{NP}^B$ [BGS75]. On the other hand, it is relatively easy to see that there is an oracle relative to which P and NP collapse (just pick any PSPACE-complete language). Hence, the existence of B shows that the P-NP-question is difficult to solve in the sense that non-relativizing proof techniques will be necessary. Conflicting relativizations like this give a sort of *independence result*: Relativizing arguments will neither be able to prove nor to disprove $\mathrm{P} \neq \mathrm{NP}$.

Many more such results have been established in complexity theory since; and certainly one would like to know which one of two conflicting oracles is nearer to the "actual", i.e., unrelativized, world without an oracle. Therefore, one has looked for notions of "typical" oracles with the hope that separations for such a restricted type of oracle also hold, if no oracle is present. One type of oracles that have been considered along this avenue are random oracles, where membership of words in the oracle is essentially determined by independent fair coin tosses. Bennett and Gill [BG81], introducing random oracles, established the *random oracle hypothesis*, stating that every separation relative to a random oracle also holds in the unrelativized case. Unfortunately, the random oracle hypothesis is false [CCG$^+$94], in fact, it even fails very badly, namely in the case that both occurring classes are defined with the same machine model [VW97] (technically: the leaf language model, see below).

Another type of oracles that have attracted a lot of attention are *generic oracles*, made popular in computational complexity theory by Blum and Impagliazzo [BI87] (but going back to much older developments in mathematical logic). Generic oracles are "typical" in the sense, that they have all the properties that can be enforced by a stage construction. Going back to recursion theory (see, e.g., [Soa87]), stage constructions have been the main technique to obtain oracles with certain desired properties. Baker, Gill, and Solovay also relied on this method; thus, a generic oracle separates P and NP, but since it has additional properties enforced by other stage constructions, it even makes the polynomial hierarchy infinite. When looking at the characteristic sequence of generic oracles, "anything even remotely possible will happen, and happen infinitely often" [BI87, p. 120]. For example, generic oracles will have infinitely often intervals of consecutive zeroes, whose length cannot be bounded recursively. In this sense, generic oracles are arbitrary, but not random.

Though a corresponding *generic oracle hypothesis* also does not hold [Fos93], there are a number of very good motivations to study generic oracles. We only mention two of them: First, generic relativizations (i.e., relativizations via generic oracles) do make statements about the real world. Blum and Impagliazzo showed that any language acceptable by a (time- or space-bounded) Turing machine with a generic oracle can be accepted without an oracle in essentially the same resources. This is a "half-sided" positive relativization result, since it implies that if resource-bounded classes coincide relative to a generic oracle, then they coincide absolutely. Also, Blum and Impagliazzo proved that, if P is not equal to UP under a generic oracle, then $\mathrm{P} \neq \mathrm{NP}$. (This result actually goes back to an older paper by Hartmanis and Hemachandra, see [HH90].) Hence, generic oracles immediately relate to a main motivation for a lot of current research in complexity theory, namely the P-NP-problem. Second, generic oracles proved to be very useful when looking for simultaneous collapses and separations; a number of such results can be found in [BI87].

Around the time of the appearance of random and generic oracles in complexity theory, *leaf languages* were introduced as a uniform way to characterize many complexity classes between P (polynomial time) and PSPACE (polynomial space) using only one computation model: the nondeterministic Turing machine running in polynomial time (NPTM). This way makes use of characterizations of classes in terms of conditions on the values printed at the leaves of computation trees of NPTMs. A leaf language is essentially nothing else than a set of finite sequences of such values. A word w given as input to a NPTM M is said to be accepted by M (with leaf language A) if the sequence of output values is a sequence in A. For example, the class NP can clearly be characterized by the condition "there is one leaf in the computation tree that outputs a 1"; thus the leaf language for NP consists of all binary sequence with at least one occurrence of the letter 1.

Interestingly, there are connections between leaf language definability and oracle separability. Bovet, Crescenzi, and Silvestri [BCS92] and Vereshchagin [Ver93] showed how the question of the separability of complexity classes by an oracle can be expressed equivalently by a relation among the characterizing leaf languages. In this way, the task to construct an oracle separation often reduces to a combinatorial problem, without the need to perform an actual stage construction.

In this paper, we relate the question if two classes can be separated by a generic oracle to a relation among the corresponding leaf languages. In much the same way as in the just cited results, the proof of a generic separation now becomes a combinatorial task. We also present a number of consequences of our result. For example, we show that all oracle separations among certain complexity classes (those that relativizably have an arithmetic complete set) hold simultaneously relative to all generic oracles. We further examine relationships between leaf languages and genericity with respect to bounded resources.

Another field of applications of our result concerns *type-2 complexity theory*. Type-0 objects are numbers or words, type-1 objects are functions (or relations) of type-0 objects, and type-2 objects are functions (or relations) of type-1 and type-0 objects. Thus, a type-2 function takes as input a word/number and a function (or relation/language) and outputs a word/number. Type-2 computations occur in different areas of mathematics and computer science, where a function is given as a "black box". For example in numerical mathematics, an algorithm for integration takes as input numbers a, b and a function f and computes $\int_a^b f(x)dx$. Function f is presented as a black box or oracle, that can be queried, and such an evaluation of f consumes no resources. Thus, the natural computation device for type-2 objects are oracle Turing machines. Type-2 objects and their complexity classes were studied already in the seventies by Constable and Mehlhorn [Con73, Meh76]. In a very interesting recent paper by Cook, Impagliazzo, and Yamakami [CIY97], the inclusion relations between type-2 classes are studied and related to inclusion relative to a generic oracle. Making use of our relation between leaf languages and generic oracles, we show that the leaf language approach completely clarifies the inclusion structure among "well-definable" type-2 complexity classes. This result again reduces the question of inclusion among these classes to properties of the involved leaf languages, without an oracle construction. Cook, Impagliazzo, and Yamakami also raised the issue if type-1 and type-2 computations relative to a generic oracle coincide. Again, for classes "well-definable" in a certain sense, we exhaustively answer this question.

Finally we discuss in our leaf language framework the possibility of separating P and UP by a generic oracle (and thus, by the Blum-Impagliazzo condition, proving P $\neq$ NP).

2 Preliminaries

Genericity

In the present paper, we use the notion of *generic oracles* as used by Blum and Impagliazzo in [BI87]. (It should be remarked that many different notions of genericity have been studied in computational complexity theory; for an overview the reader might consult [FFKL93, AS96]. For a discussion of various conditions of genericity in recursion theory, see [Kur83].)

We briefly repeat the central definitions from [BI87]. An *oracle O* is a set of natural numbers. We will identify such sets with their characteristic functions, i.e., an oracle is nothing else than a total function $O\colon \mathbb{N} \to \{0,1\}$. We will also consider *finite oracles*—these are partial functions $v\colon \mathbb{N} \to \{0,1\}$ whose domain, denoted by $D(v)$, is finite. In particular, finite oracles v whose domain is a finite prefix of the natural numbers, i.e., a set $\{0,1,\dots,n\}$ for some $n \in \mathbb{N}$, will also be identified with finite binary strings, namely $v(0)v(1)\dots v(n)$. In this vein, we also identify infinite oracles with infinite binary strings. The set of all oracles with finite prefix v thus is $v \cdot \{0,1\}^\omega$.

We say that a finite oracle w *extends* a finite oracle v, in symbols: $v \sqsubseteq w$, if $D(v) \subseteq D(w)$ and the functions v and w agree on $D(v)$. A (total) oracle O *extends* a finite oracle v if the functions O and v agree on $D(v)$.

As mentioned in the introduction, a generic oracle is intuitively an oracle that in a sense has all properties that can be enforced by a stage construction. During the stages of these constructions, usually finite oracles are extended. In order to be able to complete the next stage, at every stage we must have still enough possibilities for such an extension. Formally we need a property called "denseness":

A set D of finite oracles is *dense*, if every finite oracle v has an extension to a finite oracle $w \in D$.

Dense sets D should be thought of as those finite prefixes of oracles that fulfill (or, *meet*) the condition aimed at during a specific stage of a usual oracle construction. The result of a stage construction as a whole has to meet a countable number $D_1, D_2, \ldots$ of such conditions. In the separation of P from NP by Baker, Gill, and Solovay, the sets D_i consist of those finite prefixes of oracles, for which a set $L(O)$ (depending on oracle O) cannot be decided by the i-th deterministic polynomial time oracle Turing machine. Hence, if O meets all D_i, the resulting $L(O)$ cannot be in P^O. From the way $L(O)$ is defined in [BGS75], however, it is very easy to see that for every O, $L(O) \in \mathrm{NP}^O$, and in this way we obtain the desired relativized separation.

Generic oracles now are oracles that meet all conditions describable in a certain language. Here, we consider all conditions that result from sets in the arithmetical hierarchy (see, e.g., [Soa87]). Hence we define:

Let $\mathcal{C} = \{D_1, D_2, \ldots\}$ be a countable collection of dense sets of finite oracles. An oracle is $\mathcal{C}$-*generic* if, for each i, O has a finite prefix $w_i \in D$. A set D of finite oracles (or a set of finite words) is *arithmetic*, if membership in D can be expressed as a finite-length first order formula with recursive predicates. Let $\mathcal{A}$ be the collection of dense, arithmetic sets of finite oracles (in other words, the class of all dense sets from the arithmetical hierarchy [Soa87, Chap. VI]). An oracle is *generic* if it is $\mathcal{A}$-generic.

We will mainly consider generic sets in the sense of $\mathcal{A}$-generic; however we will also be interested in $\mathcal{C}$-generic oracles for a resource-bounded class $\mathcal{C}$ later.

Leaf Languages

In the leaf language approach to the characterization of complexity classes, the acceptance of a word given as input to a nondeterministic polynomial time Turing machine (NPTM) depends only on the values printed at the leaves of the computation tree. To be more precise, let M be a nondeterministic Turing machine, halting on each path printing a symbol from an alphabet Σ, with some order on the nondeterministic choices. Then, $\mathrm{leafstring}^M(x)$ is the concatenation of the symbols printed at the leaves of the computation tree of M on input x (according to the order of M's paths induced by the order of M's choices).

Call a computation tree of a machine M *balanced*, if M branches at most binary, all of its computation paths have the same length, and moreover, if we identify every path with the string over $\{0, 1\}$ describing the sequence of nondeterministic choices on this path, then there is some string z such that all paths y with $|y| = |z|$ and $y \preceq z$ (in lexicographic ordering) exist, but no path y with $y \succ z$ exists.

Given now a pair of languages $A, R \subseteq \Sigma^*$ such that $A \cap R = \emptyset$, this defines a complexity class $\mathrm{BLeaf}(A, R)$ as follows: A language L belongs to $\mathrm{BLeaf}(A, R)$ if there is an NPTM M whose computation tree is always balanced, such that for all x, $x \in L \implies \mathrm{leafstring}_M(x) \in A$ and $x \notin L \implies \mathrm{leafstring}_M(x) \in R$.

In the case that $A = \overline{R}$ we also simply write $\mathrm{BLeaf}(A)$ for $\mathrm{BLeaf}(A, R)$. The classes which can be defined by a pair $(A, \overline{A})$ are *syntactic classes* in the terminology of Papadimitriou [Pap94], while those which cannot are *semantic classes*, i.e., promise classes.

This computation model was introduced by Papadimitriou and Sipser around 1979, and published for the first time by Bovet, Crescenzi, and Silvestri, and independently by Vereshchagin [BCS92, Ver93] (see also the textbook [Pap94, pp. 504f]).

Important for us will be reductions among leaf languages, in particular: $\leq_m^{\mathrm{plt}}$-reductions as introduced in [BCS92, Ver93]. First, we define the class of those functions that will constitute our reductions:

A function $f\colon \Sigma^* \to \Sigma^*$ is *polylog-time bit-computable* if there exist two polynomial time oracle transducers $R\colon \Sigma^* \times \mathbb{N} \to \Sigma$ and $l\colon \Sigma^* \to \mathbb{N}$ such that, for any $x \in \Sigma^*$, $f(x) = R^x(|x|,1)R^x(|x|,2)\cdots R^x(|x|,l^x(|x|))$.

There are quite "natural" functions that have this property:

Lemma 1. *Let M be an balanced NPTM with oracle (NPOTM) with input alphabet $\{0,1\}$ and oracle alphabet Σ. For $x \in \Sigma^*$, the function $f(x) =_{\mathrm{def}} leafstring_{M^x}(|x|)$ is polylog-time bit-computable.*

Proof. The transducer l from the definition of $\leq_m^{\mathrm{plt}}$ in this case has to compute the number of paths in the balanced computation tree of M with input $|x|$ and oracle x. Since l itself has input $|x|$ and oracle x at its disposal, it can simulate M to achieve this.

The transducer R on input $(|x|,k)$ then has to compute the output of M on computation path k with input $|x|$ and oracle x. Again, since R can use $|x|$ and x directly, the simulation of path k of M is easy (note that the computation tree of M is balanced, hence the bits of the binary representation of k correspond to the nondeterministic choices M on that path). □

$\leq_m^{\mathrm{plt}}$-reductions are now defined as follows:

Let (A,R) and (A',R') be two pairs of languages. (A,R) is *polylog-time reducible* to (A',R') (in symbols: $(A,R) \leq_m^{\mathrm{plt}} (A',R')$), if there exists a polylog-time bit-computable function f, such that $f(A) \subseteq A'$ and $f(R) \subseteq R'$. We note that for the case $R = \overline{A}$ and $R' = \overline{A'}$ this is equivalent to $x \in A \Leftrightarrow f(x) \in A'$.

The importance of plt-reductions stems from that fact that these relate to oracle separations of leaf language definable classes, as proved in [BCS92, Ver93]. First, if in the definition of $\mathrm{BLeaf}(A,R)$ above, M is an NPOTM with access to oracle O, we denote the obtained class by $\mathrm{BLeaf}^O(A,R)$. The main result of [BCS92, Ver93] can now be stated formally as:

Theorem 2. *Let (A,R) and (A',R') be pairs of leaf languages. Then*

$$\mathrm{BLeaf}^O(A,R) \subseteq \mathrm{BLeaf}^O(A',R') \textit{ for all oracles } O \iff (A,R) \leq_m^{\mathrm{plt}} (A',R').$$

3 Plt-Reductions and Generic Oracles

Our main result relates the existence of separating generic oracles to non-plt-reducibility:

Theorem 3. *Let A and B be arithmetic sets. Then*

$$\mathrm{BLeaf}^G(A) \subseteq \mathrm{BLeaf}^G(B) \textit{ for all generic oracles } G \iff A \leq_m^{\mathrm{plt}} B.$$

In other words, two complexity classes can be separated with a generic oracle if and only if the corresponding leaf languages are not plt-reducible to one-another.

The implication from right to left of course immediately follows from Theorem 2. For the left to right implication, i.e., if $A \not\leq_m^{\text{plt}} B$ then there is a generic G such that $\text{BLeaf}^G(A) \not\subseteq \text{BLeaf}^G(B)$, informally, the proof proceeds as follows: For any oracle O, a *test language* $L(O)$ is defined, that is easily seen to be in $\text{BLeaf}^O(A', R')$. The desired oracle G is constructed by a stage construction such that in stage $2m+1$ the m-th dense arithmetic set is met, and in stage $2m$ it is ensured that the m-th $\text{BLeaf}(A)$-NPOTM does not accept $L(G)$. Thus, the odd stages will ensure that G is generic while the even stages will ensure that $\text{BLeaf}^G(A) \not\subseteq \text{BLeaf}^G(B)$. We remark that a similar construction was used by Foster in [Fos93], where a generic oracle separating IP from PSPACE was constructed. In that paper, the odd stages ensure that the resulting oracle is generic while the even stages ensure IP $\neq$ PSPACE relative to the constructed oracle.

The rest of this section is devoted to a formal proof of Theorem 3.

Proof. It is sufficient to prove the implication from left to right: Let A and B be arithmetic sets and let $\text{BLeaf}^G(A) \subseteq \text{BLeaf}^G(B)$ for all generic oracles G.

We define a function $\delta : \mathbb{N} \to \mathbb{N}$ as $\delta(n) = n + \frac{n(n-1)}{2}$. For every oracle O we define the test language $L(O)$ as $L(O) =_{\text{def}} \{ x \mid O(\delta(x)+1) \dots O(\delta(x)+x) \in A \}$. Then it holds for all oracles O: $L(O) \in \text{BLeaf}^O(A)$. By the assumption then also holds for all *generic* oracles G that $L(G) \in \text{BLeaf}^G(B)$. Let M_k be the k-th machine in an enumeration of all *balanced* NPOTMs. I. e. for all generic oracles G there exists a k, such that $x \in L(G) \Leftrightarrow \text{leafstring}_{M_k^G}(x) \in B$ for all x.

We now claim that in the previous sentence, the two quantifiers can be swapped.

Claim: There exists a balanced NPOTM $\hat{M}$ such that for all generic oracles G and for all x, $x \in L(G) \Leftrightarrow \text{leafstring}_{\hat{M}^G}(x) \in B$.

We use this machine $\hat{M}$ in the definition of a $\leq_m^{\text{plt}}$-reduction from A to B:

Input: a word w, $|w| = x$, $w = w(0) \dots w(x-1)$.
Output: a word v with $w \in A \Leftrightarrow v \in B$.
Algorithm: Let $s =_{\text{def}} 0^{\delta(|x|)} w 0^m$ be the prefix of a generic oracle G, where m is large enough to ensure that during a simulation of $\hat{M}^G(x)$, all queries to G are positioned in s. Then we have $x \in L(G) \Leftrightarrow w \in A$.
We can now construct an NPOTM M' with input x and oracle w, which simulates the behavior of $\hat{M}$ (this is possible since we have constructed G such that knowledge about w suffices to simulate G). This machine is also balanced and $\text{leafstring}_{\hat{M}^G}(x) = \text{leafstring}_{M'^w}(x)$. But Lemma 1 showed that $v =_{\text{def}} \text{leafstring}_{M'^w}(x)$ is then polylog bit computable. And the claim about $\hat{M}$ yields $x \in L(G) \Leftrightarrow v \in B$.
Combining both equivalences we get $w \in A \Leftrightarrow v \in B$.

It remains to prove the claim.

We make the the following assumption which is the inverse of our claim and lead it to a contradiction:

Assumption Z: For all k there exists an x and a generic oracle G such that

$$x \in L(G) \Leftrightarrow \text{leafstring}_{M_k^G}(x) \notin B.$$

To contradict Z we show that the family $\mathcal{C} = \{C_0, C_1, \ldots, C_{2m}, C_{2m+1}, \ldots\}$ with

- $C_{2m} =_{\text{def}} \left\{ z \mid \exists x : x \in L(z) \Leftrightarrow \text{leafstring}_{M_m^z}(x) \notin B \right\}$
- $C_{2m+1} =_{\text{def}} D_m$ (the m-th dense arithmetic set)

is a family of dense sets. For every $\mathcal{C}$-generic oracle G it then holds that

- G is generic (with A and B also the C_{2m} are arithmetic),
- for all m there exists an x with $x \in L(G) \Leftrightarrow \text{leafstring}_{M_m^G}(x) \notin B$.

This contradicts the fact that for all generic oracles G there exists a k, such that $x \in L(G) \Leftrightarrow \text{leafstring}_{M_k^G}(x) \in B$ for all x, which we have shown at the beginning of the proof. So assumption Z must be wrong.

It remains to prove that—under assumption Z—the C_{2m} are dense, i.e. for all oracles s there exists a $z \in C_{2m}$ with $s \sqsubseteq z$.

Define $\hat{s}$ such that $s \sqsubseteq \hat{s}$ and $|\hat{s}| = \delta(r)$ for a suitable r. We take a look at the following

Assumption Y: For all oracles $H \in \hat{s} \cdot \{0,1\}^\omega$ and for all x,

$$x \in L(H) \Leftrightarrow \text{leafstring}_{M_m^H}(x) \in B.$$

There are two possibilities:

1) Y is false:
This means there is an $H \in \hat{s} \cdot \{0,1\}^\omega$ and an x with $x \in L(H) \Leftrightarrow \text{leafstring}_{M_m^H}(x) \notin B$. Let x be the lexicographically smallest word, for which such an H exists. Let q_x be the lexicographically largest oracle question, that gets asked to H during the computation of $M_m^H(x)$. We define $z =_{\text{def}} H(0)H(1)\ldots H(q_x)$ which yields $\hat{s} \sqsubseteq z$ and $z \in C_{2m}$.
2) Y is true:
We define an NPOTM M' the following way: Let O be a generic oracle and $x \in \mathbb{N}$. Then we define

$$\text{leafstring}_{M'^O}(x) = \begin{cases} \text{leafstring}_{M_m^{\hat{O}}}(x) & \text{if } |\hat{s}| \leq \delta(x), \\ \text{a word from } B & \text{if } |\hat{s}| > \delta(x) \text{ and} \\ & \quad O(\delta(x)+1)\ldots O(\delta(x)+x) \in A \\ \text{a word from } \overline{B} & \text{if } |\hat{s}| > \delta(x) \text{ and} \\ & \quad O(\delta(x)+1)\ldots O(\delta(x)+x) \notin A \end{cases}$$

where $\hat{O} =_{\text{def}} \hat{s} \cdot O(\delta(r)+1)O(\delta(r)+2)\cdots$. Then M' is an NPOTM and for every generic oracle O we have $x \in L(O) \iff \text{leafstring}_{M'^O}(x) \in B$, which violates assumption Z. So, Y cannot be true unless Z is false—which finishes our proof.

□

4 Consequences

Universal Oracles

The following result is well-known from recursion theory, cf. [BI87]:

Proposition 4. *Let Π be any property of oracles, describable with a first-order formula with recursive predicates and a predicate $O(x)$ (denoting membership in the oracle), that does not depend on any finite number of bits in the oracle. Then either all generic oracles have property Π or none has.*

This immediately leads to the following consequence of Theorem 3:

Corollary 5. *Let A and B be arithmetic sets. Then*

$$\mathrm{BLeaf}^G(A) \not\subseteq \mathrm{BLeaf}^G(B) \text{ for all generic oracles } G \iff A \not\leq_m^{\mathrm{plt}} B.$$

Hence we conclude that any oracle separation between classes definable via arithmetic leaf languages holds relative to every fixed generic oracle.

Corollary 6. *Let A and B be arithmetic sets, and let $\widehat{G}$ be a fixed generic oracle. Then*

$$\begin{aligned} &\mathrm{BLeaf}^O(A) \not\subseteq \mathrm{BLeaf}^O(B) \text{ for some oracle } O \\ &\qquad \iff \mathrm{BLeaf}^{\widehat{G}}(A) \not\subseteq \mathrm{BLeaf}^{\widehat{G}}(B) \\ &\qquad \iff \mathrm{BLeaf}^G(A) \not\subseteq \mathrm{BLeaf}^G(B) \text{ for all generic oracles } G. \end{aligned}$$

Proof. Immediate from Theorem 2 and Corollary 5. □

In particular, the previous result means that all relativized separations that can be achieved among classes $\mathrm{BLeaf}(A)$ for arithmetic A hold *simultaneously*.

Corollary 7. *All oracle separations among classes, that relativizably have an arithmetic complete set, simultaneously hold relative to the same fixed generic oracle $\widehat{G}$.*

Proof. The fact that a class $\mathcal{C}$ can be characterized as $\mathrm{BLeaf}(A, R)$ with $R = \overline{A}$, i.e, $\mathcal{C} = \mathrm{BLeaf}(A)$, is equivalent to the claim that $\mathcal{C}$ has a complete set in all relativized worlds [BCS92, Theorem 4.2]. If a class $\mathcal{C}$ has an arithmetic complete set, then it is clearly characterizable with an arithmetic leaf language.

Hence, every class that relativizably has an arithmetic complete set, is a class of the form $\mathrm{BLeaf}(A)$ for arithmetic A. The result now follows immediately from Corollary 6. □

In other words, generic oracles induce an isomorphism between (the order of) relativized complexity classes and (the order of) plt-degrees. As is usual, we define a plt-degree to be the class of sets all equivalent with respect to $\leq_m^{\mathrm{plt}}$ (observe that $\leq_m^{\mathrm{plt}}$ is reflexive and transitive). A plt-degree is said to be *arithmetic* iff it has an arithmetic representative.

Corollary 8. *Let $\widehat{G}$ be any fixed generic oracle. Then the family*

$$\{\, \mathrm{BLeaf}^{\widehat{G}}(A) \mid A \text{ is arithmetic} \,\},$$

partially ordered with respect to set inclusion, is isomorphic to the partial order of all arithmetic plt-degrees.

As mentioned in the introduction, Blum and Impagliazzo proved that if two resource-bounded classes coincide relative to a generic oracle, then they coincide absolutely [BI87]. We can rephrase Corollary 5, extending their result to all leaf-language definable classes.

Corollary 9. *Let A and B be arithmetic sets. Then*

$$\begin{aligned} &\mathrm{BLeaf}^G(A) \subseteq \mathrm{BLeaf}^G(B) \text{ for some generic oracle } G \\ &\qquad \Longrightarrow \mathrm{BLeaf}(A) \subseteq \mathrm{BLeaf}(B). \end{aligned}$$

Resource-bounded Genericity

Almost all interesting syntactic complexity classes between P and PSPACE can be characterized using regular leaf languages (the prominent exception to this is the class PP, see, e.g., [Vol99]). For this case, the relevant class of dense sets of finite oracles (i.e., the sets C_{2m} in the proof of Theorem 3) are easily seen to be decidable in $\mathrm{QP} =_{\mathrm{def}} \mathrm{DTIME}\big(n^{\log^{O(1)} n}\big)$, i.e., quasipolynomial time. Thus, the obtained oracle is not only $\mathcal{A}$-generic, but even QP-generic. Hence, the following corollary is proven:

Corollary 10. *Let A and B be regular sets. Then*

$$\begin{aligned}&\mathrm{BLeaf}^O(A) \not\subseteq \mathrm{BLeaf}^O(B) \text{ for some oracle } O\\ &\qquad\iff \mathrm{BLeaf}^G(A) \not\subseteq \mathrm{BLeaf}^G(B) \text{ for some QP-generic oracle } G.\end{aligned}$$

Moreover, when separating a complexity class from P, even a P-generic oracle in the above sense is enough.

Corollary 11. *Let A be a regular set. Then*

$$\begin{aligned}&\mathrm{BLeaf}^O(A) \not\subseteq \mathrm{P}^O \text{ for some oracle } O\\ &\qquad\iff \mathrm{BLeaf}^G(A) \not\subseteq \mathrm{P}^G \text{ for some P-generic oracle } G.\end{aligned}$$

Type-2 Complexity

In [CIY97], Cook, Impagliazzo, and Yamakami showed that any two complexity classes satisfying some general conditions are distinct relative to a generic oracle if and only if the corresponding type-2 classes are distinct. As we will see, this provides a bridge between leaf language theory and type-2 complexity theory.

A *type-2 function* F over Σ^* assigns to all words $x \in \Sigma^*$ and sets $X \subseteq \Sigma^*$ a value in Σ^*. A *type-2 relation* is a 0-1-valued type-2 function. Let R be any type-2 relation and let $\mathfrak{C}$ be any class of type-2 relations. For a fixed set X, we define $R[X] =_{\mathrm{def}} \{\, x \mid R(x, X) \,\}$ and $\mathfrak{C}[X] =_{\mathrm{def}} \{\, R[X] \mid R \in \mathfrak{C} \,\}$.

We say that a type-2 function F is polynomial-time computable if there exist a deterministic oracle Turing machine M and polynomial p such that, given input string x on the input tape and set X as an oracle, M outputs $F(x, X)$ after at most $p(|x|)$ steps, where each oracle query counts as only one step.

A type-2 relation R is polynomial-time many-one reducible to a type-2 relation S, in symbols $R \leq_m^{\mathrm{p}} S$, iff there exist a type-2 polynomial-time computable function F and type-2 polynomial-time computable relation Q such that for all words x and sets X, it holds that $R(x, X) = S(F(x, X), Q[x, X])$, where $Q[x, X] =_{\mathrm{def}} \{z \mid Q(x, z, X)\}$.

The question studied by Cook, Impagliazzo, and Yamakami is what inclusion relations hold among type-2 classes. For classes closed under $\leq_m^{\mathrm{p}}$, this was related to relativized inclusion under a generic oracle [CIY97, Theorem 3.2]:

Proposition 12. *Let $\mathfrak{C}$ and $\mathfrak{D}$ be classes of computable type-2 relations and suppose that $\mathfrak{C}$ and $\mathfrak{D}$ are closed under $\leq_m^{\mathrm{p}}$. Then for any generic oracle G,*

$$\mathfrak{C} \subseteq \mathfrak{D} \iff \mathfrak{C}[G] \subseteq \mathfrak{D}[G].$$

In the leaf language framework, type-2 complexity classes can be defined in the following way. Given two sets $A, R \subseteq \Sigma^*$ with $A \cap R = \emptyset$, the type-2 relation S belongs to the type-2 class $\mathbf{BLeaf}(A,R)$ if and only if there exists a polynomial-time oracle Turing machine M such that for all words x and sets X, $S(x,X) \Longrightarrow \text{leafstring}_{M^X}(x) \in A$ and $\neg S(x,X) \Longrightarrow \text{leafstring}_{M^X}(x) \in R$.

Since it can be observed that each class $\mathbf{BLeaf}(A,R)$ is closed under $\leq_m^{\text{p}}$, we obtain from Theorem 3 and Theorem 12 a complete combinatorial characterization of inclusion relations between reasonably definable syntactic type-2 classes.

Corollary 13. *Let A and B be arithmetic sets. Then the following statements are equivalent.*

(1) $A \leq_m^{\text{plt}} B$.
(2) $\text{BLeaf}^G(A) \subseteq \text{BLeaf}^G(B)$ *for some generic oracle G.*
(3) $\mathbf{BLeaf}(A)[G] \subseteq \mathbf{BLeaf}(B)[G]$ *for some generic oracle G.*
(4) $\mathbf{BLeaf}(A) \subseteq \mathbf{BLeaf}(B)$.

Proof. The equivalence of (1) and (2) is just Theorem 3, and the equivalence of (3) and (4) is just Proposition 12. Equivalence of (2) and (3) follows from Theorem 15 given below. □

Typically, a type-1 complexity class $\mathcal{C}$ has a natural type-2 counterpart $\mathbf{C}$, based on the same resources used to define $\mathcal{C}$. Cook, Impagliazzo, and Yamakami [CIY97] raised the issue of determining for which classes $\mathcal{C}$ and $\mathbf{C}$, their relativized versions coincide, i.e., $\mathcal{C}^O = \mathbf{C}[O]$ for all oracles O. Consider for example the class BPP of all languages acceptable in polynomial-time by probabilistic Turing machines with two-sided bounded-error probability. Not every probabilistic machine has this latter property, hence BPP is a promise class. The relativized class BPP^O contains all sets acceptable relative to oracle O by some probabilistic Turing machine that relative to oracle O has bounded-error. On the other hand, for the type-2 class **BPP**, we require that our machines have bounded-error for all inputs and for *all* oracles. Hence $\mathbf{BPP}[O]$ consists of all sets acceptable relative to oracle O by some probabilistic Turing machine that relative to all oracles has bounded-error. Thus, $\mathbf{BPP}[O] \subseteq \text{BPP}^O$ but the converse is not clear, and indeed, there is an oracle O for which $\mathbf{BPP}[O] \neq \text{BPP}^O$ [CIY97]. But for generic oracles, the situation changes. Cook *et al.* proved that $\mathbf{BPP}[G] = \text{BPP}^G$ for any generic oracle G. A number of other classes behaving like BPP were exhibited in [CIY97]. A complete characterization of those classes $\mathcal{C}$ with type-2 counterpart $\mathbf{C}$, that have the property that $\mathcal{C}^G = \mathbf{C}[G]$ for generic G, was left open.

Using the method of forcing we can give an almost exhaustive answer to this question for the classes definable by pairs of leaf languages (A,R). The sufficient condition we obtain in Theorem 15 below, that $A \cup R$ is co-recursively enumerable, is very general and is satisfied by all leaf-language definable complexity classes we know of. In particular of course, complementary pairs of leaf languages ($A = \overline{R}$) satisfy this condition. Thus, for any A, $\text{BLeaf}^G(A) = \mathbf{BLeaf}(A)[G]$, proving equivalence of (2) and (3) in Corollary 13.

An *oracle property* is a type-2 relation $S(x,X)$ that does not depend on words x but only on sets X. Therefore, we denote oracle properties simply by $S(X)$. We say that a finite oracle v *forces an oracle property* S if $S(X)$ holds for every oracle X extending v. The following lemma is well-known – a formal proof can be found in [CIY97, Lemma 2.1].

Lemma 14. *Suppose S is a Π_1^0 oracle property and G is a generic oracle. Then $S(G)$ holds if and only if some finite prefix of G forces S.*

Theorem 15. *Let (A,R) be a pair of leaf languages such that $A \cup R$ is co-r.e. Then* $\mathrm{BLeaf}^G(A,R) = \mathbf{BLeaf}(A,R)[G]$ *for any generic oracle G.*

Proof. It suffices to show the inclusion $\subseteq$. Our proof follows the proof of Proposition 4.2 in [CIY97]. Let $L \in \mathrm{BLeaf}^G(A,R)$, i.e., there exists a balanced NPOTM M such that for all x, $\mathrm{leafstring}_{M^G}(x) \in A \cup R$, and $x \in L \Leftrightarrow \mathrm{leafstring}_{M^G}(x) \in A$. We define the oracle property Q by

$$Q(X) \Longleftrightarrow_{\mathrm{def}} \text{for all } x,\ \mathrm{leafstring}_{M^X}(x) \in A \cup R.$$

Since $A \cup R$ is co-r.e., Q is clearly a Π_1^0 oracle-property for which $Q(G)$ is true. Hence, by Lemma 14, there exists a finite oracle v which is a prefix of G such that for all oracles O extending v, $Q(O)$ holds. Now define a type-2 relation S' by

$$S'(x,X) \Longleftrightarrow_{\mathrm{def}} \mathrm{leafstring}_{M^{X^v}}(x) \in A,$$

where X^v denotes the oracle X with a prefix of length $|v|$ replaced with v. Clearly, $S' \in \mathbf{BLeaf}(A,R)$ and, since $G^v = G$, we obtain $S'[G] = L$. Thus, $L \in \mathbf{BLeaf}(A,R)[G]$. $\square$

5 Conclusion

In the present paper, we related leaf language separations to separations by generic oracles and obtained a number of consequences concerning, e.g., simultaneous separations and type-2 complexity. It should be remarked that our main result can also be proved for complexity classes of functions. "Leaf languages" for the definition of function classes and oracle separations were studied in [KSV00]. The main result from that paper can be extended in the same way as our paper extends the Bovet-Crescenzi-Silvestri-Vereshchagin result, i.e., two function classes definable in a certain way in the leaf-language framework are separable by any oracle iff they are separable by a generic oracle iff they are separated by any generic oracle. Among the possible applications of this result we only mention that relative to any generic oracle, the Embedding Conjecture from [KW00] about the structure of the Boolean hierarchy of NP-partitions holds.

The in our point of view most important open question brings us back to Sect. 1, where we mentioned that Blum and Impagliazzo related the existence of separating oracles to unrelativized separations, most importantly in the case of the classes P and NP. They proved: If $\mathrm{P}^G \neq \mathrm{UP}^G$ relative to some generic oracle G, then $\mathrm{P} \neq \mathrm{NP}$. Because it is known that P and UP can be separated by *some* oracle, one might hope that, by Corollary 6, there exists a generic separating oracle. For this, it would be necessary that UP could be defined by a leaf language. This would lead to a result like "if A is an arithmetic language such that for all oracles O, $\mathrm{BLeaf}^O(A) = \mathrm{UP}^O$, then $\mathrm{P} \neq \mathrm{NP}$." Unfortunately, it is known that the prerequisite of this statement does not hold: There is no A such that relativizably, $\mathrm{UP} = \mathrm{BLeaf}(A)$ [BCS92]. To characterize UP relativizably, we need a leaf language pair (A,R) with $R \neq \overline{A}$, and in this case, our main result (Theorem 3) does not apply.

The main obstacle in the proof of Theorem 3, when dealing with pairs (A,R), $R \neq \overline{A}$, is the denseness condition for the sets C_{2m}, which we could only prove for $R = \overline{A}$. But maybe there are certain restrictions to general pairs (A,R) such that then the C_{2m} are still dense, and maybe even the pair for UP fulfills these restrictions. The main open issue thus is to find sufficient conditions that leaf language pairs have to satisfy such that Theorem 3 still holds. Interesting in this context is a result by Fortnow and Yamakami [FY96] who proved a separation of $\mathrm{UP}^{\mathrm{NP}}$ and P^{NP} relative to a generic oracle; but again, $\mathrm{UP}^{\mathrm{NP}}$ does not seem to have an obvious characterization by a complementary pair of leaf languages.

References

[AS96] K. Ambos-Spies. Resource-bounded genericity. In S. B. Cooper, T. A. Slamand, and S. S. Wainer, editors, *Computability, Enumerability, Unsolvability. Directions in Recursion Theory*, volume 224 of *London Mathematical Society Lecture Notes*, pages 1–59. Cambride University Press, 1996.

[BCS92] D. P. Bovet, P. Crescenzi, and R. Silvestri. A uniform approach to define complexity classes. *Theoretical Computer Science*, 104:263–283, 1992.

[BG81] C. Bennett and J. Gill. Relative to a random oracle $\mathrm{P}^A \neq \mathrm{NP}^A \neq \mathrm{coNP}^A$ with probability 1. *SIAM Journal on Computing*, 10:96–113, 1981.

[BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the P=NP problem. *SIAM Journal on Computing*, 4:431–442, 1975.

[BI87] M. Blum and R. Impagliazzo. Generic oracles and oracle classes. In *Proceedings 28th IEEE Symposium on Foundations of Computer Science*, pages 118–126. IEEE Computer Society Press, 1987.

[CCG$^+$94] R. Chang, B. Chor, O. Goldreich, J. Hartmanis, J. Hastad, D. Ranjan, and P. Rohatgi. The random oracle hypothesis is false. *Journal of Computer and System Sciences*, 49:24–39, 1994.

[CIY97] S. A. Cook, R. Impagliazzo, and T. Yamakami. A tight relationship between generic oracles and type-2 complexity theory. *Information and Computation*, 137(2):159–170, 1997.

[Con73] R. L. Constable. Type two computable functionals. In *Proceedings 5th Symposium on Theory of Computing*, pages 108–121. ACM-Press, 1973.

[FFKL93] S. Fenner, L. Fortnow, S. Kurtz, and L. Li. An oracle builder's toolkit. In *Proceedings 8th Structure in Complexity Theory*, pages 120–131. IEEE Computer Society Press, 1993.

[Fos93] J. A. Foster. The generic oracle hypothesis is false. *Information Processing Letters*, 45:59–62, 1993.

[FY96] L. Fortnow and T. Yamakami. Generic separations. *Journal of Computer and Systems Sciences*, 52(1):191–197, 1996.

[HH90] J. Hartmanis and L. Hemachandra. Robust machines accept easy sets. *Theoretical Computer Science*, 74(2):217–226, 1990.

[KSV00] S. Kosub, H. Schmitz, and H. Vollmer. Uniform characterizations of complexity classes of functions. *International Journal of Foundations of Computer Science*, 11(4):525–551, 2000.

[Kur83] S. Kurtz. Notions of weak genericity. *Journal of Symbolic Logic*, 48(3):724–743, 1983.

[KW00] S. Kosub and K. W. Wagner. The boolean hierarchy of NP-partitions. In *Proceedings 17th Symposium on Theoretical Aspects of Computer Science*, volume 1770 of *Lecture Notes in Computer Science*, pages 157–168. Springer-Verlag, 2000.

[Meh76] K. Mehlhorn. Polynomial and abstract subrecursive classes. *Journal of Computer and System Sciences*, 12:147–178, 1976.

[Pap94] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, Reading, MA, 1994.

[Soa87] R. I. Soare. *Recursively Enumerable Sets and Degrees – A Study of Computable Functions and Computably Generated Sets.* Perspectives in Mathematical Logic. Springer Verlag, Berlin Heidelberg, 1987.

[Ver93] N. K. Vereshchagin. Relativizable and non-relativizable theorems in the polynomial theory of algorithms. *Izvestija Rossijskoj Akademii Nauk*, 57:51–90, 1993. In Russian.

[Vol99] H. Vollmer. Uniform characterizations of complexity classes. *Complexity Theory Column 23, ACM-SIGACT News*, 30(1):17–27, 1999.

[VW97] H. Vollmer and K. W. Wagner. Measure one results in computational complexity theory. In D.-Z. Du and K.-I. Ko, editors, *Advances in Algorithms, Languages, and Complexity.* Kluwer Academic Publishers, 1997.