Electronic Colloquium on Computational Complexity, Report No. 147 (2005)

Machines that can Output Empty Words

Christian Glaßer and Stephen Travers*

Theoretische Informatik
Julius-Maximilians Universität Würzburg
Am Hubland,
97074 Würzburg, Germany

Abstract

We propose the e-model for leaf languages which generalizes the known balanced and unbalanced concepts. Inspired by the neutral behavior of rejecting paths of NP machines, we allow transducers to output empty words.

The paper explains several advantages of the new model. A central aspect is that it allows us to prove strong gap theorems: For any class $\mathcal{C}$ that is definable in the e-model, either $\mathrm{coUP} \subseteq \mathcal{C}$ or $\mathcal{C} \subseteq \mathrm{NP}$. For the existing models, gap theorems, where they exist at all, only identify gaps for the definability by *regular* languages. We prove gaps for the general case, i.e., for the definability by *arbitrary* languages. We obtain such general gaps for NP, coNP, $1\mathrm{NP}$, and $\mathrm{co1NP}$. For the regular case we prove further gap theorems for Σ_2^{P}, Π_2^{P}, and Δ_2^{P}. These are the first gap theorems for Δ_2^{P}.

This work is related to former work by Bovet, Crescenzi, and Silvestri, Vereshchagin, Hertrampf et al., Burtschick and Vollmer, and Borchert et al.

1 Introduction

Bovet, Crescenzi, and Silvestri [BCS92] and Vereshchagin [Ver93] independently introduced leaf languages. This concept allows a uniform definition of many interesting complexity classes like NP and PSPACE. The advantage of such an approach is obvious: It allows to prove quite general theorems in a concise way. For example, Glaßer et al. [GOP$^+$05] recently showed that if $\mathcal{C}$ is a class that is balanced-leaf-language definable by a regular language, then all many-one complete problems of $\mathcal{C}$ are polynomial-time many-one autoreducible. This general theorem answered several open questions, since classes like NP, PSPACE, and the levels of the PH are definable in this way.

Moreover, leaf languages allow concise oracle constructions. The background is the BCSV-theorem [BCS92, Ver93] that connects polylog-time reducibility (plt-reducibility) with the robust inclusion of two complexity classes (i.e., the inclusion with respect to all oracles). This connection reduces oracle constructions to their combinatorial core. In particular, neither do we have to care about the detailed stagewise construction of the oracle, nor do we have to describe the particular coding of the single

*Emails: {glasser, travers}@informatik.uni-wuerzburg.de

ISSN 1433-8092

stages. As an example, Lemma 5.6 below presents a short proof for the existence of an oracle relative to which $\mathrm{UP} \vee \mathrm{UP} \not\subseteq 1\mathrm{NP}$. A direct oracle construction would be substantially longer.

In this paper we offer a useful generalization of the known leaf-language concepts. Despite of its broader definition, the new concept is convenient and has the nice features we appreciate with traditional leaf languages. It even combines certain advantages of single known concepts. We summarize the benefit of the new notion:

1. contains the traditional concepts
2. works with balanced computation trees
3. admits a BCSV-theorem [BCS92, Ver93]
4. establishes a tight connection between the polynomial-time hierachy and the Straubing-Thérien hierarchy (the quantifier-alternation hierarchy of the logic FO$[<]$ on words)

The new e-model of leaf languages is inspired by the observation that rejecting paths of nondeterministic computations act as *neutral* elements. In this sense we allow nondeterministic transducers not only to output single letters, but also to output the empty word ε which is the neutral element of Σ^*. More precisely, we consider nondeterministic polynomial-time-bounded Turing machines M such that on every input, every computation path stops and outputs an element from $\Sigma \cup \{\varepsilon\}$. Let $M(x)$ denote the computation tree on input x, and define $\beta_M(x)$ as the concatenation of all outputs of $M(x)$. For any language B, let $\mathrm{Leaf}_\varepsilon^\mathrm{p}(B)$ (the e-class of B) be the class of languages L such that there exists a nondeterministic polynomial-time-bounded Turing machine M as above such that for all x,

$$x \in L \iff \beta_M(x) \in B.$$

If we demand that M never outputs ε, then this defines $\mathrm{Leaf}_\mathrm{u}^\mathrm{p}(B)$ (the u-class of B). If we demand that M is balanced and never outputs ε, then this defines $\mathrm{Leaf}_\mathrm{b}^\mathrm{p}(B)$ (the b-class of B). (M is *balanced* if there exists a polynomial-time computable function that on input (x, n) computes the n-th path of $M(x)$.) The notions e-class, u-class, and b-class are extended from a single language B to a class of languages $\mathcal{C}$ in the standard way: $\mathrm{Leaf}_\varepsilon^\mathrm{p}(\mathcal{C})$ (the e-class of $\mathcal{C}$) is the union of all $\mathrm{Leaf}_\varepsilon^\mathrm{p}(B)$ where $B \in \mathcal{C}$. For a survey on the leaf-language approach we refer to Wagner [Wag04].

It is immediately clear that the u-model and the b-model are restrictions of the e-model.

$$\mathrm{Leaf}_\mathrm{b}^\mathrm{p}(B) \subseteq \mathrm{Leaf}_\mathrm{u}^\mathrm{p}(B) \subseteq \mathrm{Leaf}_\varepsilon^\mathrm{p}(B)$$

Moreover, it is intuitively clear that the presence of the neutral element ε gives the class $\mathrm{Leaf}_\varepsilon^\mathrm{p}(B)$ some inherent nondeterministic power which makes $\mathrm{Leaf}_\varepsilon^\mathrm{p}(B)$ seemingly bigger than P. We will discuss this issue and we will identify $\mathrm{UP} \cap \mathrm{coUP}$ as a lower bound (we obtain stronger bounds if we restrict to regular languages B). The advantage of the e-model over the u-model is its simplicity: In the e-model we can assume balanced computation trees which in turn leads to easy plt-reductions. The advantage over the b-model is the established tight connection between the polynomial-time hierarchy and the Straubing-Thérien hierarchy, a well-studied hierarchy of regular languages. Glaßer [Gla05] shows that such a connection does not hold for the b-model. This connection within the e-model makes it possible to exactly characterize leaf-language classes in the environment of NP.

In order to describe our results we have to define the levels of the Straubing-Thérien hierarchy (STH). In the scope of this paper it suffices to summarize that the STH is a hierarchy of levels that contain regular languages. We use a notation that already suggests a connection to the polynomial-time hierarchy (PH).

A language belongs to level Σ_k^{FO} if it can be defined by a sentence of the logic FO[<] on words such that the sentence starts with an existential quantifier and has at most $k-1$ quantifier alternations. Π_k^{FO} denotes the level of the complements of elements in Σ_k^{FO}. Δ_{k+1}^{FO} denotes the intersection of Σ_k^{FO} and Π_k^{FO}. The formal definition can be found in the preliminaries.

Results: We start with observations that let us easily transfer the known BCSV-theorem to the new notion. Along these lines we show that the polynomial-time hierarchy (PH) is connected with the Straubing-Thérien hierarchy in the following sense: The e-class of level Σ_k^{FO} of the STH equals level Σ_k^{P} of the PH. Note that this leaves room for the possibility that languages outside Σ_k^{FO} form e-classes that are still contained in Σ_k^{P}. So even the e-class of a superset of Σ_k^{FO} might be equal to Σ_k^{P}. For the lower levels, however, we are able to rule out this possibility. This proves a substantially *tighter connection* between both hierarchies. For instance, under the reasonable assumption $\mathrm{coUP} \not\subseteq \mathrm{NP}$, we show that the languages in Σ_1^{FO} are the only languages whose e-classes are contained in NP. Hence, under this assumption, a language belongs to Σ_1^{FO} if and only if its e-class is contained in NP. This connects Σ_1^{FO} and NP in the strongest possible way. We obtain several other strong relationships of this type, they are summarized in Table 1. In particular, we prove the first gap theorem for Δ_2^{P} (Corollary 4.9). This is possible by the e-model's tight connection to the STH, by the forbidden-pattern characterization of Σ_2^{FO} which was proved by Pin and Weil [PW97], and by the equality $\mathrm{Leaf}_u^p(\Sigma_2^{FO}) = \Delta_2^{P}$ which was showen by Borchert, Schmitz, and Stephan [BSS99] and Borchert et al. [BLS+04].

Some comments about the results in Table 1 are appropriate. First, they can be interpreted as gap theorems for leaf-language definability. For instance, the row about Σ_1^{FO} tells us that any e-class either is contained in NP or contains at least coUP. Hence, once an e-class becomes bigger than NP, its complexity jumps to at least $\mathrm{NP} \cup \mathrm{coUP}$. Second, there exist several evidences that classes in the columns 3–5 are not contained in the corresponding class of column 2. In any case there exist oracles relative to which this non-containment holds. Third, all classes in the first column are decidable, i.e., on input of a finite automaton A we can decide whether the language accepted by A belongs to the class. This allows a decidable and precise classification of e-classes under the assumption that the classes in the 4th column are not contained in the respective class in the 2nd column. On input of a regular language B (via its finite automaton) we can determine whether or not B's e-class is contained in the classes of the 2nd column.

With U we identify the class of all languages whose e-class is (robustly) contained in $1\mathrm{NP}$. A language belongs to U if and only if membership of a word can be expressed in terms of a unique occurrence of a substring and in terms of forbidden substrings. This shows that U is a class of regular languages. We prove a decidable characterization of U, a so-called forbidden-pattern characterization. It exactly reveals the structure in a finite automaton that is responsible for shifting a language outside U.

Gap theorems for leaf-language definability are rather rare. With the following theorem we summarize the known results.

Theorem 1.1 *Let B be a nontrivial regular language.*

1. *[Bor95] The u-class of B either is contained in P, or contains at least one of the following classes: NP, coNP, $\mathrm{MOD}_p\mathrm{P}$ for some prime p.*
2. *[BKS99] The u-class of B either is contained in NP, or contains at least one of the following classes: coNP, $\mathrm{co1NP}$, $\mathrm{MOD}_p\mathrm{P}$ for some prime p.*

[1]Some remarks about notations: $\mathcal{C} \vee \mathcal{D}$ (resp., $\mathcal{C} \dot{\vee} \mathcal{D}$) is the class of unions (resp., disjoint unions) of some $L_1 \in \mathcal{C}$ and some $L_2 \in \mathcal{D}$. From this, the operators $\wedge$ and $\dot{\wedge}$ are derived via DeMorgan's law. $\mathrm{AU}\Sigma_2^{P}$ and $\mathrm{AU}\Pi_2^{P}$ denote levels of the unambiguous polynomial-time hierarchy. More details can be found in the preliminaries section.

$\mathcal{C}$	$\mathrm{Leaf}^{\mathrm{P}}_{\varepsilon}(\mathcal{C}) =$	if $B \notin \mathcal{C}$ then $\mathrm{Leaf}^{\mathrm{P}}_{\varepsilon}(L)$ contains	if $B \in \mathrm{REG} - \mathcal{C}$ then $\mathrm{Leaf}^{\mathrm{P}}_{\varepsilon}(L)$ contains	if $B \in \mathrm{SF} - \mathcal{C}$ then $\mathrm{Leaf}^{\mathrm{P}}_{\varepsilon}(L)$ contains
$\emptyset$	$\emptyset$	UP or coUP	NP, coNP, or $\mathrm{MOD}_p\mathrm{P}$ for a prime p	NP or coNP
Σ^{FO}_1	NP	coUP	coNP, $\mathrm{co1NP}$, or $\mathrm{MOD}_p\mathrm{P}$ for a prime p	coNP or $\mathrm{co1NP}$
Π^{FO}_1	coNP	UP	NP, $\mathrm{1NP}$, or $\mathrm{MOD}_p\mathrm{P}$ for a prime p	NP or $\mathrm{1NP}$
U	$\mathrm{1NP}$	$\mathrm{UP} \vee \mathrm{UP}$ or $\mathrm{UP} \dot{\vee} \mathrm{coUP}$	$\mathrm{UP} \vee \mathrm{UP}$ or $\mathrm{UP} \dot{\vee} \mathrm{coUP}$	$\mathrm{UP} \vee \mathrm{UP}$ or $\mathrm{UP} \dot{\vee} \mathrm{coUP}$
coU	$\mathrm{co1NP}$	$\mathrm{coUP} \wedge \mathrm{coUP}$ or $\mathrm{UP} \underset{\cdot}{\wedge} \mathrm{coUP}$	$\mathrm{coUP} \wedge \mathrm{coUP}$ or $\mathrm{UP} \underset{\cdot}{\wedge} \mathrm{coUP}$	$\mathrm{coUP} \wedge \mathrm{coUP}$ or $\mathrm{UP} \underset{\cdot}{\wedge} \mathrm{coUP}$
Δ^{FO}_2	Δ^{P}_2	–	$\mathrm{AU}\Sigma^{\mathrm{P}}_2$ or $\mathrm{AU}\Pi^{\mathrm{P}}_2$	$\mathrm{AU}\Sigma^{\mathrm{P}}_2$ or $\mathrm{AU}\Pi^{\mathrm{P}}_2$
Σ^{FO}_2	Σ^{P}_2	–	$\mathrm{AU}\Pi^{\mathrm{P}}_2$	$\mathrm{AU}\Pi^{\mathrm{P}}_2$
Π^{FO}_2	Π^{P}_2	–	$\mathrm{AU}\Sigma^{\mathrm{P}}_2$	$\mathrm{AU}\Sigma^{\mathrm{P}}_2$

Table 1: Summary of the obtained gap theorems where B is a language different from $\emptyset$ and Σ^*.[1]

3. *[Sch01] The u-class of B either is contained in Σ^{P}_2, or contains $\mathrm{AU}\Pi^{\mathrm{P}}_2$.*
4. *[Gla05] The b-class of B either is contained in P, or contains at least one of the following classes: NP, coNP, $\mathrm{MOD}_p\mathrm{P}$ for some prime p.*
5. *[Gla05] The b-class of B either is contained in NP, or contains at least one of the following classes: coNP, $\mathrm{co1NP}$, $\mathrm{MOD}_p\mathrm{P}$ for some prime p.*

2 Preliminaries

2.1 Basic Notions

We denote with NL, P, NP, coNP and PSPACE the standard complexity classes whose definitions can be found in any textbook on computational complexity (cf. [Pap94], for example). The class UP is the class of decision problems solvable by an NP machine such that if the input belongs to the language, exactly one computation path accepts and if the input does not belong to the language, all computation paths reject. Contrary, the class $\mathrm{1NP}$ is the class of decision problems solvable by an NP machine such that the input belongs to the language if and only if exactly one computation path accepts.[2] For any $k > 1$, $\mathrm{MOD}_k\mathrm{P}$ is the class of decision problems solvable by an NP machine such that the number of accepting paths is divisible by k if and only if the input does not belong to the language. The characteristic function of a set A is denoted as χ_A. We will always assume that our alphabet Σ contains at least 2 letters.

[2]Observe that in contrast to UP, a machine can legally have more than one accepting path.

Let $\preceq$ denote the usual subword relation, i.e. $v \preceq w$ if $v = v_1 \dots v_n$ for letters $v_1, \dots, v_n$ and $w \in \Sigma^* v_1 \Sigma^* v_2 \dots \Sigma^* v_n \Sigma^*$. We write $v \prec w$ if $v \preceq w$ and $v \neq w$. For $k \geq 0$ we write $v \preceq_k w$ if v is a nonempty word that appears precisely k-times as a subword of w. In addition we define $\varepsilon \preceq_1 w$ for every word w. For $k \geq 0$ we write $v \preceq_{\geq k} w$ if there exists $l \geq k$ such that $v \preceq_l w$. For $k \geq 0$ and a finite set B of words $v_1, \dots, v_{|B|}$ we write $B \preceq_k w$ if k can be written as $k = k_1 + \dots + k_{|B|}$ such that

$$v_1 \preceq_{k_1} w, \quad v_2 \preceq_{k_2} w, \quad \dots, \quad v_{|B|} \preceq_{k_{|B|}} w.$$

So $v \preceq w$ if and only if there exists $k \geq 1$ such that $v \preceq_k w$. Also, $v \not\preceq w$ if and only if $v \preceq_0 w$.

We call a language B nontrivial if $B \neq \emptyset$ and $B \neq \Sigma^*$. If $L, K \subseteq \Sigma^*$ are disjoint languages, we also write $(L, K) \subseteq \Sigma^*$, i.e. whenever we talk about a pair $(L, K) \subseteq \Sigma^*$ of languages, we assume that L and K are disjoint.

Definition 2.1 *Let $\mathcal{K}, \mathcal{M}$ be complexity classes. We define*

$$\begin{aligned} \mathcal{K} \vee \mathcal{M} &=_{\text{def}} \{A \cup B \mid A \in \mathcal{K}, B \in \mathcal{M}\},\ \mathcal{K} \wedge \mathcal{M} =_{\text{def}} \mathrm{co}(\mathrm{co}\mathcal{K} \vee \mathrm{co}\mathcal{M}), \\ \mathcal{K} \dot{\vee} \mathcal{M} &=_{\text{def}} \{A \cup B \mid A \in \mathcal{K}, B \in \mathcal{M}, A \cap B = \emptyset\},\ \mathcal{K} \underset{\cdot}{\wedge} \mathcal{M} =_{\text{def}} \mathrm{co}(\mathrm{co}\mathcal{K} \dot{\vee} \mathrm{co}\mathcal{M}). \end{aligned}$$

Definition 2.2 *For any language $L \subseteq \Sigma^*$ and $a \notin \Sigma$, we define $L_a \subseteq (\Sigma \cup \{a\})^*$ as*

$$L_a =_{\text{def}} \{a^{m_0} w_1 a^{m_1} w_2 a^{m_2} \dots a^{m_{n-1}} w_n a^{m_n} \mid m_0, \dots, m_n \geq 0, w_1 w_2 \dots w_n \in L\}.$$

2.2 The Unambiguous Alternation Hierarchy

Niedermeier and Rossmanith [NR98] introduced the unambiguous alternation hierarchy. For its definition we use Hemaspaandra's characterization in terms of unambiguous alternating quantifiers. For any complexity class $\mathcal{C}$, define $\exists^{\mathrm{u}} \cdot \mathcal{C}$ as the class of languages L such that there exist a polynomial p and $L' \in \mathcal{C}$ such that for all x,

$$\begin{aligned} x \in L &\Rightarrow \text{ there exists exactly one } y \in \Sigma^{=p(|x|)} \text{ such that } (x, y) \in L' \\ x \notin L &\Rightarrow \text{ there exists no } y \in \Sigma^{=p(|x|)} \text{ such that } (x, y) \in L'. \end{aligned}$$

Analogously, $\forall^{\mathrm{u}} \cdot \mathcal{C}$ is the class of languages L such that there exist a polynomial p and $L' \in \mathcal{C}$ such that for all x,

$$\begin{aligned} x \in L &\Rightarrow \text{ for all } y \in \Sigma^{=p(|x|)}, (x, y) \in L' \\ x \notin L &\Rightarrow \text{ there exists exactly one } y \in \Sigma^{=p(|x|)} \text{ such that } (x, y) \notin L'. \end{aligned}$$

Definition 2.3 (attributed to unpublished work of Hemaspaandra [NR98])

$$\begin{aligned} \mathrm{AU\Sigma_0^P} = \mathrm{AU\Pi_0^P} &=_{\text{def}} \mathrm{P} \\ \mathrm{AU\Sigma_{k+1}^P} &=_{\text{def}} \exists^{\mathrm{u}} \cdot \mathrm{AU\Pi_k^P} \quad \textit{for } k \geq 0 \\ \mathrm{AU\Pi_{k+1}^P} &=_{\text{def}} \forall^{\mathrm{u}} \cdot \mathrm{AU\Sigma_k^P} \quad \textit{for } k \geq 0. \end{aligned}$$

It is expected that level n of the unambiguous alternation hierarchy is not contained in level $n - 1$ of the polynomial-time hierarchy. Spakowski and Tripathi [ST04] construct an oracle relative to which for every $n \geq 1$, level n of the unambiguous alternation hierarchy is not contained in $\Pi_{\mathrm{n}}^{\mathrm{P}}$.

2.3 Straubing-Thérien Hierarchy

Starfree languages are regular languages that can be build from single letters by using Boolean operations and concatenation. Let SF denote the class of starfree languages. Brzozowski and Cohen [CB71, Brz76] introduced the dot-depth hierarchy which measures the complexity of starfree languages in terms of necessary alternations between Boolean operations and concatenation in the definition of the language. Straubing and Thérien [Str81, Thé81, Str85] introduced a modification that is more appropriate for the algebraic theory of languages, but still covers the important aspects of the dot-depth hierarchy. This hierarchy is called Straubing-Thérien hierarchy (STH).

Perrin and Pin [PP86] proved a logical characterization of the STH. We use this characterization as definition, since it uses an easy logic on words and it shows nice parallels to the definition of the polynomial-time hierarchy. Formulas of the first-order logic FO[$<$] consist of first-order quantifiers, Boolean operators, the binary relation symbol $<$, and unary relation symbols π_a for each letter a. A sentence ϕ is satisfied by a word w if ϕ evaluates to true where variables are interpreted as positions in w and $\pi_a x$ is interpreted as "letter a appears at position x in w". A language B is FO[$<$] definable if there exists a sentence ϕ such that for all words w, $w \in L$ if and only if ϕ is satisfied by w. A $\Sigma_{\mathrm{k}}^{\mathrm{FO}}$-sentence (resp., $\Pi_{\mathrm{k}}^{\mathrm{FO}}$-sentence) is a sentence of FO[$<$] that is in prenex normal form, that starts with an existential (resp., universal) quantifier, and that has at most $k-1$ quantifier alternations. A language belongs to the class $\Sigma_{\mathrm{k}}^{\mathrm{FO}}$ (resp., $\Pi_{\mathrm{k}}^{\mathrm{FO}}$) of the STH if it can be defined by a $\Sigma_{\mathrm{k}}^{\mathrm{FO}}$-sentence (resp., $\Pi_{\mathrm{k}}^{\mathrm{FO}}$-sentence). $\Delta_{\mathrm{k+1}}^{\mathrm{FO}}$ denotes the intersection of $\Sigma_{\mathrm{k}}^{\mathrm{FO}}$ and $\Pi_{\mathrm{k}}^{\mathrm{FO}}$.

3 Machines with Computation Trees Having ε-Leaves

We introduce the e-model of leaf languages which is inspired by the observation that rejecting paths of nondeterministic computations act as neutral elements. We allow nondeterministic transducers not only to output single letters, but also to output the empty word ε. After the formal definition we introduce pte-reducibility which allows us to formulate and prove an analogon of the BCSV-theorem. Furthermore, we show that the e-model connects the polynomial-time hierarchy with the Straubing-Thérien hierarchy.

For a finite alphabet Σ and $a \notin \Sigma$, we define a homomorphism $h_{\Sigma,a} : (\Sigma \cup \{a\})^* \to \Sigma^*$ by $h_{\Sigma,a}(b) =_{\text{def}} b$ for $b \in \Sigma$ and $h_{\Sigma,a}(a) =_{\text{def}} \varepsilon$.

Definition 3.1 *Let $(L, K) \subseteq \Sigma^*$. The class $\mathrm{Leaf}_\varepsilon^{\mathrm{p}}(L, K)$ consists of all languages A for which there exists a nondeterministic polynomial time transducer M producing on every computation path a symbol from Σ or the empty word ε such that the following holds:*

$$\begin{aligned} x \in A &\Rightarrow \beta_M(x) \in L, \\ x \notin A &\Rightarrow \beta_M(x) \in K. \end{aligned}$$

For $(L, K) \subseteq \Sigma^*$, if $K = \Sigma^* - L$, we will often use $\mathrm{Leaf}_\varepsilon^{\mathrm{p}}(L)$ as abbreviation for $\mathrm{Leaf}_\varepsilon^{\mathrm{p}}(L, K)$. In these cases, we will make clear what alphabet we use for L. Notice the it makes no difference whether we use balanced or unbalanced computation trees. So for convenience we may assume that paths not only can output single letters, but arbitrary words.

Example 3.2 *1.* $\mathrm{Leaf}_\varepsilon^{\mathrm{p}}(11^*, \varepsilon) = \mathrm{Leaf}_\varepsilon^{\mathrm{p}}(0^*1(0 \vee 1)^*, 0^*) = \mathrm{NP}$.
2. Let $L =_{\text{def}} \{1\} \subseteq \{0,1\}^*$. *Then* $\mathrm{Leaf}_\varepsilon^{\mathrm{p}}(L) = 1\mathrm{NP}$.
3. $\mathrm{Leaf}_\varepsilon^{\mathrm{p}}(1, \varepsilon) = \mathrm{UP}$.

A function g is computable in polylogarithmic time if there exists $k \geq 1$ such that $g(x)$ can be computed in time $\mathcal{O}(\log^k |x|)$ by a Turing-machine which accesses the input as an oracle.

Definition 3.3 *Let $(L, K) \subseteq \Sigma_1^*, (L', K') \subseteq \Sigma_2^*$ and $a \notin \Sigma_1^* \cup \Sigma_2^*$. Then $(L, K) \leq_{\mathrm{m}}^{\mathrm{pte}} (K, K')$ if and only if there exists a function $f : (\Sigma_1 \cup \{a\})^* \to (\Sigma_2 \cup \{a\})^*$ such that*

- *there exist functions $g : (\Sigma_1 \cup \{a\})^* \to \Sigma_2 \cup \{a\}, h : (\Sigma_1 \cup \{a\})^* \to \mathbb{N}$ computable in polylogarithmic time such that for all $x \in (\Sigma_1 \cup \{a\})^*$, $f(x) = g(x, 1)g(x, 2) \ldots g(x, h(x))$,*
- *for all $x \in (\Sigma_1 \cup \{a\})^*$, $\big(h_{\Sigma_1,a}(x) \in L \Rightarrow h_{\Sigma_2,a}(f(x)) \in L'\big)$,*
- *for all $x \in (\Sigma_1 \cup \{a\})^*$, $\big(h_{\Sigma_1,a}(x) \in K \Rightarrow h_{\Sigma_2,a}(f(x)) \in K'\big)$.*

If $(L, K) \leq_{\mathrm{m}}^{\mathrm{pte}} (L', K')$ holds and $K = \Sigma_1^* - L$ and $K' = \Sigma_2^* - L'$, we will often use $L \leq_{\mathrm{m}}^{\mathrm{pte}} K$ as abbreviation.

Lemma 3.4 *For $(L, K) \subseteq \Sigma_1^*$, $(L', K') \subseteq \Sigma_2^*$ where $a \notin \Sigma_1 \cup \Sigma_2$, it holds that $(L, K) \leq_{\mathrm{m}}^{\mathrm{pte}} (L', K')$ if and only if $(L_a, K_a) \leq_{\mathrm{m}}^{\mathrm{plt}} (L'_a, K'_a)$.*

Proof This is an immediate consequence of the definition of $\leq_{\mathrm{m}}^{\mathrm{pte}}$: Let $(L, K) \subseteq \Sigma_1^*$, $(L', K') \subseteq \Sigma_2^*$ and $a \notin \Sigma_1 \cup \Sigma_2$. Observe that for the if-part, it suffices to modify the reducing function such that it outputs ε instead of a. For the only if-part, it is the other way round. $\Box$

Lemma 3.5 *For $(L, K) \subseteq \Sigma^*$ and $a \notin \Sigma$, $\mathrm{Leaf}_{\varepsilon}^{\mathrm{p}}(L, K) = \mathrm{Leaf}_{\mathrm{b}}^{\mathrm{p}}(L_a, K_a) = \mathrm{Leaf}_{\mathrm{u}}^{\mathrm{p}}(L_a, K_a) = \mathrm{Leaf}_{\varepsilon}^{\mathrm{p}}(L_a, K_a)$.*

Proof It suffices to show that $\mathrm{Leaf}_{\varepsilon}^{\mathrm{p}}(L, K) \subseteq \mathrm{Leaf}_{\mathrm{u}}^{\mathrm{p}}(L_a, K_a)$ and $\mathrm{Leaf}_{\varepsilon}^{\mathrm{p}}(L_a, K_a) \subseteq \mathrm{Leaf}_{\varepsilon}^{\mathrm{p}}(L, K)$. For the first inclusion, let $A \in \mathrm{Leaf}_{\varepsilon}^{\mathrm{p}}(L, K)$ via the nondeterministic transducer M, which outputs symbols from $\Sigma \cup \{\varepsilon\}$. M can easily be transformed into a transducer M' which proves that $A \in \mathrm{Leaf}_{\mathrm{b}}^{\mathrm{p}}(L_a, K_a)$: M' works like M, but whenever M outputs ε, M' outputs a. For the second inclusion, let again M be the nondeterministic transducer which proves $A \in \mathrm{Leaf}_{\varepsilon}^{\mathrm{p}}(L_a, K_a)$. Observe that letters a in the leafstring of M on an input x have no influence on whether x belongs to A or not. Hence, we can transform M into a machine M' that outputs ε whenever M outputs a. Hence, $A \in \mathrm{Leaf}_{\varepsilon}^{\mathrm{p}}(L, K)$. $\Box$

We obtain the following BCSV-theorem for the e-model.

Theorem 3.6 *Let $(L, K) \subseteq \Sigma_1^*$ and $(L', K') \subseteq \Sigma_2^*$. Then the following statements are equivalent:*

1. *$(L, K) \leq_{\mathrm{m}}^{\mathrm{pte}} (L', K')$.*
2. *For all oracles O it holds that $\mathrm{Leaf}_{\varepsilon}^{\mathrm{p}}(L, K)^O \subseteq \mathrm{Leaf}_{\varepsilon}^{\mathrm{p}}(L', K')^O$.*

Proof Let $L, K \subseteq \Sigma_1^*, L \cap K = \emptyset$, $L', K' \subseteq \Sigma_2^*, L' \cap K' = \emptyset$ and $a \notin \Sigma_1 \cup \Sigma_2$. Then the following equivalences hold:

$$\begin{aligned}(L,K)\leq_{\mathrm{m}}^{\mathrm{pte}}(L',K') &\overset{\mathrm{I.}}{\Leftrightarrow} (L_a,K_a)\leq_{\mathrm{m}}^{\mathrm{plt}}(L'_a,K'_a),\\ &\overset{\mathrm{II.}}{\Leftrightarrow} \forall O, \mathrm{Leaf}_{\mathrm{b}}^{\mathrm{p}B}(L_a,K_a) \subseteq \mathrm{Leaf}_{\mathrm{b}}^{\mathrm{p}B}(L'_a,K'_a),\\ &\overset{\mathrm{III.}}{\Leftrightarrow} \forall O, \mathrm{Leaf}_{\varepsilon}^{\mathrm{p}B}(L,K) \subseteq \mathrm{Leaf}_{\varepsilon}^{\mathrm{p}B}(L',K').\end{aligned}$$

Note that I. holds because of Lemma 3.4, II. holds because of [BCS92, Ver93], and III. holds because Lemma 3.5 is relativizable. □

The next theorem shows a connection between the STH and the PH via the e-model. A similar connection for the existing b- and u-models was proved by Hertrampf et al. [HLS$^+$93], Burtschick and Vollmer [BV98], and Borchert et al. [BLS$^+$04].

Theorem 3.7 *Let $k \geq 1$.*

1. $\mathrm{Leaf}_{\varepsilon}^{\mathrm{p}}(\Sigma_{\mathrm{k}}^{\mathrm{FO}}) = \Sigma_{\mathrm{k}}^{\mathrm{P}}$
2. $\mathrm{Leaf}_{\varepsilon}^{\mathrm{p}}(\Pi_{\mathrm{k}}^{\mathrm{FO}}) = \Pi_{\mathrm{k}}^{\mathrm{P}}$
3. $\mathrm{Leaf}_{\varepsilon}^{\mathrm{p}}(\Delta_{\mathrm{k}}^{\mathrm{FO}}) = \Delta_{\mathrm{k}}^{\mathrm{P}}$

Proof Let $B \subseteq \Sigma^*$ and choose a new letter $a \notin \Sigma$. We show: if $B \in \Sigma_{\mathrm{k}}^{\mathrm{FO}}$, then $B_a \in \Sigma_{\mathrm{k}}^{\mathrm{FO}}$. Let ϕ be a $\Sigma_{\mathrm{k}}^{\mathrm{FO}}$-sentence defining B. Assume $\phi = Q_1 i_1 Q_2 i_2 \cdots Q_n i_n\, \psi$ where the Q's are quantifiers, the i's are variables, and ψ is quantifier-free. Now replace the quantifiers $Q_n, \ldots, Q_1$ and the formulas α they range on:

$$\begin{aligned}\exists i\,\alpha \quad &\text{is replaced by} \quad \exists i\,(\neg\pi_a i \wedge \alpha)\\ \forall i\,\alpha \quad &\text{is replaced by} \quad \forall i\,(\pi_a i \vee \alpha)\end{aligned}$$

Denote the resulting formula by ϕ'. Observe that ϕ' defines B_a and that ϕ' can be converted to a $\Sigma_{\mathrm{k}}^{\mathrm{FO}}$-sentence. Hence $B_a \in \Sigma_{\mathrm{k}}^{\mathrm{FO}}$. The same argument shows (i) if $L \in \Pi_{\mathrm{k}}^{\mathrm{FO}}$, then $L_a \in \Pi_{\mathrm{k}}^{\mathrm{FO}}$ and (ii) if $L \in \Delta_{\mathrm{k}}^{\mathrm{FO}}$, then $L_a \in \Delta_{\mathrm{k}}^{\mathrm{FO}}$.

If we consider the u-model instead of the e-model, then the statements of the theorem are known [BV98, BSS99, BLS$^+$04]. By $\mathrm{Leaf}_{\mathrm{u}}^{\mathrm{p}}(B) \subseteq \mathrm{Leaf}_{\varepsilon}^{\mathrm{p}}(B)$, it suffices to argue for the inclusions from left to right. Let $L \in \mathrm{Leaf}_{\varepsilon}^{\mathrm{p}}(\Sigma_{\mathrm{k}}^{\mathrm{FO}})$, i.e., there exists $B \in \Sigma_{\mathrm{k}}^{\mathrm{FO}}$ such that $L \in \mathrm{Leaf}_{\varepsilon}^{\mathrm{p}}(B)$. By Lemma 3.5, $L \in \mathrm{Leaf}_{\mathrm{u}}^{\mathrm{p}}(B_a)$ where a is a new letter. As argued above, $B_a \in \Sigma_{\mathrm{k}}^{\mathrm{FO}}$ and therefore, $L \in \mathrm{Leaf}_{\mathrm{u}}^{\mathrm{p}}(\Sigma_{\mathrm{k}}^{\mathrm{FO}}) \subseteq \Sigma_{\mathrm{k}}^{\mathrm{P}}$. The inlucions $\mathrm{Leaf}_{\varepsilon}^{\mathrm{p}}(\Pi_{\mathrm{k}}^{\mathrm{FO}}) \subseteq \Pi_{\mathrm{k}}^{\mathrm{P}}$ and $\mathrm{Leaf}_{\varepsilon}^{\mathrm{p}}(\Delta_{\mathrm{k}}^{\mathrm{FO}}) \subseteq \Delta_{\mathrm{k}}^{\mathrm{P}}$ follow analogously. □

4 Gap Theorems for NP, Δ_2^{P}, and Σ_2^{P}

In this section we use existing forbidden-pattern characterizations to obtain lower bounds for certain e-classes. From this we derive gap theorems for NP, Δ_2^{P}, and Σ_2^{P}. A summary of these results can be found in Table 1.

Pin and Weil [PW97] proved the following forbidden-pattern characterization of level Σ_1^{FO} of the STH.

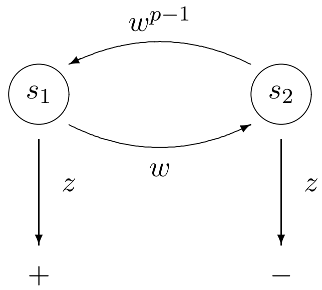

Figure 1: Forbidden pattern for SF where p is prime.

Proposition 4.1 ([PW97]) *The following are equivalent for any language A.*

1. $A \in \Sigma_1^{\mathrm{FO}}$
2. $\forall v, w \in \Sigma^*[v \preceq w \Rightarrow \chi_A(v) \leq \chi_A(w)]$
3. $\forall v, w \in \Sigma^* \forall a \in \Sigma[\chi_A(vw) \leq \chi_A(vaw)]$

This characterization enables us to prove lower bounds for the e-class of languages outside Σ_1^{FO}. In combination with Theorem 3.7 we obtain a gap theorem for NP (Corollary 4.3).

Theorem 4.2 *Let A be an arbitrary language.*

1. *If $A \notin \Sigma_1^{\mathrm{FO}}$, then* $\mathrm{coUP} \subseteq \mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$.
2. *If $A \in \mathrm{REG} - \Sigma_1^{\mathrm{FO}}$, then* $\mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$ *contains at least one of the following classes:* coNP, $\mathrm{co1NP}$, $\mathrm{MOD}_p\mathrm{P}$ *for a prime p.*
3. *If $A \in \mathrm{SF} - \Sigma_1^{\mathrm{FO}}$, then* $\mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$ *contains at least one of the following classes:* coNP, $\mathrm{co1NP}$.

Proof If $A \notin \Sigma_1^{\mathrm{FO}}$, then by Proposition 4.1, there exist words v, w and a letter a such that $vw \in A$ and $vaw \notin A$. Let $L \in \mathrm{coUP}$, i.e., there exists a nondeterministic polynomial-time machine M such that on input $x \notin L$, M has exactly one accepting path, and input $x \in L$, M has no accepting path. We modify M such that accepting paths output a, rejecting paths output ε, an additional path on the left outputs v, and an additional path on the right outputs w. It follows that for inputs $x \in L$ the generated leaf word is vw, and for inputs $x \notin L$ the generated word is vaw. This shows $L \in \mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$ and hence $\mathrm{coUP} \subseteq \mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$.

Now assume $A \in \mathrm{REG} - \Sigma_1^{\mathrm{FO}}$.

Case 1: $A \notin \mathrm{SF}$. By Sch¨utzenberger [Sch65] and McNaughton and Papert [MP71], A's minimal automaton contains the counting pattern (Fig. 1). So there exist words y, w, z and a prime p such that for all i, $yw^{ip}z \in A$ and $yw^{ip+1}z \notin A$. We show $\mathrm{MOD}_p\mathrm{P} \subseteq \mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$. Let $L \in \mathrm{MOD}_p\mathrm{P}$ and let M be a nondeterministic polynomial-time machine such that $x \in L$ if and only if the number of accepting paths of M on input x is $\equiv 0 (\mathrm{mod}\, p)$. Without loss of generality we may assume that if $x \notin L$, then the latter number is $\equiv 1 (\mathrm{mod}\, p)$. (If not, then simulate M's computation $p-1$ times in a row, which takes the number of accepting paths to the power of $p-1$ and hence, by Fermat's theorem, results in a number that either is $\equiv 0 (\mathrm{mod}\, p)$ or is $\equiv 0 (\mathrm{mod}\, p)$.) We modify M such that accepting paths output w, rejecting paths output ε, an additional path on the left outputs y, and an additional path on the right outputs z. Hence, for inputs $x \in L$ the generated leaf word is of the form $yw^{ip}z$, and for inputs $x \notin L$ the generated leaf word is of the form $yw^{ip+1}z$. This shows $L \in \mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$ and hence $\mathrm{MOD}_p\mathrm{P} \subseteq \mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$.

Case 2: $A \in \mathrm{SF} - \Sigma_1^{\mathrm{FO}}$. By Proposition 4.1, there exist words v, w and a letter a such that $vw \in A$ and $vaw \notin A$. By the pumping lemma, there exist $j, n \geq 1$ such that for all i, $va^jw \in A \Leftrightarrow va^{j+in}w \in A$. Choose the smallest such n. By Sch¨utzenberger [Sch65] and McNaughton and Papert [MP71], $A \in \mathrm{SF}$ implies that A's minimal automaton does *not* contain the counting pattern (Fig. 1). Therefore, n must be equal to 1 and it follows that va^ja^*w either is a subset of A or is a subset of $\overline{A}$.

Assume $va^ja^*w \subseteq \overline{A}$. Hence $vw \in A$ and $v(a^j)^+w \subseteq \overline{A}$. We show $\mathrm{coNP} \subseteq \mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$. Let $L \in \mathrm{coNP}$ and let M be a nondeterministic polynomial-time machine that accepts $\overline{L}$. We modify M such that accepting paths output a^j, rejecting paths output ε, an additional path on the left outputs v, and an additional path on the right outputs w. If $x \in L$, then the modifi ed machine produces the leaf word vw; otherwise it produces a leaf word from $v(a^j)^+w$. This shows $L \in \mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$ and hence $\mathrm{coNP} \subseteq \mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$.

Assume $va^ja^*w \subseteq A$. Choose the smallest $k \in [1, j)$ such that $va^ka^+w \subseteq A$. Hence $va^kw \notin A$. We show $\mathrm{co1NP} \subseteq \mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$. Let $L \in \mathrm{co1NP}$ and let M be a nondeterministic polynomial-time machine such that $x \notin L$ if and only if M on x produces exactly one accepting path. We modify M such that accepting paths output a^k, rejecting paths output ε, an additional path on the left outputs v, and an additional path on the right outputs w. If $x \in L$, then the modifi ed machine produces a leaf word in $va^ka^+w \cup \{vw\}$; otherwise it produces the leaf word va^kw. This shows $L \in \mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$ and hence $\mathrm{co1NP} \subseteq \mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$. $\square$

Corollary 4.3 *Let B be a nontrivial language.*

1. *The e-class of B either is contained in* NP*, or contains* coUP.
2. *If $B \in \mathrm{REG}$, then the e-class of B either is contained in* NP*, or contains at least one of the following classes:* coNP, co1NP, $\mathrm{MOD}_p\mathrm{P}$ *for a prime p.*
3. *If $B \in \mathrm{SF}$, then the e-class of B either is contained in* NP*, or contains at least one of the following classes:* coNP, co1NP.

Proof Follows from Theorems 3.7 and 4.2. $\square$

Now we can prove general lower bounds for e-classes. In particular, no complexity class below UP is defi nable with this concept.

Corollary 4.4 *Let A be a nontrivial language.*

1. $\mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$ *contains at least one of the following classes:* UP, coUP.
2. *If $A \in \mathrm{REG}$, then* $\mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$ *contains at least one of the following classes:* NP, coNP, $\mathrm{MOD}_p\mathrm{P}$ *for a prime p.*

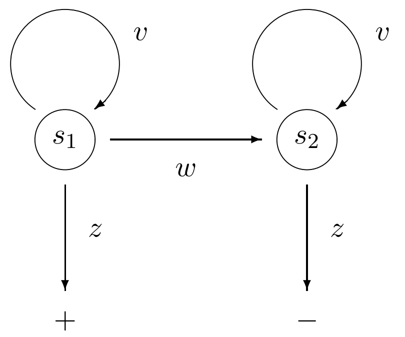

Figure 2: Forbidden pattern for Σ_2^{FO} where $w \preceq v$.

3. *If $A \in \mathrm{SF}$, then $\mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$ contains at least one of the following classes:* NP, coNP.

Proof By assumption there exist a word in A and a word not in A. If $\varepsilon \in A$, then by Proposition 4.1, $A \notin \Sigma_1^{\mathrm{FO}}$; otherwise $\overline{A} \notin \Sigma_1^{\mathrm{FO}}$. It follows from Theorem 4.2 that $\mathrm{coUP} \subseteq \mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$ or $\mathrm{coUP} \subseteq \mathrm{Leaf}_\varepsilon^{\mathrm{P}}(\overline{A}) = \mathrm{coLeaf}_\varepsilon^{\mathrm{P}}(A)$. Hence, $\mathrm{UP} \subseteq \mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$ or $\mathrm{coUP} \subseteq \mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$.

If A additionally belongs to REG, then $\mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$ or $\mathrm{Leaf}_\varepsilon^{\mathrm{P}}(\overline{A})$ contains at least one of the following classes: coNP, $\mathrm{co1NP}$, $\mathrm{MOD}_p\mathrm{P}$ for a prime p. Hence $\mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$ contains at least one of the following classes: NP, coNP, $\mathrm{MOD}_p\mathrm{P}$ for a prime p. If A even belongs to SF, then the same argument shows that $\mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$ contains at least one of the following classes: NP, coNP. □

Under reasonable assumptions that there is no regular A such that A's e-class lies strictly between coNP and $\mathrm{1NP}$. By symmetry, the same holds for NP and $\mathrm{co1NP}$.

Corollary 4.5 *Let $A \in \mathrm{REG}$ be a nontrivial language. Assume $\mathrm{NP} \not\subseteq \mathrm{1NP}$ and $\mathrm{MOD}_p\mathrm{P} \not\subseteq \mathrm{1NP}$ for all primes p. Then the following implication holds.*

$$\mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A) \subsetneq \mathrm{1NP} \Rightarrow \mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A) \subseteq \mathrm{coNP}.$$

Proof If $\mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A) \not\subseteq \mathrm{coNP}$, then $A \notin \mathrm{co}\mathcal{L}_{1/2}$. By Theorem 4.2, $\mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$ contains at least one of the following classes: NP, $\mathrm{1NP}$, $\mathrm{MOD}_p\mathrm{P}$ for a prime p. □

Starting with a forbidden-pattern characterization for Σ_2^{FO} [PW97] (Figure 2) we develop a lower bound for the e-class of Σ_2^{FO}. Again, this yields a gap theorem, this time for Σ_2^{P} (Corollary 4.7).

Theorem 4.6 *If $A \in \mathrm{REG} - \Sigma_2^{\mathrm{FO}}$, then $\mathrm{AU}\Pi_2^{\mathrm{P}} \subseteq \mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$.*

Proof Pin and Weil [PW97] proved the following forbidden pattern characterization of Σ_2^{FO}: A regular language belongs to Σ_2^{FO} if and only if the transition graph of its minimal automaton does not contain the subgraph shown in Figure 2. So by assumption, A's minimal automaton contains this graph.

Let $L \in \mathrm{AU}\Pi_2^{\mathrm{P}}$, i.e., there exist $B \in \mathrm{P}$ and polynomials p and q such that for all x,

$$\begin{aligned} x \in L \quad &\Rightarrow \quad \forall y \in \Sigma^{p(|x|)}, \exists! z \in \Sigma^{q(|x|)}[(x, y, z) \in B], \\ x \notin L \quad &\Rightarrow \quad \text{there exists } y \in \Sigma^{p(|x|)} \text{ such that the following holds:} \\ &\qquad \text{(i) } \forall z \in \Sigma^{q(|x|)}[(x, y, z) \notin B], \\ &\qquad \text{(ii) } \forall u \in \Sigma^{p(|x|)} - \{y\}, \exists! z \in \Sigma^{q(|x|)}[(x, u, z) \in B]. \end{aligned}$$

We describe a nondeterministic machine M on input x: First, M nondeterministically guesses $y \in \Sigma^{p(|x|)}$. Now M splits into $|v|$ paths which we associate with the letters of v. Consider the first occurrence of w as a subword of v. The paths that are associated with the positions involved in this occurrence output the respective letters of v and stop. On all other paths (i.e., those which are not involved in the first occurence of w in v) the computation is continued as follows: Assume we are on a path that is assiciated with letter c in v. M nondeterministically guesses $z \in \Sigma^{q(|x|)}$. If $(x, y, z) \in B$, then output ε and stop. Otherwise, output c and stop.

In order to determine the leaf string $\beta_M(x)$ we first consider certain factors of this string. More precisely, let β_u be the leaf string that is produced by the paths that guess $y = u$. Note that

$$\beta_M(x) = \beta_0 \beta_1 \cdots \beta_{2^{p(|x|)}}.$$

Assume $u \in \Sigma^{p(|x|)}$ such that $\exists! z \in \Sigma^{q(|x|)}[(x, u, z) \in B]$. Consider the path where M guesses $y = u$. In the next steps, M splits into $|v|$ paths associated with the letters of v. The paths involved in the first occurrence of w in v will output the respective letters from v. Each remaining path continues the computation. By assumption, there exists exactly one z such that $[(x, y, z) \in B]$. The path guessing that z will output the respective letter in v, while all other paths will output ε. Therefore, $\beta_u = v$.

Assume $u \in \Sigma^{p(|x|)}$ such that $\forall z \in \Sigma^{q(|x|)}[(x, u, z) \notin B]$. Consider the path where M guesses $y = u$. Again M splits into $|v|$ paths. The paths involved in the occurrence of w will output the respective letters from v. However, now there is no z such that $[(x, y, z) \in B]$ and therefore, all remaining paths output ε. It follows that $\beta_u = w$.

Now let us consider $\beta_M(x)$. If $x \in L$, then for all u, $\exists! z \in \Sigma^{q(|x|)}[(x, u, z) \in B]$. Therefore, all u, $\beta_u = v$ and it follows that $\beta_M(x) \in v^*$. Otherwise, $x \notin L$. So there exists $y \in \Sigma^{p(|x|)}$ such that (i) $\forall z \in \Sigma^{q(|x|)}[(x, y, z) \notin B]$ and (ii) for all $u \neq y$, $\exists! z \in \Sigma^{q(|x|)}[(x, u, z) \in B]$. Therefore, (i) $\beta_y = w$ and (ii) for all $u \neq y$, $\beta_u = v$. It follows that $\beta_M(x) \in v^* w v^*$. So we obtained:

$$\begin{aligned} x \in L \quad &\Rightarrow \quad \beta_M(x) \in v^* \\ x \notin L \quad &\Rightarrow \quad \beta_M(x) \in v^* w v^* \end{aligned}$$

Let y be a word leading from the initial state to s_1 in the minimal automaton of A. Let M' be the modification of M that on the left additionally outputs y and on the right additionally outputs z. Hence, $x \in L$ if and only if $\beta_{M'}(x) \in A$. This shows $L \in \mathrm{Leaf}_\varepsilon^{\mathrm{P}}(A)$. □

Corollary 4.7 *Let B be a nontrivial, regular language. The e-class of B either is contained in Σ_2^{P}, or contains $\mathrm{AU}\Pi_2^{\mathrm{P}}$.*

Proof Follows from Theorems 3.7 and 4.6. □

In addition, Theorem 4.6 gives us a lower bound for the e-class of Δ_2^{FO}:

Corollary 4.8 *If $A \in \mathrm{REG} - (\Sigma_2^{\mathrm{FO}} \cap \mathrm{co}\mathcal{L}_{3/2})$, then $\mathrm{Leaf}_\varepsilon^{\mathrm{p}}(A)$ contains at least one of the following classes:* $\mathrm{AU}\Pi_2^{\mathrm{P}}$, $\mathrm{AU}\Sigma_2^{\mathrm{P}}$.

Proof By assumption, A or $\overline{A}$ is outside Σ_2^{FO}. By Theorem 4.6, $\mathrm{AU}\Pi_2^{\mathrm{P}} \subseteq \mathrm{Leaf}_\varepsilon^{\mathrm{p}}(A)$ or $\mathrm{AU}\Pi_2^{\mathrm{P}} \subseteq \mathrm{coLeaf}_\varepsilon^{\mathrm{p}}(A)$. The latter is equivalent to $\mathrm{AU}\Sigma_2^{\mathrm{P}} \subseteq \mathrm{Leaf}_\varepsilon^{\mathrm{p}}(A)$. $\square$

Note that the following is the first gap theorem for Δ_2^{P}. It holds for both the u-model and the e-model.

Corollary 4.9 *Let B be a nontrivial, regular language.*

1. *The e-class of B either is contained in Δ_2^{P}, or contains at least one of the following classes:* $\mathrm{AU}\Sigma_2^{\mathrm{P}}$, $\mathrm{AU}\Pi_2^{\mathrm{P}}$.
2. *The u-class of B either is contained in Δ_2^{P}, or contains at least one of the following classes:* $\mathrm{AU}\Sigma_2^{\mathrm{P}}$, $\mathrm{AU}\Pi_2^{\mathrm{P}}$.

Proof The first statement is an immediate consequence of Theorem 3.7 and Corollary 4.8. For the second statement, let $\mathcal{B}_{3/2}$ denote level $3/2$ of the dot-depth hierarchy [CB71, PW97]. Schmitz [Sch01] showed that if $A \in \mathrm{REG} - (\mathcal{B}_{3/2} \cap \mathrm{co}\mathcal{B}_{3/2})$, then $\mathrm{Leaf}_{\mathrm{u}}^{\mathrm{p}}(A)$ contains at least one of the following classes: $\mathrm{AU}\Sigma_2^{\mathrm{P}}$, $\mathrm{AU}\Pi_2^{\mathrm{P}}$. Borchert et al. [BLS+04] mention that $\mathrm{Leaf}_{\mathrm{u}}^{\mathrm{p}}(\mathcal{B}_{3/2} \cap \mathrm{co}\mathcal{B}_{3/2}) = \Delta_2^{\mathrm{P}}$ can be obtained by an extension of their method. $\square$

5 A Gap Theorem for $1\mathrm{NP}$

In view of the gap theorems for NP and coNP (Corollary 4.3) it becomes evident that the classes $1\mathrm{NP}$ and $\mathrm{MOD}_p\mathrm{P}$ play an important role, since they appear as lower bounds. In this section we analyze $1\mathrm{NP}$ in detail and prove a gap theorem for this class. This case is more challenging since we cannot utilize an existing forbidden-pattern characterization. With Theorem 5.10 we give such a characterization for the class of languages corresponding to $1\mathrm{NP}$. Additionally, this theorem shows that with this class we have in fact identified *all* languages whose e-class is robustly contained in $1\mathrm{NP}$. This lets us derive a gap theorem for $1\mathrm{NP}$. For a given language L, we define the following conditions:

P1: There exist words $u \in L$, $v \notin L$, and $w \in L$ such that $u \preceq v \preceq w$.

P2: There exist $k \geq 2$ and nonempty words $u, v, w \in L$ such that $\{u, v\} \preceq_k w$ and $(\forall x)[x \prec u$ or $x \prec v \Rightarrow x \notin L]$.[3]

We interpret the patterns P1 and P2 as forbidden patterns and define a class of languages U of languages which neither fulfill P1 nor P2:

$$\mathrm{U} =_{\mathrm{def}} \{L : \text{P1 and P2 fail for } L\}$$

We will later on see that U is in fact a class of regular languages, and, more important, precisely characterizes the class $1\mathrm{NP}$ in the e-model of leaf-languages. The next two lemmas show that the e-class of a language which fulfills P1 or P2 is already quite powerful.

[3]Note that in P2, the words u and v can be the same.

Lemma 5.1 *Let $L \subseteq \Sigma^*$ such that L satisfies P1. Then*

$$\mathrm{Leaf}_\varepsilon^{\mathrm{p}}(L) \supseteq \mathrm{UP} \dot{\vee} \mathrm{coUP}.$$

Proof Let $L \subseteq \Sigma^*$ such that there exist words $u \in L$, $v \notin L$, and $w \in L$ such that $u \preceq v \preceq w$, i.e. L satisfies pattern P1. Furthermore, let $A \in \mathrm{UP} \dot{\vee} \mathrm{coUP}$. Hence $A = B \cup C$ where $B \in \mathrm{UP}$, $C \in \mathrm{coUP}$, and $B \subseteq \overline{C}$. Let M_B be the UP-machine accepting B, and let M_C be the coUP-machine accepting C. Observe that whenever M_B on an input x produces an accepting path (and thus accepts the input in an UP-sense), M_C also produces an accepting path and hence rejects (in an coUP-sense).

In order to prove $\mathrm{Leaf}_\varepsilon^{\mathrm{p}}(L) \supseteq \mathrm{UP} \dot{\vee} \mathrm{coUP}$, we show how to construct a nondeterministic polynomial-time Turing machine M such that the following holds for all x:

$$\begin{aligned} x \notin A &\implies \beta_M(x) = v \\ x \in B &\implies \beta_M(x) = w \\ x \in C &\implies \beta_M(x) = u \end{aligned}$$

Since $u \preceq v \preceq w = w_1 \dots w_k$, we can mark the letters of one fixed occurrence of u in w, we do the same with one fixed occurrence of v in w. Let $I_u \subsetneq \{1, \dots, l\}$ be the indices of letters in w that are marked to belong to u, and let $I_v \subsetneq \{1, \dots, l\}$ be the indices of letters in w that are marked to belong to v. Note that $\#I_v = |v|$, $\#I_u = |u|$, and $I_u \subsetneq I_v$.

For $1 \leq i \leq k$, we construct Turing machines M_i as follows:

- If $i \in I_u$, M_i develops only one path and outputs w_i on this path.
- If $i \in I_v \backslash I_u$, M_i simulates machine M_C on the same input. On every rejecting path of M_C, M_i outputs ϵ, if an accepting path exists, this path outputs w_i.
- If $i \notin I_v$, M_i simulates machine M_B on the same input. On every rejecting path of M_B, M_i outputs ϵ, if an accepting path exists, this path outputs w_i.

Turing machine M is constructed as follows: On input x, M branches into k nondeterministic paths. On path i, M then simulates M_i on input x. Notice that M can only produce leafstrings from $\{u, v, w\}$. It is easy to see that M satisfies the above condition: It holds that $x \in A \Leftrightarrow \beta_M(x) \in L$, and hence $A \in \mathrm{Leaf}_\varepsilon^{\mathrm{p}}(L)$. □

Lemma 5.2 *Let $L \subseteq \Sigma^*$ such that L satisfies P2. Then*

$$\mathrm{Leaf}_\varepsilon^{\mathrm{p}}(L) \supseteq \mathrm{UP} \vee \mathrm{UP}.$$

Proof Let $L \subseteq \Sigma^*$ such that there exists $k \geq 2$ and nonempty words $u, v, w = w_1 \dots w_l \in L$ such that $\{u, v\} \preceq_k w$ and $(\forall x)[x \prec u \text{ or } x \prec v \Rightarrow x \notin L]$. If $u \preceq_0 w$ we set $u := v$, if $v \preceq_0 w$ we set $v := u$. We obtain $\{u, v\} \preceq_k w$ and $u \preceq w, v \preceq w$. Observe that since L satisfies P2, the empty word ε cannot be in L, which has to have nonempty minimal words. Furthermore, let $A \in \mathrm{UP} \vee \mathrm{UP}$. Hence $A = B \cup C$ where $B \in \mathrm{UP}$ and $C \in \mathrm{UP}$. Let M_B be the UP-machine accepting B, and let M_C be the UP-machine accepting C.

In order to prove $\mathrm{Leaf}^{\mathrm{p}}_{\varepsilon}(L) \supseteq \mathrm{UP} \vee \mathrm{UP}$, we show how to construct a nondeterministic polynomial-time Turing machine M such that the following holds for all x:

$$\begin{aligned} x \notin A &\implies \beta_M(x) \prec u \\ x \in B \setminus C &\implies \beta_M(x) = u \\ x \in C \setminus B &\implies \beta_M(x) = v \\ x \in B \cap C &\implies \beta_M(x) = w \end{aligned}$$

Observe that no proper subword of u can be in L, since P2 requests that u and v are minimal words in L. Consequently, constructing a machine M as above yields $x \in A \Leftrightarrow \beta_M(x) \in L$ and hence $A \in \mathrm{Leaf}^{\mathrm{p}}_{\varepsilon}(L)$.

Since $u \preceq w$ and $v \preceq w$, we can mark the letters of one fixed occurrence of u in w, we do the same with one fixed occurrence of v in w.[4] Let $I_u \subsetneq \{1, \dots, l\}$ be the indices of letters in w that are marked to belong to u, and let $I_v \subsetneq \{1, \dots, l\}$ be the indices of letters in w that are marked to belong to v. Observe that $I_u \neq I_v$ and hence $\#(I_u \cap I_v) < \min(|u|, |v|)$.

For $1 \leq i \leq l$, we construct Turing machines M_i as follows:

I. If $i \in I_u \cap I_v$, M_i develops only one path and outputs w_i on this path.

II. If $i \in I_u \backslash I_v$, M_i simulates machine M_B on the same input. On every rejecting path of M_B, M_i outputs ϵ, if an accepting path exists, this path outputs w_i.

III. If $i \in I_v \backslash I_u$, M_i simulates machine M_C on the same input. On every rejecting path of M_C, M_i outputs ϵ, if an accepting path exists, this path outputs w_i.

IV. If $i \notin I_u \cup I_v$, M_i produces an accepting path if and only if M_B and M_C (running on the same input as M_i) produce an accepting path. Rejecting paths of M_i output ε.

Turing machine M is constructed as follows: On input x, M branches into l nondeterministic paths. On paths i for $1 \leq i \leq l$, M then simulates M_i on input x.

We consider the four different possibilities for the behavior of M on an input x. We do this by analyzing the behavior of the machines M consists of. Notice that depending on u, v, w, there might not be any machines of types I and IV.

Case 1: $x \notin A$. Hence, UP-machines M_B and M_C do not accept x, i.e. both produce only rejecting paths. Clearly, all machines of type II, III, and IV only output empty words. If $I_u \cap I_v \neq \emptyset$, precisely those letters of w are output that belong simultaneously to the marked occurrence of u and to the marked occurrence v. Let $i_1 < i_2 < \ldots < i_{\#(I_u \cap I_v)}$ be the elements of $I_u \cap I_v$, then $\beta_M(x) = w_{i_1} \ldots w_{i_{\#(I_u \cap I_v)}}$. As we then have $\beta_M(x) \prec u$, we can conclude that $\beta_M(x) \notin L$, since no subword of u is element of L. If $I_u \cap I_v = \emptyset$, $\beta_M(x) = \varepsilon$. Again, we conclude $\beta_M(x) \notin L$ since $\varepsilon \notin L$.

Case 2: $x \in B \setminus C$, i.e. M_B produces an accepting path on input x whereas M_C produces only rejecting paths. This means all machines of type III and IV output empty words. Recall that letters belonging to u and v simultaneously are created by machines of type I regardless of the input. So we obtain $\beta_M(x) = u$ and $\beta_M(x) \in L$.

Case 3: $x \in C \setminus B$. Analogous to case 2.

[4] If $u = v$, we fix two different occurrences of u in w.

Case 4: $x \in B \cap C$, i.e. M_B and M_C both produce an accepting path on input x. If $I_u \cup I_v = \{1, \ldots, k\}$, it is clear that $\beta_M(x) = w \in L$. $I_u \cup I_v \subsetneq \{1, \ldots, k\}$, the missing letters of w are produced by the machines of type IV. We again obtain $\beta_M(x) = w \in L$.

From the above case differentiation, we obtain $x \in A \Leftrightarrow \beta_M(x) \in L$ which proves $A \in \mathrm{Leaf}_\varepsilon^{\mathrm{p}}(L)$. $\square$

The next lemma gives simple languages that define the classes $1\mathrm{NP}$ and $\mathrm{UP}\,\dot\vee\,\mathrm{coUP}$ in terms of leaf-languages.

Lemma 5.3 *1.* $\mathrm{Leaf}_\varepsilon^{\mathrm{p}}(1, (\varepsilon \vee 111^*)) = 1\mathrm{NP}$.
2. $\mathrm{Leaf}_\varepsilon^{\mathrm{p}}((\varepsilon \vee 12), 2) = \mathrm{UP}\,\dot\vee\,\mathrm{coUP}$.
3. $\mathrm{Leaf}_\varepsilon^{\mathrm{p}}((1 \vee 2 \vee 12), \varepsilon) = \mathrm{UP} \vee \mathrm{UP}$.

Proof 1. For $\supseteq$, simply modify the $1\mathrm{NP}$-machine such that every accepting path outputs 1 and every rejecting path outputs ε. For $\subseteq$, modify the ε-machine such that every path that outputs ε then rejects, and every path that outputs 1 then accepts.

2. $\supseteq$: Let $A \in \mathrm{UP}\,\dot\vee\,\mathrm{coUP}$, that means $A = B \cup \overline{C}$ where $B, C \in \mathrm{UP}$ and $B \subseteq C$. Let M_B, M_C be the UP-machines that prove $B, C \in \mathrm{UP}$. As $B \subseteq C$, it holds for all inputs x that whenever M_B on input x produces an accepting path M_C on input x also produces an accepting path. We now construct a nondeterministic Turing-machine M as follows: In input x, M first branches nondeterministically. On the left path, M simulates M_B on input x, on the right path, it simulates M_C on input x. All rejecting paths of these simulations output ε, the accepting path of M_B (if existent) outputs 1, the accepting path of M_C (if existent) outputs 2. It is easy to see that the following now holds:

$$\begin{array}{rcl} \forall x, & & \beta_M(x) \in \{\varepsilon, 2, 12\}, \\ x \in B & \Rightarrow & \beta_M(x) = 12, \\ x \in \overline{C} & \Rightarrow & \beta_M(x) = \varepsilon, \\ x \notin A & \Rightarrow & \beta_M(x) = 2. \end{array}$$

This proves $A \in \mathrm{Leaf}_\varepsilon^{\mathrm{p}}((\varepsilon \vee 12), 2)$.

$\subseteq$: Let $A \in \mathrm{Leaf}_\varepsilon^{\mathrm{p}}((\varepsilon \vee 12), 2)$ via the nondeterministic ε-machine M. Observe that $A = B \cup \overline{C}$, where $B =_{\text{def}} \{x \mid \beta_M(x) = 12\}$ and $C =_{\text{def}} \{x \mid 2 \preceq \beta_M(x)\}$. Clearly, $B, C \in \mathrm{UP}$ and $B \subseteq C$. Hence, $A \in \mathrm{UP}\,\dot\vee\,\mathrm{coUP}$.

3. Analogous. $\square$

In order to show that for any language L, fulfillment of P1 suffices for the class $\mathrm{Leaf}_\varepsilon^{\mathrm{p}}(L)$ to be not robustly contained in $1\mathrm{NP}$, we first prove that languages characterizing $\mathrm{UP}\,\dot\vee\,\mathrm{coUP}$ cannot be pte-reduced to languages characterizing $1\mathrm{NP}$.

Lemma 5.4 $((\varepsilon \vee 12), 2) \not\leq_{\mathrm{m}}^{\mathrm{pte}} (1, (\varepsilon \vee 111^*))$.

Proof We assume that $(L, K) \leq_{\mathrm{m}}^{\mathrm{pte}} (L', K')$. Due to Lemma 3.4, this is equivalent to $(L_0, K_0) \leq_{\mathrm{m}}^{\mathrm{plt}} (L_0', K_0')$. Recall that $(L_0, K_0) =_{\text{def}} ((0^* \vee 0^*10^*20^*), 0^*20^*)$ and $(L_0', K_0') =_{\text{def}} (0^*10^*, (0^* \vee 0^*10^*1(0 \vee 1)^*))$. Say $(L_0, K_0) \leq_{\mathrm{m}}^{\mathrm{plt}} (L_0', K_0')$ holds via plt-reduction f.

This means there exist functions g, h which are computable in time $c \cdot \log^k$ fur suitable $c, k \geq 0$ such that $f(x) = g(x,1)g(x,2)\ldots g(x,h(x))$ and the following holds:

$$\begin{aligned} x \in L_0 \quad &\Rightarrow \quad f(x) = g(x,1)g(x,2)\ldots g(x,h(x)) \in L_0', \\ x \in K_0 \quad &\Rightarrow \quad f(x) = g(x,1)g(x,2)\ldots g(x,h(x)) \in K_0'. \end{aligned}$$

Let M_g be the deterministic polylog-time machine that computes g within the above time bound. We choose n sufficiently large such that $n > 2 \cdot c \cdot \log^k(n + c\log^k n) + 2$ and consider the input $w =_{\text{def}} 0^n$. Since $w \in L_0$, there exists precisely one $1 \leq i \leq h(w)$ such that $g(w,i) = 1$. Hence, M_g on input (w,i) outputs 1, while it outputs 0 on input (w,k) for all other k. Since $n > 2 \cdot g(n, h(n)) + 2$, M_g on input (w,i) cannot have queried all positions in w. Let j be a position that is not queried by M_g on input (w,i). We then set $v =_{\text{def}} 0^{j-1}20^{n-j}$. Notice that M_g still outputs 1 when ran on input (v,i) since w and v only differ on a position not queried by M_g on input (w,i). As $v \in K_0$, f has to output a word from K_0'. Since $g(v,i) = 1$, there has to be another i' such that $g(v,i') = 1$. Due to $n > 2 \cdot c \cdot \log^k(n + c\log^k n) + 2$, we can easily find a position $j' < j$ such that M_g neither queries j' on input (v,i) nor on input (v,i'). Let $u =_{\text{def}} 0^{j'-1}10^{j-j'+1}20^{n-j}$. As we still have $g(u,i) = g(u,i') = 1$, $f(u) \in K_0'$ although $u \in L_0$. By this contradiction, we have shown that no such f can exist. $\square$

Lemma 5.5 *There exists an oracle O such that* $\mathrm{UP} \,\dot\vee\, \mathrm{coUP} \not\subseteq 1\mathrm{NP}^O$.

Proof This follows directly from $\mathrm{Leaf}^{\mathrm{p}}_{\varepsilon}((\varepsilon \vee 12), 2) = \mathrm{UP} \,\dot\vee\, \mathrm{coUP}$, $\mathrm{Leaf}^{\mathrm{p}}_{\varepsilon}(1, (\varepsilon \vee 111^*)) = 1\mathrm{NP}$ (Lemma 5.3), $((\varepsilon \vee 12), 2) \not\leq^{\mathrm{pte}}_{\mathrm{m}} (1, (\varepsilon \vee 111^*))$ (Lemma 5.4) and Theorem 3.6. $\square$

Similarly to the above note, we prove that languages characterizing $\mathrm{UP} \vee \mathrm{UP}$ cannot be pte-reduced to languages characterizing 1NP. This is a step towards showing that for any language L, fulfillment of P2 suffices for the class $\mathrm{Leaf}^{\mathrm{p}}_{\varepsilon}(L)$ not to be robustly contained in 1NP.

Lemma 5.6 $((1 \vee 2 \vee 12), \varepsilon) \not\leq^{\mathrm{pte}}_{\mathrm{m}} (1, (\varepsilon \vee 111^*))$.

Proof We assume that $(L, K) \leq^{\mathrm{pte}}_{\mathrm{m}} (L', K')$. Due to Lemma 3.4, this is equivalent to $(L_0, K_0) \leq^{\mathrm{plt}}_{\mathrm{m}} (L_0', K_0')$. Recall that $(L_0, K_0) =_{\text{def}} ((0^*10^* \vee 0^*10^*20^* \vee 0^*20^*), 0^*)$ and $(L_0', K_0') =_{\text{def}} (0^*10^*, (0^* \vee 0^*10^*1(0 \vee 1)^*))$. Say $(L_0, K_0) \leq^{\mathrm{plt}}_{\mathrm{m}} (L_0', K_0')$ holds via plt-reduction f.

This means there exist functions g, h which are computable in time $c \cdot \log^k$ fur suitable $c, k \geq 0$ such that $f(x) = g(x,1)g(x,2)\ldots g(x,h(x))$ and the following holds:

$$\begin{aligned} x \in L_0 \quad &\Rightarrow \quad f(x) = g(x,1)g(x,2)\ldots g(x,h(x)) \in L_0', \\ x \in K_0 \quad &\Rightarrow \quad f(x) = g(x,1)g(x,2)\ldots g(x,h(x)) \in K_0'. \end{aligned}$$

Let M_g be the deterministic polylog-time machine that computes g within the above time bound. We choose n sufficiently large such that $\frac{n}{2} > 2 \cdot c \cdot \log^k(2n + c\log^k 2n)$ and consider words $x_i, y_i, z_{i,j} \in \{0,1,2\}^{2n}$: For $i, j \in \{1, \ldots, n\}$, we define $x_i =_{\text{def}} 0^{i-1}10^{n-i}0^n$, $y_i =_{\text{def}} 0^n0^{i-1}20^{n-i}$, and $z_{i,j} =_{\text{def}} 0^{i-1}10^{n-i}0^{j-1}20^{n-j}$. Observe that for $i, j \in \{1, \ldots, n\}$, x_i, y_i, and $z_{i,j}$ are all in L_0 and hence $f(x_i), f(y_i)$, and $f(z_{i,j})$ are all in L_0'. Therefore, we have

$$\forall a \in \{x, y\} \forall i \in \{1, \ldots, n\} \exists! j : g(a_i, j) = 1.$$

For $1 \leq i \leq n$, we define

$$\begin{aligned}
d(i) &=_{\text{def}} l, \text{ where } g(x_i, l) = 1, \\
B(i) &=_{\text{def}} \{l \in \{1, \ldots, 2n\} \mid M_g \text{ on input } (x_i, d(i)) \text{ queries position } l \text{ in } x_i\} \\
e(i) &=_{\text{def}} l, \text{ where } g(y_i, l) = 1, \\
C(i) &=_{\text{def}} \{l \in \{1, \ldots, 2n\} \mid M_g \text{ on input } (y_i, e(i)) \text{ queries position } l \text{ in } x_i\}
\end{aligned}$$

Claim: There exist $i, j \in \{1, \ldots, n\}$ such that $i \notin C(j)$ and $n + j \notin B(i)$.

Proof of the claim: Assuming that our claim is wrong, we conclude that for all $(i, j) \in \{1, \ldots, n\}^2$, it holds that $i \in C(j)$ or $n+j \in B(i)$. Without loss of generality, we can assume that $i \in C(j)$ holds for at least half of all (i, j), i.e. for at least $\frac{n^2}{2}$ tuples.[5] Observe that there now exists $1 \leq j \leq n$ such that among these $\frac{n^2}{2}$ tuples, there are tuples $(i_1, j), (i_2, j), \ldots, (i_{n/2}, j)$ such that $i_1 < i_2 < \ldots < i_{n/2}$. Hence, it holds that $i_1 \in C(j), i_2 \in C(j), \ldots i_{n/2} \in C(j)$. This in turn implies that M_g on input $(y_j, e(j))$ queries at least $\frac{n}{2}$ positions in y_j. Since we have chosen n sufficiently large such that $\frac{n}{2} > 2 \cdot g(2n, h(2n))$, M_g cannot query all these positions. So we have contradicted our assumption and thus proven the claim.

By this, we know that there exist $i, j \in \{1, \ldots, n\}$ such that $i \notin C(j)$ and $n + j \notin B(i)$. Using a standard technique, we can show that $d(i) \neq e(j)$.

Let us assume for a moment that $d(i) = e(j)$. This means that on input $(x_i, d(i))$, M_g does not query position $n+j$ in x_i, and on input $(y_j, d(i))$, M_g does not query position i in y_j. Recall that x_i and y_i only differ on positions i and $n+j$. Since M_g cannot distinguish between $(x_i, d(i))$ and $(y_j, d(i))$ until it has queried either position i or position $n + j$ (and may not be allowed to do so, depending on whether it is running on $(x_i, d(i))$ or $(y_j, d(i))$), the only way to get out of the dilemma is to neither query position i nor position $n + j$. However, this implies that $g(x_i, d(i)) = g(y_j, e(j)) = g(0^{2n}, d(i)) = 1$. Moreover, M_g cannot distinguish whether it is running on input $(x_i, d(i))$, $(y_j, d(i))$, or $(0^{2n}, d(i))$. Since $0^n \in K$,, there exists (at least one) $e' \neq d(i)$ such that $g(0^{2n}, e') = 1$. Let p be a position in 0^{2n} such that M_g neither queries p when running on input $(0^{2n}, d(i))$, nor when running on input $(0^{2n}, e')$. Such a position exists since $\frac{n}{2} > 2 \cdot g(2 \cdot n, h(2 \cdot n))$. Consequently, $g(0^{p-1}10^{2n-p}, d(i)) = 1$ and $g(0^{p-1}10^{2n-p}, e') = 1$. Hence, $f(0^{p-1}10^{2n-p}) \in K'$ although $0^{p-1}10^{2n-p} \in L$. This is a contradiction, hence $d(i) \neq e(j)$.

Since $i \notin C(j)$ and $n + j \notin B(i)$ it follows that $g(x_i, d(i)) = g(y_j, e(j)) = g(z_{i,j}, d(i)) = g(z_{i,j}, e(j)) = 1$. From $d(i) \neq e(j)$, we can then conclude that $f(z_{i,j}) \in 0^*10^*10^*$ and thus $f(z_{i,j}) \in K'$ although $z_{i,j} \in L$. This contradiction proves that no such f can exist; hence $(L, K) \not\leq_{\mathrm{m}}^{\mathrm{pte}} (L', K')$. □

Lemma 5.7 *There exists an oracle O such that* $\mathrm{UP} \vee \mathrm{UP} \not\subseteq 1\mathrm{NP}^O$.

Proof This follows directly from $\mathrm{Leaf}_\varepsilon^{\mathrm{p}}((1 \vee 2 \vee 12), \varepsilon) = \mathrm{UP} \vee \mathrm{UP}$ (Lemma 5.3.3), $\mathrm{Leaf}_\varepsilon^{\mathrm{p}}(1, (\varepsilon \vee 111^*)) = 1\mathrm{NP}$ (Lemma 5.3), $((1 \vee 2 \vee 12), \varepsilon) \not\leq_{\mathrm{m}}^{\mathrm{pte}} (1, (\varepsilon \vee 111^*))$ (Lemma 5.6) and Theorem 3.6. □

We now know that e-classes of languages outside U are not in 1NP. The next theorem will enable us to better understand the languages inside U. As it turns out, we can avail ourselves of a well-known algebraic property of Σ^* to obtain a convenient characterization of U.

[5] Otherwise, $n+j \in B(i)$ has to hold for at least $\frac{n^2}{2}$ tuples. The reasoning is analog.

Definition 5.8 *A partial ordering is a* well-partial ordering *if it contains no infinite descending sequence and no infinite antichain (i.e., a set of pairwise incomparable elements).*

Theorem 5.9 ([Hig52]) $(\Sigma^*, \preceq)$ *is a well-partial ordering.*

The following theorem gives the announced characterization of U, the class that precisely corresponds to $1\mathrm{NP}$ in the e-model.

Theorem 5.10 *The following statements are equivalent for any language $L \subseteq \Sigma^*$.*

1. $L \in \mathcal{R}^{\mathrm{pte}}(1)$*, the pte-closure of* $\{1\}$.
2. *For all oracles O it holds that* $\mathrm{Leaf}_\varepsilon^{\mathrm{p}}(L)^O \subseteq 1\mathrm{NP}^O$.
3. $L \in \mathrm{U}$*, that means both conditions, P1 and P2, fail for* L.
4. *There exist finite sets $A, B \subseteq \Sigma^*$ such that*

$$L = \{w \mid A \preceq_1 w \text{ and } (\forall v \in B)[v \not\preceq w]\}.^6 \tag{1}$$

Proof $\mathbf{1} \Leftrightarrow \mathbf{2}$: This is an immediate consequence of Theorem 3.6, since for all oracles O, $\mathrm{Leaf}_\varepsilon^{\mathrm{p}}(1)^O = 1\mathrm{NP}^O$.

$\mathbf{2} \Rightarrow \mathbf{3}$: Assume that relative to all oracles, $\mathrm{Leaf}_\varepsilon^{\mathrm{p}}(L) \subseteq 1\mathrm{NP}$. From Lemmas 5.1, 5.5 and Lemmas 5.2, 5.7, we know that if L satisfies P1 or P2, we can construct an oracle O such that $\mathrm{Leaf}_\varepsilon^{\mathrm{p}}(L)^O \not\subseteq 1\mathrm{NP}^O$. This contradicts our assumption. Therefore, L neither satisfies P1 nor P2.

$\mathbf{3} \Rightarrow \mathbf{4}$: Let

$$A = \{v \in L \mid (\forall v' \prec v)[v' \notin L]\}.$$

Observe that A can be seen as the set of minimal words in L. Furthermore, the elements in A are pairwise incomparable with respect to $\preceq$. From Theorem 5.9 and Definition 5.8 it follows that A is finite. Let

$$B = \{w \notin L \mid (\exists v \in A)[v \preceq w \text{ and } \forall w'[v \preceq w' \prec w \Rightarrow w' \in L]]\}.$$

This set can be thought of as the set of minimal words outside L that have predecessors in A. We claim that B is finite as well: Otherwise, since A is finite, there exists $v \in A$ such that the following subset of B is infinite.

$$B' = \{w \notin L \mid v \preceq w \text{ and } \forall w'[v \preceq w' \prec w \Rightarrow w' \in L]\}.$$

Observe that the elements in B' are pairwise incomparable with respect to $\preceq$. Again, from Theorem 5.9 and Definition 5.8 it follows that B is finite.

We are going to show equation (1). Let $w \in L$. So there exists $v \in A$ such that $v \preceq w$. Assume there exist different $v_1, v_2 \in A$ such that $v_1 \preceq w$ and $v_2 \preceq w$. It follows that v_1, v_2, and w are nonempty. This implies that L satisfies condition P2 which contradicts our assumption. Therefore, there exists exactly one $v \in A$ such that $v \preceq w$. If $v \preceq_k w$ for some $k \geq 2$, then L satisfies condition P2 which again is a contradiction. So $v \preceq_1 w$ and hence $A \preceq_1 w$.

[6] B can be thought of as the set of forbidden subwords, i.e., events that may not occur in words from L. Contrary, A represents the set of events such that every word in L triggers exactly one such event.

Assume now that there exists $v \in B$ such that $v \preceq w$. By B's definition, there exists $v' \in A$ such that $v' \preceq v$ and for all w', $[v' \preceq w' \prec v \Rightarrow w' \in L]$. In particular, $v' \preceq v \preceq w$ and $v' \in L$, $v \notin L$, and $w \in L$. Hence L satisfies condition P1 which contradicts our assumption. So there does not exist such $v \in B$ and therefore, w belongs to the right-hand side of equation (1). This shows the inclusion $\subseteq$ in equation (1).

Let w be an element of the right-hand side of (1). Hence there exists precisely one $v \in A$ such that $v \preceq w$. Assume $w \notin L$ and choose a shortest word $u \notin L$ such that $v \preceq u \preceq w$. It follows that $u \in B$. Together with $u \preceq w$ this implies that w is not an element of the right-hand side of (1). This contradiction shows $w \in L$ and finishes the proof of equation (1).

4 ⇒ 2 : Let $A = \{u_1, \ldots, u_m\}$ and $B = \{v_1, \ldots, v_n\}$ where $m = |A|$ and $n = |B|$. Let $L' \in \mathrm{Leaf}_\varepsilon^{\mathrm{p}}(L)$. So there exists a polynomial-time Turing machine M whose computation paths output symbols from $\Sigma \cup \{\varepsilon\}$ such that $x \in L' \Leftrightarrow \beta_M(x) \in L$. Define a nondeterministic machine N that works as follows on input x. First, N splits into $m+n$ paths $p_1, \ldots, p_m$ and $q_1, \ldots, q_n$. If $u_i = \varepsilon$, then path p_i outputs 1. If $u_i \neq \varepsilon$, then on path p_i the machine nondeterministically guesses an occurrence of u_i (by guessing the positions of u_i's letters) in the leaf string $\beta_M(x)$. If such a guess is successful, then N outputs 1, otherwise it outputs ε. Similarly, on path q_i the machine nondeterministically guesses an occurrence of v_i in $\beta_M(x)$. If such a guess is successful, then N outputs 11 (by producing two neighbouring paths with output 1), otherwise it outputs ε. From (1) it follows that

$$x \in L' \Leftrightarrow \beta_M(x) \in L \Leftrightarrow \beta_N(x) = 1.$$

Hence $L' \in \mathrm{Leaf}_\varepsilon^{\mathrm{p}}(1)$ and therefore $\mathrm{Leaf}_\varepsilon^{\mathrm{p}}(L) \subseteq \mathrm{Leaf}_\varepsilon^{\mathrm{p}}(1)$. Finally, observe that our argumentation is relativizable. □

Observe that due to the characterization of U given by Theorem 5.10.4, we immediately obtain that U only contains regular languages. We can now formulate the new gap theorem.

Theorem 5.11 *Let L be a nontrivial language.*

1. *If $L \in \mathrm{U}$, then the e-class of L is contained in* $1\mathrm{NP}$.
2. *If $L \notin \mathrm{U}$, then the e-class of L contains* $\mathrm{UP} \dot\vee \mathrm{coUP}$ *or* $\mathrm{UP} \vee \mathrm{UP}$.

Proof Follows from Theorem 5.10 and the fact that the Lemmas 5.2 and 5.1 are relativizable. □

Acknowledgments

We thank Bernd Borchert, Victor Selivanov, and Klaus W. Wagner for very interesting discussions and many helpful suggestions.

References

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

[BKS99] B. Borchert, D. Kuske, and F. Stephan. On existentially first-order definable languages and their relation to NP. *Theoretical Informatics and Applications*, 33:259–269, 1999.

[BLS+04] B. Borchert, K. Lange, F. Stephan, P. Tesson, and D. Thérien. The dot-depth and the polynomial hierarchy correspond on the delta levels. In *Developments in Language Theory*, pages 89–101, 2004.

[Bor95] B. Borchert. On the acceptance power of regular languages. *Theoretical Computer Science*, 148:207–225, 1995.

[Brz76] J. A. Brzozowski. Hierarchies of aperiodic languages. *RAIRO Inform. Theor.*, 10:33–49, 1976.

[BSS99] B. Borchert, H. Schmitz, and F. Stephan. Unpublished manuscript, 1999.

[BV98] H.-J. Burtschick and H. Vollmer. Lindström quantifiers and leaf language definability. *International Journal of Foundations of Computer Science*, 9:277–294, 1998.

[CB71] R. S. Cohen and J. A. Brzozowski. Dot-depth of star-free events. *Journal of Computer and System Sciences*, 5:1–16, 1971.

[Gla05] C. Glaßer. Polylog-time reductions decrease dot-depth. In *Proceedings 22nd Symposium on Theoretical Aspects of Computer Science*, volume 3404 of *Lecture Notes in Computer Science*. Springer Verlag, 2005.

[GOP+05] C. Glaßer, M. Ogihara, A. Pavan, A. L. Selman, and L. Zhang. Autoreducibility, mitoticity, and immunity. In *Proceedings 30th International Symposium on Mathematical Foundations of Computer Science*, volume 3618 of *Lecture Notes in Computer Science*, pages 387–398. Springer-Verlag, 2005.

[Hig52] G. Higman. Ordering by divisibility in abstract algebras. In *Proc. London Math. Soc.*, volume 3, pages 326–336, 1952.

[HLS+93] U. Hertrampf, C. Lautemann, T. Schwentick, H. Vollmer, and K. W. Wagner. On the power of polynomial time bit-reductions. In *Proceedings 8th Structure in Complexity Theory*, pages 200–207, 1993.

[MP71] R. McNaughton and S. Papert. *Counterfree Automata*. MIT Press, Cambridge, 1971.

[NR98] R. Niedermeier and P. Rossmanith. Unambiguous computations and locally definable acceptance types. *Theoretical Computer Science*, 194(1-2):137–161, 1998.

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

[PP86] D. Perrin and J. E. Pin. First-order logic and star-free sets. *Journal of Computer and System Sciences*, 32:393–406, 1986.

[PW97] J. E. Pin and P. Weil. Polynomial closure and unambiguous product. *Theory of computing systems*, 30:383–422, 1997.

[Sch65] M. P. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8:190–194, 1965.

[Sch01] H. Schmitz. *The Forbidden-Pattern Approach to Concatenation Hierarchies*. PhD thesis, Fakultät für Mathematik und Informatik, Universität Würzburg, 2001.

[ST04] H. Spakowski and R. Tripathi. On the power of unambiguity in alternating machines. Technical Report 851, University of Rochester, 2004.

[Str81] H. Straubing. A generalization of the Schützenberger product of finite monoids. *Theoretical Computer Science*, 13:137–150, 1981.

[Str85] H. Straubing. Finite semigroups varieties of the form V * D. *J. Pure Appl. Algebra*, 36:53–94, 1985.

[Thé81] D. Thérien. Classification of finite monoids: the language approach. *Theoretical Computer Science*, 14:195–208, 1981.

[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.

[Wag04] K. W. Wagner. Leaf language classes. In *Proceedings International Conference on Machines, Computations, and Universality*, volume 3354 of *Lecture Notes in Computer Science*. Springer Verlag, 2004.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'