

## THE DOT-DEPTH AND THE POLYNOMIAL HIERARCHIES CORRESPOND ON THE DELTA LEVELS

BERND BORCHERT

*Universität Tübingen, Germany*  
*borchert@informatik.uni-tuebingen.de*

KLAUS-JÖRN LANGE

*Universität Tübingen, Germany*  
*lange@informatik.uni-tuebingen.de*

FRANK STEPHAN\*

*National University of Singapore*  
*fstephan@comp.nus.edu.sg*

PASCAL TESSON†

*Université Laval, Québec, Canada*  
*pascal.tesson@ift.ulaval.ca*

and

DENIS THÉRIEN‡

*McGill University, Montreal, Canada*  
*denis@cs.mcgill.ca*

Received (received date)

Revised (revised date)

Communicated by Editor's name

### ABSTRACT

It is well-known that the  $\Sigma_k$ - and  $\Pi_k$ -levels of the dot-depth hierarchy and the polynomial hierarchy correspond via leaf languages. We extend this correspondence to the  $\Delta_k$ -levels of these hierarchies:  $\text{Leaf}^P(\Delta_k^L) = \Delta_k^P$ . The same methods are used to give evidence against an earlier conjecture of Straubing and Thérien about a leaf-language upper bound for BPP.

---

\*Research supported in part by NUS grant number RP3992710

†Most of this research took part while the author was visiting the University of Tübingen, supported by a fellowship from the Alexander von Humboldt Foundation.

‡Research supported by grants from NSERC, FQRNT and the Alexander von Humboldt Foundation.

## 1. Introduction

The leaf-language framework introduced by Hertrampf et al. [HL\*93] associates to any language or class of languages a complexity class. Most of the research in this field has focused on regular leaf languages: it is known for instance that the regular languages capture PSPACE in this sense and that the  $\Sigma_k$ - and  $\Pi_k$ -levels of the dot-depth hierarchy correspond with the  $\Sigma_k$ - and  $\Pi_k$ -levels of the polynomial hierarchy, i.e. for all  $k \geq 1$ :

$$\begin{aligned}\text{Leaf}^P(\Sigma_k^L) &= \Sigma_k^P, \\ \text{Leaf}^P(\Pi_k^L) &= \Pi_k^P.\end{aligned}$$

This was shown by Burtschick & Vollmer [BV98] and somewhat implicitly in [HL\*93]. As an immediate consequence the class of all starfree regular languages  $\mathcal{SF}$  and the polynomial hierarchy correspond via leaf languages (a fact already stated in [HL\*93]):

$$\text{Leaf}^P(\mathcal{SF}) = \text{PH}.$$

Furthermore, the  $k$ -th full level  $\mathcal{DD}_k$  of the dot-depth hierarchy (the Boolean closure of  $\Sigma_k^L$ ) and the Boolean closure of  $\Sigma_k^P$  (for  $k = 1$  called the *Boolean hierarchy over NP*) correspond via leaf languages, i.e.

$$\text{Leaf}^P(\mathcal{DD}_k) = \text{BC}(\Sigma_k^P).$$

Schmitz, Wagner and Selivanov [SW98, Sel01] further obtained correspondences between the classes of the Boolean hierarchies defined over the respective  $\Sigma_k$  classes.

The aim of this paper is to extend the correspondence of the dot-depth hierarchy and the polynomial hierarchy to the  $\Delta_k$ -levels of the two hierarchies which, in the dot-depth hierarchy, are defined as the intersections of the corresponding  $\Sigma_k$ - and  $\Pi_k$ -levels and, in the polynomial hierarchy, are defined as the polynomial-time Turing reducibility closure of the  $\Sigma_{k-1}$ -class (see Figure 1). For level 2 this correspondence was already shown in the unpublished manuscript [BSS99] and the proof used the Schützenberger characterization of  $\Delta_L^2$  as unambiguous products [Sch76] and a method from Wagner [Wa90] showing that the ODD MAX SAT problem is polynomial-time many-one complete for  $\Delta_2^P$ . The main result of this paper is that for all  $k \geq 2$ :

$$\text{Leaf}^P(\Delta_k^L) = \Delta_k^P.$$

As we will note, this correspondence still holds if we consider the  $\Delta$ -levels in the closely related Cohen-Brzozowski definition of dot-depth. The right to left containment was obtained independently by Travers [Tr02] (unpublished).

The most natural unit of classification for classes of regular languages is the notion of (positive)  $*$ -varieties of languages (see e.g. [Pi97]) and most results on leaf-languages, including the ones mentioned thus far, focus on characterizing  $\text{Leaf}^P(\mathcal{V})$  for some of the most-studied positive  $*$ -varieties  $\mathcal{V}$ . Our approach is based on relating well-known operators on positive  $*$ -varieties of languages with corresponding operators on complexity classes. We obtain our main result as a corollary of the

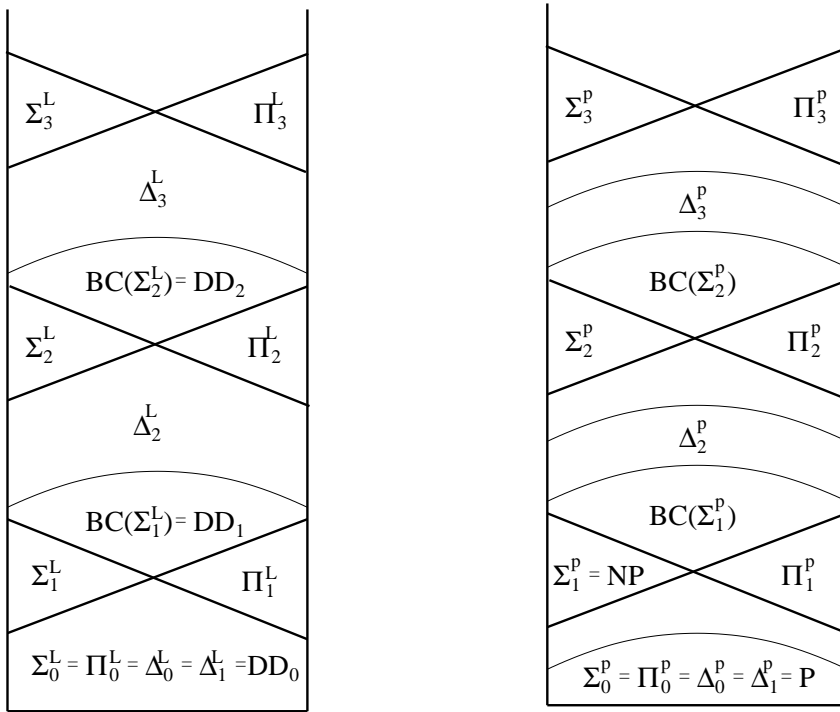


Figure 1: The dot-depth and the polynomial hierarchy

correspondence between the combined operator  $\text{UPol} \circ \text{BPol}$  on certain  $*$ -varieties of regular languages  $\mathcal{V}$  and the combination of polynomial-time Turing reducibility closure (denoted in dot operator notation  $T \cdot$  throughout this paper) applied to the projection operator  $\exists \cdot$  on complexity classes:

$$\text{Leaf}^P(\text{UPol}(\text{BPol}(\mathcal{V}))) = T \cdot \exists \cdot \text{Leaf}^P(\mathcal{V}).$$

Crucial ingredients of the proof of this operator correspondence are for the left to right containment a characterization of the  $\text{UPol}$  operator via certain unambiguous products by Pin, Straubing & Thérien [PST88], and for the other direction a generalization of the method of Wagner [Wa90] mentioned above.

Little is known about the  $\Delta$ -levels of the dot-depth hierarchy, with the notable exception of  $\Delta_L^2$ : the survey [TT02] showed that this class, also known as  $\mathcal{DA}$ , is of computational relevance and has many nice properties and characterizations. On the polynomial hierarchy side it is fair to say that the class  $\Delta_2^P$  is a “nobody” compared with the “celebrity” NP. But  $\Delta_2^P$  can be considered as the problems of polynomial-time complexity less or equal to that of NP-complete problems since the polynomial-time Turing reducibility is a very natural way to compare polynomial-time complexities of computational problems (“given an oracle for one problem for free, can you do the other problem in polynomial-time?”).

Straubing and Thérien conjectured that the class BPP of problems solvable by

two sided-error probabilistic polynomial time algorithms is contained in the class  $\text{Leaf}^P(\text{UPol}(\text{BPol}(\mathcal{G}_{\text{sol}})))$  where  $\mathcal{G}_{\text{sol}}$  denotes the  $*$ -variety of regular languages whose syntactic monoid is a solvable group. Our result allow us to better characterize this complexity class and to give evidence that this conjecture does not hold if we further restrict the groups to be  $p$ -groups for some prime  $p$ .

The paper is organized as follows. In Section 2 we recall the definition of the dot-depth hierarchy, and of the related operators  $\text{Pol}$  and  $\text{UPol}$ . In Section 3 we present some background on computational complexity and introduce the leaf language formalism. In Section 4 we prove the correspondence of the  $\text{Pol}$  operator on positive varieties of languages with the  $\exists$ -operator on complexity classes and obtain as a corollary the correspondence between the  $\Sigma_k$  and  $\Pi_k$  levels of the dot-depth and polynomial hierarchies. In Section 5 we give basic properties of the  $\Delta_k$  classes and prove our main result about the correspondence of the  $\Delta_k$  levels. Finally, in Section 6, we adapt our methods to study the conjecture of Straubing and Thérien on leaf-language upper bounds for BPP. An extended abstract of this paper appeared in the proceedings of the 8th Conference on Developments in Language Theory.

## 2. The Dot-Depth Hierarchy

We give in this section a quick overview of the language-theoretic background needed in our development and refer the reader to, e.g. [Pi97] for additional information and insight on these concepts.

A *class of languages*  $\mathcal{C}$  is a mapping which assigns to each alphabet  $\Sigma$  a set of languages  $\Sigma^*\mathcal{C}$  over  $\Sigma$ . Sometimes we write  $L \in \mathcal{C}$  as an abbreviation for  $L \in \Sigma^*\mathcal{C}$  for some alphabet  $\Sigma$ .

Let a class of languages  $\mathcal{C}$  be given. The *polynomial closure*  $\text{Pol}(\mathcal{C})$  is the class of languages consisting, for every alphabet  $\Sigma$ , of the finite unions of *marked products* of languages from  $\Sigma^*\mathcal{C}$ , i.e. languages  $L = L_0a_1L_1 \cdots a_nL_n$  such that the  $L_i$  are languages from  $\Sigma^*\mathcal{C}$  and the  $a_i$  are letters from  $\Sigma$ . We further say that the product  $L = L_0a_1L_1 \cdots a_nL_n$  is *unambiguous* if for every word  $w$  in  $L$  there is a unique factorization  $w = w_0a_1w_1 \cdots a_nw_n$  with  $w_i \in L_i$ . We denote as  $\text{UPol}(\mathcal{C})$  the class of finite disjoint unions of unambiguous marked products of languages of  $\mathcal{C}$ . The classes of languages  $\text{Co-}\mathcal{C}$  and  $\text{BC}$  are defined, for every alphabet  $\Sigma$ , as the set of complements of languages from  $\Sigma^*\mathcal{C}$  and as the Boolean closure of  $\Sigma^*\mathcal{C}$ , respectively.  $\text{Co-Pol}$  and  $\text{BPol}$  are defined as the combined operators  $\text{Co-} \circ \text{Pol}$  and  $\text{B} \circ \text{Pol}$ , respectively.

Let  $\mathcal{I}$  be set the class of languages which consists for every alphabet  $\Sigma$  only of the two languages  $\emptyset$  and  $\Sigma^*$ . The classes of the dot-depth hierarchy are defined as follows:

### Definition 1 (dot-depth hierarchy)

$$\begin{array}{ll}
 (a) \Sigma_0^L & := \Pi_0^L := \Delta_0^L := \mathcal{DD}_0 := \mathcal{I} \\
 (b) \Sigma_{k+1}^L & := \text{Pol}(\mathcal{DD}_k) \\
 (c) \Pi_{k+1}^L & := \text{Co-Pol}(\mathcal{DD}_k) \\
 (d) \mathcal{DD}_{k+1} & := \text{BPol}(\mathcal{DD}_k) \\
 (e) \Delta_{k+1}^L & := \Sigma_{k+1}^L \cap \Pi_{k+1}^L \\
 (f) \mathcal{SF} & := \bigcup_{i \geq 0} \Sigma_i^L
 \end{array}$$

It should be noted that the dot-depth hierarchy presented here is sometimes known as the Straubing-Thérien hierarchy. Other papers use the term referring to the closely related Cohen-Brzozowski hierarchy which is defined analogously: level 0 of the Cohen-Brzozowski hierarchy consists of the so-called generalized definite languages. We will denote as  $\Sigma_k^B, \Pi_k^B$  and  $\Delta_k^B$  the levels of the Cohen-Brzozowski hierarchy. In this hierarchy, languages are considered as subsets of  $\Sigma^+$  and not  $\Sigma^*$ . Although the two alternative dot-depth hierarchies do not coincide, our results can also be obtained for the levels of the Cohen-Brzozowski hierarchy (see Theorem 4).

Results of Pin and Weil [PW97] show that  $\Sigma_{k+1}^L \cap \Pi_{k+1}^L = \text{UPol}(\mathcal{DD}_k)$  for all  $k \geq 0$  (see also Table 1). Therefore the line (e) in the above definition could be equivalently given in terms of the UPol operator. It is known that the dot-depth hierarchy is infinite, i.e. all classes  $\Sigma_k^L, \Pi_k^L, \mathcal{DD}_k, \Delta_k^L$  for  $k \geq 1$  are all different [St94] (see also Figure 1). The union of all these classes is the class of *starfree* languages  $\mathcal{SF}$ .

Table 1 shows the action of the operators Pol, Co-, B, and UPol on the dot-depth hierarchy. All its entries either follow from definition or are consequences of [PW97].

class/operator	Pol	Co-	B	UPol
$\Sigma_k^L$	$\Sigma_k^L$	$\Pi_k^L$	$\mathcal{DD}_k$	$\Sigma_k^L$
$\Pi_k^L$	$\Sigma_{k+1}^L$	$\Sigma_k^L$	$\mathcal{DD}_k$	$\Delta_{k+1}^L$
$\mathcal{DD}_k$	$\Sigma_{k+1}^L$	$\mathcal{DD}_k$	$\mathcal{DD}_k$	$\Delta_{k+1}^L$
$\Delta_{k+1}^L$	$\Sigma_{k+1}^L$	$\Delta_{k+1}^L$	$\Delta_{k+1}^L$	$\Delta_{k+1}^L$

Table 1: The behaviour of the operators on the dot-depth hierarchy

A *positive \*-variety of languages*  $\mathcal{V}$  is a class of regular languages closed under:

**Positive Boolean operations:** For every alphabet  $\Sigma$  the set  $\Sigma^*\mathcal{V}$  is closed under union and intersection,

**left and right quotients:** For every alphabet  $\Sigma$  and any word  $u$  in  $\Sigma^*$ , if  $L$  is in  $\Sigma^*\mathcal{V}$  then the languages  $u^{-1}L = \{w \mid uw \in L\}$  and  $Lu^{-1} = \{w \mid wu \in L\}$  are also in  $\Sigma^*\mathcal{V}$ ,

**inverse homomorphic images:** If  $\Gamma, \Sigma$  are alphabets and  $h$  is a homomorphism from  $\Gamma^*$  to  $\Sigma^*$  and  $L$  is a language from  $\Sigma^*\mathcal{V}$  then  $h^{-1}(L) = \{x \in \Gamma^* \mid h(x) \in L\}$  is in  $\Sigma_0^*\mathcal{V}$ .

A *\*-variety of languages* is defined in the same way with the additional requirement that the classes  $\Sigma^*\mathcal{V}$  are closed under complement.

Examples of \*-varieties of languages include  $\mathcal{DD}_k$  and  $\Delta_k^L$  for  $k \geq 0$ , and  $\mathcal{SF}$  while  $\Sigma_k^L$  and  $\Pi_k^L$  are positive \*-varieties of languages but are not \*-varieties. The class  $\mathcal{I}$

defined above is called the *trivial* variety and is contained in every positive  $*$ -variety of languages.

We will say that the positive  $*$ -variety  $\mathcal{V}$  *contains the sub-alphabets* if for each alphabet  $\Sigma$  and each  $\Sigma_0 \subseteq \Sigma$ , we have  $\Sigma_0^* \in \Sigma^* \mathcal{V}$ . This is equivalent to having a non-trivial intersection with  $\Pi_1^L$ . For  $*$ -varieties of languages (as a special case of positive  $*$ -varieties of languages) this restriction is equivalent to the requirement that the variety contains at least one language whose syntactic monoid is not a group. In particular, for starfree  $*$ -varieties of languages, the restriction is equivalent to being non-trivial.

### 3. Leaf Languages and the Polynomial Hierarchy

We will call a *complexity class* a subset of the set of languages over the alphabet  $\{0, 1\}$  and use calligraphic letters from the beginning of the alphabet (like  $\mathcal{C}$ ) as variables for complexity classes and from the end of the alphabet (like  $\mathcal{V}$ ) for classes of regular languages. As usual, we denote as  $P$  and  $NP$  the complexity classes of languages computable by, respectively, a deterministic and a non-deterministic Turing machine in polynomial time and as  $PSPACE$  the complexity class of languages computable in polynomial space. For a complexity class  $\mathcal{C}$  let  $T \cdot \mathcal{C}$  be the set of languages computable in deterministic polynomial time by an oracle Turing machine which uses a language from  $\mathcal{C}$  as an oracle<sup>§</sup>. Let  $\exists \cdot \mathcal{C}$  be the set of all languages  $L$  such that

$$L = \{x \mid \text{there exists } y \text{ with } |y| \leq q(|x|) \text{ such that } \langle x, y \rangle \in A\}$$

for some language  $A \in \mathcal{C}$  and some polynomial  $q$ . For any prime  $p$ , we similarly define  $MOD_p \mathcal{C}$  as the set of languages  $L$  such that

$$L = \{x \mid \text{there exists } 0 \pmod{p} y \text{ with } |y| \leq q(|x|) \text{ such that } \langle x, y \rangle \in A\}$$

for some language  $A \in \mathcal{C}$  and some polynomial  $q$ . Let  $co \cdot \mathcal{C}$  be the set of complements of  $\mathcal{C}$  and  $BC \cdot \mathcal{C}$  be the Boolean closure of  $\mathcal{C}$ , i.e. all languages obtainable by a finite number of applications of the Boolean operations union, intersection and complementation, starting with languages from  $\mathcal{C}$ . Using this operator notation one can write for example  $P = T \cdot \emptyset$ ,  $NP = \exists \cdot P$ ,  $co-NP = co \cdot NP$ ,  $BC(NP) = BC \cdot NP$ , and  $\Delta_2^P = T \cdot NP$ . The classes of the polynomial hierarchy are defined as follows [MS72, Pa94], (see Figure 1):

**Definition 2 (polynomial hierarchy)** *Let  $k \geq 1$ .*

- (a)  $\Sigma_0^P := \Pi_0^P := \Delta_0^P := P$
- (b)  $\Sigma_{k+1}^P := \exists \cdot \Pi_k^P$
- (c)  $\Pi_{k+1}^P := co \cdot \Sigma_{k+1}^P$
- (d)  $\Delta_{k+1}^P := T \cdot \Sigma_k^P$
- (e)  $PH := \bigcup_{i \geq 0} \Sigma_{k+1}^P$

---

<sup>§</sup>This class is often denoted  $P(\mathcal{C})$ ,  $P^{\mathcal{C}}$  or  $\leq_T^P(\mathcal{C})$ .

Note that  $\text{NP} = \Sigma_1^p$  and  $\text{co-NP} = \Pi_1^p$ . The following table shows the behaviour of the operators  $\exists \cdot$ ,  $\text{co} \cdot$ ,  $\text{BC} \cdot$ , and  $\text{T} \cdot$  on the classes of the polynomial hierarchy, see. Note the similarity with Table 1: the  $\exists \cdot$  and  $\text{T} \cdot$  operators on the polynomial hierarchy behave exactly like the  $\text{Pol}$  operator and the combined  $\text{UPol} \circ \text{B}$  operator on the dot-depth hierarchy, respectively.

class/operator	$\exists \cdot$	$\text{co} \cdot$	$\text{BC} \cdot$	$\text{T} \cdot$
$\Sigma_k^p$	$\Sigma_k^p$	$\Pi_k^p$	$\text{BC}(\Sigma_k^p)$	$\Delta_{k+1}^p$
$\Pi_k^p$	$\Sigma_{k+1}^p$	$\Sigma_k^p$	$\text{BC}(\Sigma_k^p)$	$\Delta_{k+1}^p$
$\text{BC}(\Sigma_k^p)$	$\Sigma_{k+1}^p$	$\text{BC}(\Sigma_k^p)$	$\text{BC}(\Sigma_k^p)$	$\Delta_{k+1}^p$
$\Delta_{k+1}^p$	$\Sigma_{k+1}^p$	$\Delta_{k+1}^p$	$\Delta_{k+1}^p$	$\Delta_{k+1}^p$

Table 2: The behaviour of the operators on the polynomial hierarchy

It is not known whether the polynomial hierarchy collapses to some finite level or is infinite. Nevertheless, there exists a relativized world (one in which any machine has access to some specific oracle) in which all  $\Sigma_k^p$ ,  $\Pi_k^p$  and  $\Delta_k^p$  classes are different from one another [Yao85]. Moreover, in this world,  $\Delta_k^p$  is different from  $\text{BC}(\Sigma_{k-1}^p)$  for any  $k \geq 2$  since  $\Delta_k^p$  has a polynomial-time many-one complete problem in any relativization whereas, by the relativizable results of Kadin [Ka88], the class  $\text{BC}(\Sigma_{k-1}^p)$  does not. For  $k = 1$  and  $k = 2$  oracles separating  $\Delta_k^p$  from its superset  $\Sigma_k^p \cap \Pi_k^p$  were constructed in [BGS75] and [He84], respectively, but for larger  $k$  the authors could not find a construction in the literature. On the other hand, any PSPACE-complete oracle  $A$  collapses all these classes to P. Figure 1 gives a synoptical view of the dot-depth hierarchy and the polynomial hierarchy.

Let some language  $L$  over some alphabet  $\Sigma$  be given. The *leaf language* approach [BCS92, HL\*93, BKS99] assigns to the language  $L$  a class  $\text{Leaf}^P(L)$  of languages on the alphabet  $\{0, 1\}$  as follows. Let some nondeterministic polynomial-time Turing machine  $N$  be given. Assume that it not only accepts or rejects on every computation path but outputs a letter from the alphabet  $\Sigma$  on each computation path when it terminates.  $N$  produces for every input  $x \in \{0, 1\}^*$  a computation tree (not necessarily balanced) whose paths are ordered in the natural way and whose leaves are labeled by letters from  $\Sigma$ . Therefore the letters on the leaves form a word over the alphabet  $\Sigma$  which we call the yield of the computation tree and denote as  $\text{leafstring}(N, x)$ . For each  $N$  let  $\text{Leaf}^N(L) \in \{0, 1\}^*$  be the set of inputs  $x$  such that  $\text{leafstring}(N, x)$  is in  $L$ , and let  $\text{Leaf}^P(L)$  be the class of languages  $\text{Leaf}^N(L)$  for some  $N$ . As an example, for the language  $S_1 := \{0, 1\}^*1\{0, 1\}^*$  over alphabet  $\{0, 1\}$  we have  $\text{Leaf}^P(S_1) = \text{NP}$ . For a class  $\mathcal{C}$  of languages let  $\text{Leaf}^P(\mathcal{C})$  be the union of the classes  $\text{Leaf}^P(L)$  for some  $L \in \mathcal{C}$ . For any regular language  $L$  or class of regular languages  $\mathcal{C}$ , the classes  $\text{Leaf}^P(L)$  and  $\text{Leaf}^P(\mathcal{C})$  are complexity classes within PSPACE and in fact  $\text{PSPACE} = \text{Leaf}^P(\mathcal{REG})$  where  $\mathcal{REG}$  denotes the class of regular languages [HL\*93]. Many such results relate subclasses of PSPACE with subclasses of  $\mathcal{REG}$  in this way.

#### 4. The Sigma and Pi Levels of the Hierarchies

We show in this section that, under certain technical conditions, the Pol operator on positive varieties of regular languages corresponds via leaf languages to the dot operator  $\exists \cdot$  on complexity classes. This was already mentioned but not proven explicitly in the paper of Hertrampf et al. [HL\*93] and the proof is included for completeness.

Let  $L$  be a language over the alphabet  $\Gamma$  be given. Let  $E_L$  be the language  $\Sigma^*bLb\Sigma^*$  over the alphabet  $\Sigma = \Gamma \cup \{b\}$  where  $b$  is some letter (a *marker*) not in  $\Gamma$ . Consider the homomorphism  $h : \Sigma \rightarrow \Gamma$  which maps  $b$  to the empty word  $\epsilon$  and is the identity on  $\Gamma$ . If  $L$  is in a positive variety  $\mathcal{V}$  which contains the sub-alphabets then  $L$  as a set of words over  $\Sigma$  lies in  $\Sigma^*\mathcal{V}$  because  $L$  as a subset of  $\Sigma^*$  is equal to  $h^{-1}(L) \cap \Gamma^*$ . Hence,  $E_L$  is in  $\text{Pol}(\mathcal{V})$ .

**Lemma 1 ([HL\*93])** *Let  $\mathcal{V}$  be a positive \*-variety of languages which contains the sub-alphabets.*

(a)  $\text{Leaf}^P(\text{Pol}(\mathcal{V})) = \exists \cdot \text{Leaf}^P(\mathcal{V})$ .

(b) *If  $\text{Leaf}(L) = \text{Leaf}(\mathcal{V})$  for a single language  $L \in \mathcal{V}$  then  $\text{Leaf}(E_L)$  equals the two classes from (a).*

**Proof.** Direction (a)  $\subseteq$ : Let  $L = L_0a_1L_1 \cdots a_nL_n$  be a language in  $\Sigma^*\text{Pol}(\mathcal{V})$  where each  $L_i$  is a language from  $\Sigma^*\mathcal{V}$  and all markers  $a_i$  are in  $\Sigma$ . Let  $\Sigma_0 := \Sigma \cup \{0\}$  for a new letter  $0$  not in  $A$ . Note that since  $\mathcal{V}$  is closed under inverse homomorphic images each language  $L_i \sqcup 0^*$  (the language  $L_i$  shuffled with 0's) is also in  $\mathcal{V}$ . We define the language  $K$  over the  $(n+1)$ -fold product  $\Sigma_0 \times \dots \times \Sigma_0$  as a form of direct product of the languages  $(L_i \sqcup 0^*)$ :

$$K = \left\{ \left( \begin{array}{c} a_{10} \\ a_{11} \\ \vdots \\ a_{1n} \end{array} \right) \left( \begin{array}{c} a_{20} \\ a_{21} \\ \vdots \\ a_{2n} \end{array} \right) \cdots \left( \begin{array}{c} a_{s0} \\ a_{s1} \\ \vdots \\ a_{sn} \end{array} \right) \mid \text{for each } i, a_{1i} \dots a_{si} \in L_i \sqcup 0^* \right\}.$$

From the closure properties of  $\mathcal{V}$ , we have  $K \in \mathcal{V}$  since each  $L_i \sqcup 0^*$  is in  $\mathcal{V}$ . Let some nondeterministic polynomial-time Turing machine  $M$  working with leaf language  $L$  be given. An NDTM working with the leaf language  $K$  can be seen as writing its leafstring on  $n+1$  different, independent tracks corresponding to the  $n+1$  copies of the alphabet  $\Sigma_0$ . Let  $M'$  be such an NDTM which, on input  $\langle x, y \rangle$  checks whether  $y$  encodes  $n$  paths  $p_1 < \dots < p_n$  of the computation produced by  $M$  on input  $x$  and first checks if the letters written at these positions by  $M$  are  $a_1, \dots, a_n$ . If this is not the case  $M'$  rejects, i.e. writes a computation tree whose leafstring is not in  $K$  (this word exists because  $\mathcal{V}$  is not trivial since it contains the sub-alphabets). Otherwise, let  $p_0 = 0$  and  $p_{n+1} = \text{max} + 1$  where  $\text{max}$  is the length of the leafstring produced by  $M$ :  $M'$  simulates the computation of  $M$  on input  $x$  and produces as output for the leaf at position  $p$  with  $p_i < p < p_{i+1}$  the tuple-letter  $(0, \dots, 0, b, 0, \dots, 0)^T$  where  $b$  is in the  $i$ -th track of the tuple and is the letter which  $M$  outputs on path  $p$ . On the guessed paths  $p_1 < \dots < p_n$  it produces an all-0 tuple.

The  $i$ -th track of the resulting leafstring is then a word of the form  $0^{n_i}w_i0^{m_i}$  (for some numbers  $n_i, m_i$ ) where  $w_i$  is the subword of the leafstring originally produced by  $M$  on the paths between  $p_i$  and  $p_{i+1}$ . All  $w_i$  belong to  $L_i$  if and only if all words  $0^{n_i}w_i0^{m_i}$  belong to  $L_i \sqcup 0^*$  if and only if the yield of  $M'$  on  $\langle x, y \rangle$  lies in  $K$ . Thus,  $M$  accepts  $x$  if and only if there exists  $y$  for which  $M'$  accepts  $\langle x, y \rangle$ . In general  $L$  could be a finite union of such marked products: in that case  $M'$  further needs to guess which marked product to work with.

Direction (a)  $\supseteq$ : Let  $L$  be a language  $\Sigma^*\mathcal{V}$  and  $R$  be a language of  $\exists \cdot \text{Leaf}^P(\mathcal{V})$ :

$$R = \{x \mid \text{there exists } y \text{ with } |y| \leq q(|x|) \text{ such that } \langle x, y \rangle \in \text{Leaf}^M(L)\}$$

where  $q$  is a polynomial. We will construct a NDTM  $M'$  using  $E_L$  as a leaf language to accept the language  $R$ . On input  $x$  the NDTM  $M'$  produces (say in lexicographic order) for each word  $y$  such that  $|y| \leq q(|x|)$  a separate path for the coded pair  $\langle x, y \rangle$ . On each of these paths it branches into two paths: on the left path it writes the marker  $b$ , and the right path it produces a computation tree by simulating the computation of  $M$  on the coded pair  $\langle x, y \rangle$ . Finally it produces for the whole computation tree a rightmost path with letter  $b$ . Obviously,  $M'$  produces a leafstring from  $E_L$  if and only if there is a  $y$  such that  $M$  produces a leafstring from  $L$  on input  $\langle x, y \rangle$ .

(b) follows by construction in (a).  $\square$

Note that the assumption that  $\mathcal{V}$  contains the sub-alphabets is only needed to show the right to left containment and so  $\text{Leaf}^P(\text{Pol}(\mathcal{V})) \subseteq \exists \cdot \text{Leaf}^P(\mathcal{V})$  holds for any positive  $*$ -variety  $\mathcal{V}$ . We will later need the above lemma to prove Lemma 5. As a byproduct, we can also reprove a theorem due partly to Burtschick and Vollmer [BV98] and partly to Hettrampf et al. [HL\*93]. We define for  $k \geq 1$  *canonical languages*  $S_k$  and  $P_k$  for the classes  $\Sigma_k^L$  and  $\Pi_k^L$  as follows. Let  $P_1$  be the language  $0^*$  over the alphabet  $\{0, 1\}$ . Let  $S_k$  be the complement of  $P_k$  for all  $k \geq 1$ . Let  $P_k$  be  $E_{S_{k-1}}$  with marker  $k$ . For example,  $P_2$  is the set of words over alphabet  $\{0, 1, 2\}$  such that between every two different 2's there exists a 1. It is easy to see that  $S_k$  is an element of  $\Sigma_k^L$  and  $P_k$  is an element of  $\Pi_k^L$ . A simple induction yields:

**Theorem 1 ([BV98, HL\*93])** *Let  $k \geq 1$ .*

(a)  $\text{Leaf}^P(S_k) = \text{Leaf}^P(\Sigma_k^L) = \Sigma_k^P,$

(b)  $\text{Leaf}^P(P_k) = \text{Leaf}^P(\Pi_k^L) = \Pi_k^P.$

(c)  $\text{Leaf}^P(\mathcal{DD}_k) = \text{BC}(\Sigma_k^P).$

(d)  $\text{Leaf}^P(\mathcal{SF}) = \text{PH}$

Note that  $\text{Leaf}^P(\mathcal{SF}) = \text{PH}$  is not expected to have a single leaf language, i.e. a language  $L \in \mathcal{SF}$  such that  $\text{PH} = \text{Leaf}^P(L)$  since, as observed in [BS97], every complexity class  $\text{Leaf}^P(L)$  has a  $\leq_m^P$ -complete language. Similarly,  $\text{BC}(\Sigma_k^P)$  for  $k \geq 1$  is not expected to have a single leaf language.

## 5. The Delta Levels of the Hierarchies

In this section we prove the correspondence of the Delta levels of the dot-depth and the polynomial hierarchies. This follows from Lemma 5 which states that, under certain technical conditions, the unambiguous polynomial closure operator UPol on positive  $*$ -varieties of regular languages and the polynomial-time Turing closure operator on complexity classes correspond via the leaf-language mechanism. We first mention useful properties of the operators  $T \cdot$  and  $BC \cdot$ :

**Proposition 1** *Let  $\mathcal{C}$  be a complexity class and  $\mathcal{V}$  be a positive variety of languages.*

- (a)  $T \cdot T \cdot \mathcal{C} = T \cdot \mathcal{C}$ .
- (b) *If  $\mathcal{C} \subseteq T \cdot \text{Leaf}^P(\mathcal{V})$  then  $BC \cdot \mathcal{C} \subseteq T \cdot \text{Leaf}^P(\mathcal{V})$ ,*
- (c)  $BC \cdot \text{Leaf}^P(\mathcal{V}) \subseteq T \cdot \text{Leaf}^P(\mathcal{V})$ ,
- (d)  $T \cdot BC \cdot \text{Leaf}^P(\mathcal{V}) = T \cdot \text{Leaf}^P(\mathcal{V})$ .

**Proof.** Part (a) is trivial while (c) and (d) follow from (b). Finally, (b) is proven by induction on the structure of the Boolean expression. If the top Boolean operation is negation, it suffices to flip the oracle answer. It remains to show that if  $L_1 \in T \cdot \text{Leaf}^P(K_1)$  and  $L_2 \in T \cdot \text{Leaf}^P(K_2)$  with  $K_1, K_2 \in \mathcal{V}$  then we can find some  $K \in \mathcal{V}$  such that  $L_1 \cap L_2 \in T \cdot \text{Leaf}^P(K)$ . For instance, we can use a construction similar to that in Lemma 1: if  $K_1, K_2$  are languages in  $\Sigma_1$  and  $\Sigma_2$  respectively, we choose  $K$  as a subset of  $(\Sigma_1 \cup \{0\}) \times (\Sigma_2 \cup \{0\})$  with  $K = \{(a_{11}, a_{21}) \dots (a_{n1}, a_{n2}) : a_{1i} \dots a_{ni} \in K_i\}$ . Then we have  $K \in \mathcal{V}$  and clearly  $\text{Leaf}^P(K_i) \subseteq \text{Leaf}^P(K)$ .  $\square$

A useful decomposition of the UPol operator was given by Straubing, Pin and Thérien: for a variety of languages  $\mathcal{V}$  let  $\ell_1 \square \mathcal{V}$  be the Boolean closure of languages<sup>¶</sup> of *unambiguous* products  $L_0 a L_1$  where  $L_0, L_1$  are in  $\Sigma^* \mathcal{V}$  and  $a \in \Sigma$ . By [PST88],  $\ell_1 \square \mathcal{V}$  is itself a variety of languages and this operator characterizes UPol( $\mathcal{V}$ ):

**Theorem 2** ([PST88]) *Let  $\mathcal{V}$  be a variety of languages. UPol( $\mathcal{V}$ ) is the class of languages obtained from  $\mathcal{V}$  by finitely many applications of the operator  $\ell_1 \square$ .*

We first show that one application of the operator  $\ell_1 \square$  can basically be simulated by a polynomial-time Turing reduction.

**Lemma 2** *Let  $\mathcal{V}$  be a variety of languages:*

$$\text{Leaf}^P(\ell_1 \square \mathcal{V}) \subseteq T \cdot \text{Leaf}^P(\mathcal{V}).$$

**Proof.** We recall the notion of syntactic congruence of a language  $L \subseteq \Sigma^*$ : Let  $x \equiv_L y$  if for any  $u, v \in \Sigma^*$  we have  $uxv \in L$  iff  $uyv \in L$ . It is well-known that the syntactic congruence of  $L$  has finite index if and only if  $L$  is regular. Assume now that  $L_0 a L_1$  is an unambiguous concatenation with  $L_0, L_1$  in  $\Sigma^* \mathcal{V}$ .

For a word  $x \in \Sigma^*$ , let  $[x]_{L_0}$  and  $[x]_{L_1}$  denote the  $\equiv_{L_0}$  and  $\equiv_{L_1}$  equivalence classes of  $x$  respectively. Following [PST88], we let  $G$  be the graph containing:

---

<sup>¶</sup>The symbols  $\square$  and  $\ell_1$  actually have a meaning as block product and the set of locally trivial categories, respectively, see [PST88, Pi97].

- For any  $x, y \in \Sigma^*$ , a vertex labeled with the pair  $([x]_{L_0}, [y]_{L_1})$ : since  $L_0$  and  $L_1$  are regular, there are only finitely many vertices.

- For any letter  $b \in \Sigma$  edges from  $([x]_{L_0}, [by]_{L_1})$  to  $([xb]_{L_0}, [y]_{L_1})$ . We label these edges with triples  $\langle [x]_{L_0}, b, [y]_{L_1} \rangle$ .

Note that every  $x \in \Sigma^*$  traces a path in this graph from vertex  $([\epsilon]_{L_0}, [x]_{L_1})$  to vertex  $([x]_{L_0}, [\epsilon]_{L_1})$ . Furthermore, if  $x = x_0bx_1$  where  $b$  is the  $j^{\text{th}}$  letter of  $b$  then the  $j^{\text{th}}$  edge of this path is  $\langle [x_0]_{L_0}, b, [x_1]_{L_1} \rangle$ . Call an edge *critical* if it is of the form  $\langle [x_0]_{L_0}, a, [x_1]_{L_1} \rangle$  with  $x_0 \in L_0$  and  $x_1 \in L_1$ : we have  $x \in L_0aL_1$  if and only if the path traced out by  $x$  in  $G$  contains such a critical edge. Because  $L_0aL_1$  is unambiguous however, any path in  $G$  contains at most one critical edge (as first observed by [PST88]). In particular, if  $\langle [x]_{L_0}, b, [y]_{L_1} \rangle$  is some edge of  $G$  then either no path with a critical edge ends at vertex  $([x]_{L_0}, [by]_{L_1})$  or no path with a critical edge starts at vertex  $([xb]_{L_0}, [y]_{L_1})$ .

Let  $N$  be a polynomial-time NDTM: we exhibit a polynomial-time algorithm  $D$  which, using queries to a  $\text{Leaf}^P(\mathcal{V})$  oracle, can test whether the leafstring  $x$  produced by  $N$  on a given input  $t$  lies in  $L_0aL_1$ . The algorithm will check if the path traced by  $x$  in  $G$  contains a critical edge. Suppose for simplicity that on any input  $t$  the computation tree of  $N$  is complete and of depth  $q(|t|)$  for some polynomial  $q$ . Thus,  $x$  has length  $2^{q(|t|)}$ . For any  $1 \leq k \leq 2^{q(|t|)}$  let  $b_k$  denote the  $k^{\text{th}}$  letter in  $x$  and  $p_{k-1}$  and  $s_{k+1}$  the words such that  $x = p_{k-1}b_k s_{k+1}$ .

**Lemma 3** *For any polynomial-time NDTM  $N$  and any  $L_0, L_1 \in \Sigma^*\mathcal{V}$  there exists a polynomial-time DTM querying a  $\text{Leaf}^P(\mathcal{V})$  oracle which given  $k \leq 2^{q(|t|)}$  computes the triple  $\langle [p_{k-1}]_{L_0}, b_k, [s_{k+1}]_{L_1} \rangle$  defined above.*

**Proof.** For a language  $L \in \mathcal{V}$ , let  $U_L$  denote the universal NDTM for  $\text{Leaf}^P(L)$  that is the NTDM which on input  $\langle y, M, 0^k \rangle$  simulates the NDTM  $M$  on input  $y$  for  $k$  steps, outputs the letter  $a$  at any leaf at which  $M$  output an  $a$  within the time-bound and accepts if the resulting leafstring lies in  $L$ . Now, let  $M_k$  be the NDTM which works like  $N$  but has no leaf with index greater than  $k$ . Then the triple  $\langle t, M_{k-1}, 0^{2^{q(|t|)}} \rangle$  is accepted by  $U_L$  if and only if  $p_{k-1}$  lies in  $L$ .

By definition of the syntactic congruence computing  $[p_{k-1}]_{L_0}$  amounts to checking, for some finite collection of pairs of words  $(u_i, v_i)$  whether or not  $u_i p_{k-1} v_i$  lies in  $L_0$  or, equivalently, whether  $p_{k-1} \in u_i^{-1} L_0 v_i^{-1}$ . The latter language also lies in  $\mathcal{V}$  and thus  $U_{u_i^{-1} L_0 v_i^{-1}}$  is indeed a  $\text{Leaf}^P(\mathcal{V})$  oracle.

We can therefore compute both  $[p_{k-1}]_{L_0}$  and  $[s_{k+1}]_{L_1}$  by using a finite number of distinct oracles in  $\text{Leaf}^P(\mathcal{V})$ . We need to combine these into a single oracle and this can be done using the methods of Lemma 1.  $\square$

To check if the path traced out by  $x$  contains a critical edge, our algorithm uses binary search to zoom in on a segment  $r$  of the path where that edge might lie if it exists at all. Initially, we set  $x = r$  (note that we of course cannot store  $r$  itself but we only need a pointer to its first and last letters). Now let  $k \leq 2^{q(|t|)}$  be the middle position of the segment  $r$ . By Lemma 3, we can use queries to an oracle in  $\text{Leaf}^P(\mathcal{V})$  to compute  $\langle [p_{k-1}]_{L_0}, b_k, [s_{k+1}]_{L_1} \rangle$ : If this edge is critical then we know

that  $x$  is in  $L_0aL_1$  and we can stop. Otherwise, we consider two cases:

- if no path with a critical edge ends at vertex  $([p_{k-1}]_{L_0}, [b_k s_{k+1}]_{L_1})$  then any critical edge in the path traced out by  $x$  must occur at some later  $k > 2^{q(|t|)-1}$  and we continue our search on the rightmost half of  $r$ .

- symmetrically if no path with a critical edge starts from  $([p_{k-1}b_k]_{L_0}, [s_{k+1}]_{L_1})$  we must continue our search for a critical edge in the first half of  $r$ .

As we noted above the unambiguity of  $L_0aL_1$  ensures that at least one of the above holds for any  $k$ . If *both* of the above hold then  $x$  cannot lie in  $L_0aL_1$  and the algorithm rejects. After  $q(|t|)$  steps of this binary search the segment  $r$  in which the critical edge might lie consists of a single edge and we can check if it is indeed critical and reject if it is not. So we have shown  $\text{Leaf}^P(\mathcal{U}) \subseteq \mathbb{T} \cdot \text{Leaf}^P(\mathcal{V})$  for the set  $\mathcal{U}$  of unambiguous concatenations of the form  $L = L_0aL_1$  with  $L_0, L_1 \in \mathcal{V}$ . Therefore:

$$\text{Leaf}^P(\ell_1 \square \mathcal{V}) = \text{Leaf}^P(\text{BU}) \subseteq \text{BC} \cdot \text{Leaf}^P(\mathcal{U}) \subseteq \mathbb{T} \cdot \text{Leaf}^P(\mathcal{V})$$

by Proposition 1 (b). □

**Corollary 1** *Let  $\mathcal{V}$  be a \*-variety of languages.*

$$\text{Leaf}^P(\text{UPol}(\mathcal{V})) \subseteq \mathbb{T} \cdot \text{Leaf}^P(\mathcal{V}).$$

**Proof.** By Theorem 2,  $\text{UPol}(\mathcal{V})$  is generated by finitely many applications of the operator  $\ell_1 \square$  starting with  $\mathcal{V}$ . For the languages from  $\mathcal{V}$  the statement is of course true, and for any finite number of applications of the operator the statement follows from Lemma 2 and from the idempotency of the Turing closure operator  $\mathbb{T} \cdot$  stated in Proposition 1(a). □

We do not know whether the reverse inclusion of this corollary holds in general, i.e. whether the  $\text{UPol}$  operator on varieties corresponds to the Turing reducibility closure on the polynomial-time degrees. Nevertheless, Lemma 5 below shows that, under certain conditions, a correspondence holds at least for the combined operator  $\text{UPol} \circ \text{BPol}$  on language classes and the combined operator  $\mathbb{T} \cdot \exists \cdot$  on complexity classes.

For a language  $L$  over the alphabet  $\Sigma_0$  let  $W_L$  be the language over the alphabet  $\Sigma = \Sigma_0 \cup \{a, b\}$  (where  $a, b$  are two new “marker” symbols) consisting of words  $w_1 m_1 w_2 m_2 \cdots w_n m_n w_{n+1}$  such that the  $m_i$  are markers, the  $w_i$  are words in  $\Sigma_0^*$ , and

- there exists  $i \leq n$  such that  $w_i \in L$ .
- if  $i_{\min}$  is the smallest  $i$  with  $w_i \in L$ , then  $m_{i_{\min}} = a$ .

This definition generalizes an original idea of Wagner [Wa90].

**Lemma 4** *Let  $L$  be a language from a positive \*-variety of languages  $\mathcal{V}$  which contains the sub-alphabets. Then  $W_L$  is a language from  $\text{UPol}(\text{BPol}(\mathcal{V}))$ .*

**Proof.** Let  $L \in \mathcal{V}$  over alphabet  $\Sigma_0$  be given, and let  $L_0$  be  $L$  considered as a language over the larger alphabet  $\Sigma = \Sigma_0 \cup \{a, b\}$ . Because  $\mathcal{V}$  contains the sub-alphabets the language  $L_0$  is also in  $\mathcal{V}$  (this is the only place where we need this

property of  $\mathcal{V}$ ). Let  $K$  be the language defined as

$$K = \Sigma^*\{a, b\}L_0\{a, b\}\Sigma^* \cup \Sigma^*\{a, b\}L_0 \cup L_0.$$

We have  $K$  in  $\text{Pol}(\mathcal{V})$  and therefore its complement  $\overline{K}$  is from  $\text{BPol}(\mathcal{V})$ . Now we can write

$$W_L = \overline{K}\{a, b\}L_0a\Sigma^* \cup L_0a\Sigma^*$$

This is the disjoint union of two unambiguous marked products: In the first expression the two positions of the  $a$  (or  $b$ ) and  $a$  around the  $L_0$  are – if they exist – uniquely determined because a word of  $\overline{K}\{a, b\}$  cannot contain a segment that lies in  $\{a, b\}L_0a$ , and in the second expression the position of the  $a$  is, as the first marker occurring, uniquely determined. Thus  $W_L$  lies in  $\text{UPol}(\text{BPol}(\mathcal{V}))$ .  $\square$

We say that a language  $A \in \{0, 1\}^*$  is *polynomial-time conjunctively reducible* to a language  $B$  if there is a polynomial-time oracle Turing machine for  $A$  which on input  $x$  makes queries to an oracle for  $B$  and accepts if and only if all questions are answered positively. Note that the questions can be assumed to be non-adaptive, i.e. the oracle Turing machine produces a (polynomially long) list of questions which are asked all at once. A complexity class  $\mathcal{C}$  is *closed under polynomial-time conjunctive reductions* if every language which is polynomial-time conjunctively reducible to a language in  $\mathcal{C}$  is also in  $\mathcal{C}$ . Of course every complexity class closed under polynomial-time Turing reductions is also closed under polynomial-time conjunctive reductions. Therefore, the  $\Delta_k^p$  classes have this property. Other classes have this property and in particular  $\Sigma_k^p$  and  $\Pi_k^p$ .

**Lemma 5** *Let  $\mathcal{V}$  be a  $*$ -variety of languages which contains the sub-alphabets. If  $\text{Leaf}^P(\mathcal{V})$  is closed under polynomial-time conjunctive reductions, then*

$$(a) \text{Leaf}^P(\text{UPol}(\text{BPol}(\mathcal{V}))) = \text{T} \cdot \exists \cdot \text{Leaf}^P(\mathcal{V}),$$

(b) *If  $L \in \mathcal{V}$  is a single language such that  $\text{Leaf}^P(L) = \text{Leaf}^P(\mathcal{V})$  then  $\text{Leaf}^P(W_L)$  equals the classes of (a).*

**Proof.**

The direction  $\subseteq$  follows immediately from previous lemmata and does not in fact require the extra technical assumptions:

$$\begin{aligned} \text{Leaf}^P(\text{UPol}(\text{BPol}(\mathcal{V}))) &\subseteq \text{T} \cdot \text{Leaf}^P(\text{BPol}(\mathcal{V})) \text{ (Lemma 1)} \\ &\subseteq \text{T} \cdot \text{BC} \cdot \text{Leaf}^P(\text{Pol}(\mathcal{V})) \\ &= \text{T} \cdot \text{BC} \cdot \exists \cdot \text{Leaf}^P(\mathcal{V}) \text{ (Lemma 1)} \\ &= \text{T} \cdot \exists \cdot \text{Leaf}^P(\mathcal{V}) \text{ (Proposition 1(d)).} \end{aligned}$$

The proof of the other direction generalizes an idea of Wagner [Wa90] whose result can be interpreted as showing that  $\Delta_2^p \in \text{Leaf}^P(0^*a\{0, a, b\}^*)$ . Since this fact will later be needed in the proof of Theorem 3, we sketch its proof. Note also that the language  $0^*a\{0, a, b\}^*$  is  $W_L$  for  $L = 0^*$ . We need to show that for any deterministic polynomial time machine  $D$  querying, say, a SAT oracle there exists a non-deterministic NDTM, say  $N$ , which uses the leaf language  $0^*a\{0, a, b\}^*$  and

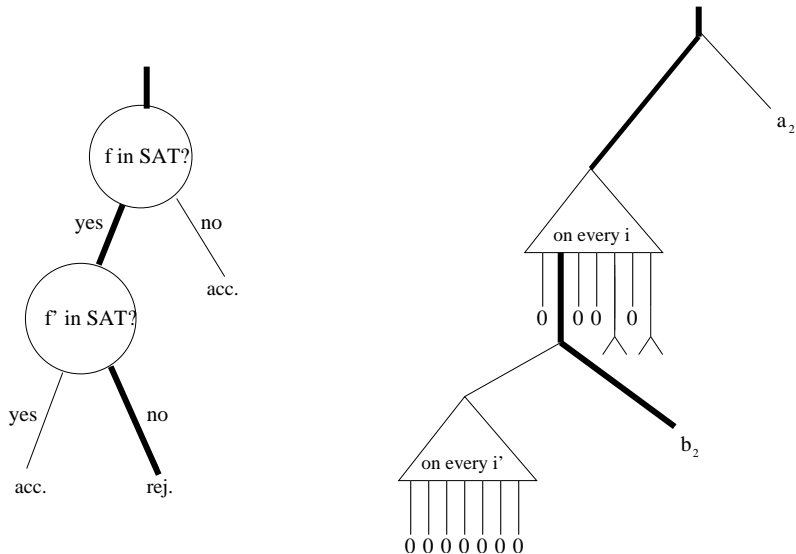


Figure 2: Building a computation tree for the leaf language  $0^*a\{0, a, b\}^*$

simulates  $D$ . First,  $N$  begins by simulating the deterministic behaviour of  $D$  until  $D$  asks a first query  $Q$ . At this point  $N$  simulates the query by non-deterministically branching in two computations (see Figure 2):

- On the left branch  $N$  attempts to verify that the oracle answers positively to the question  $Q$ : it produces a path for each possible witness of membership of  $Q$  in SAT. On every such path, it first checks whether the candidate-witness is correct. If it is not,  $N$  terminates on this computation path, writing a 0 on this leaf, otherwise,  $N$  resumes the deterministic simulation of  $D$  assuming a positive answer to the query  $Q$ .
- On the right branch  $N$  continues the deterministic simulation of  $D$  assuming a negative answer to the query  $Q$ .

We proceed in the same fashion for each query of  $D$ . When the simulation of  $D$  is complete,  $N$  terminates and writes an  $a$  on the leaf if  $D$  accepts, and a  $b$  otherwise. The key observation is that the leftmost path of  $N$  with a non-0 on its leaf corresponds to a correct simulation of  $D$  because if a query  $Q$  is answered negatively, all candidate witnesses are rejected and the left subtree thus created has all its leaves labeled with 0.

Therefore the first non-0 in the leafstring is an  $a$  if and only if  $D$  accepts its input  $x$ . In other words,  $\text{leafstring}(N, x) \in 0^*a\{0, a, b\}^*$  iff  $x$  is accepted by  $D$ . This shows  $\Delta_2^p = \text{P}^{\text{SAT}} \subseteq \text{Leaf}^{\text{P}}(0^*a\{0, a, b\}^*)$ .

More generally, we now want to show that a deterministic polynomial-time  $D$  querying an oracle of  $\exists \cdot K$  where  $K$  is from  $\text{Leaf}^{\text{P}}(L_0)$  with  $L_0 \in \mathcal{V}$  can be simulated

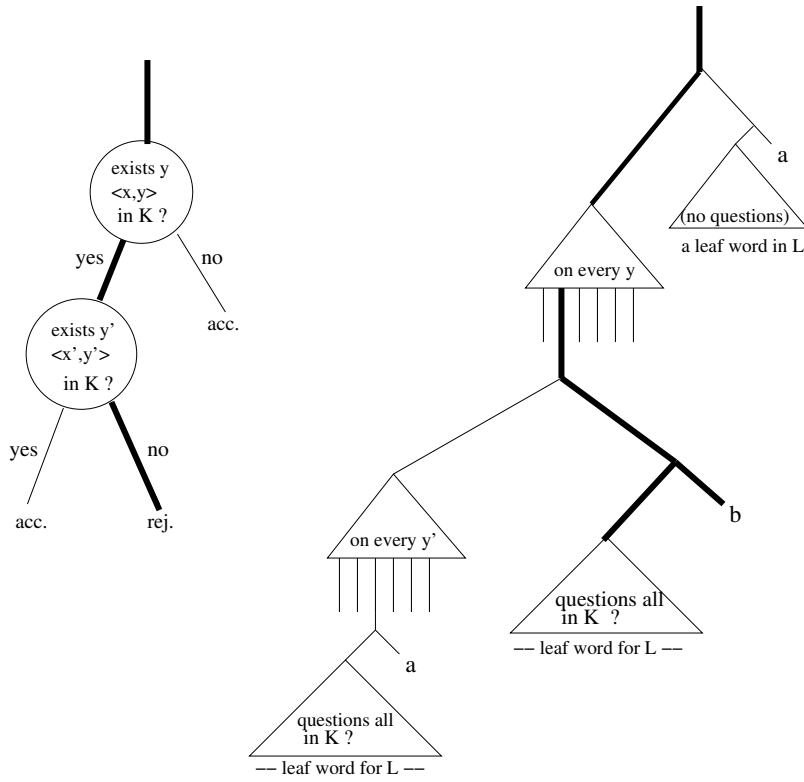


Figure 3: Building a computation tree for the leaf language  $W_L$

by a  $\text{Leaf}^P(H)$  machine  $N$  with  $H \in \text{Upol}(\text{Bpol}(\mathcal{V}))$ . Because  $\text{Leaf}^P(\mathcal{V})$  is closed under polynomial-time conjunctive reductions there exists a leaf language  $L$  in  $\mathcal{V}$  and a NDTM  $M$  such that on an input  $\langle x_1, \dots, x_n \rangle$   $M$  produces a leafstring from  $L$  if and only if  $M_0$  produces on every input  $x_i$  a leafstring from  $L_0$ . We will show  $\text{T} \cdot \exists \cdot \text{Leaf}^P(L_0) \subseteq \text{Leaf}^P(W_L)$  for the language  $W_L$  with markers  $a, b$ .

We proceed as before: whenever  $D$  queries the oracle  $\exists \cdot K$  with question  $t$ ,  $N$  branches into two computations:

- On the left branch,  $N$  produces a path for each possible witness  $w_i$  of membership of  $t$ . However, instead of immediately verifying that the pair  $\langle t, w_i \rangle$  lies in  $K$ , our simulation simply assumes this hypothesis, postpones its check and resumes the simulation of  $D$  assuming a positive answer to the query.
- On the right branch,  $N$  continues the simulation of  $D$  assuming a negative answer to the query.

Once a branch of  $N$ 's simulation of  $D$  terminates, we need to check that all input/witness pairs along that branch *do* lie in  $K$ . For this,  $N$  again branches into two paths:

- On the left path it produces a computation tree whose leafstring lies in  $L$  if and only if all input/witness pairs on that branch lie in  $K$ . This can be done because we assumed the closure under conjunctive reductions of  $\text{Leaf}^P(\mathcal{V})$ .
- On the right branch it writes a marker  $a$  if  $D$  has accepted on this path of assumed oracle answers, and a marker  $b$  in the other case.

Once again, if we consider the leafstring of  $N$ , we observe that the first subword  $w_i$  sitting between two markers and lying in  $L$  indicates the valid computation of  $D$ . Accordingly, the marker which lies at the right of  $w_i$  is  $a$  if and only if  $D$  accepts its input and this is precisely what the constructed language  $W_L$  guarantees.

This shows  $T \cdot \exists \cdot \text{Leaf}^P(L_0) \subseteq \text{Leaf}^P(W_L)$  and so, by Lemma 4 we conclude  $T \cdot \exists \cdot \text{Leaf}^P(L_0) \subseteq \text{Leaf}^P(\text{UPol}(\text{BPol}(\mathcal{V})))$ .

(b) follows from the construction in part  $\supseteq$  above.  $\square$

We define a sequence  $D_i$  of languages for  $i \geq 2$  as follows. First  $D_2 = 0^*a\{0, a, b\}^*$ , the language we used in the previous lemma. For  $k \geq 3$ , we inductively define  $D_{k+1} = W_{D_k}$ . Note that  $D_2$  lies in  $\Delta_2^L$  and so we have  $D_k \in \Delta_k$  using Lemma 4.

**Theorem 3 (Main)** *For every  $k \geq 2$ :*

$$\text{Leaf}^P(D_k) = \text{Leaf}^P(\Delta_k^L) = \Delta_k^p.$$

**Proof.** We establish  $\text{Leaf}^P(D_k) = \text{Leaf}^P(\Delta_k^L) = \Delta_k^p$  by induction on  $k$ . For  $k = 2$ , we have already shown in the previous proof that  $\Delta_2^p \subseteq \text{Leaf}^P(D_2) \subseteq \text{Leaf}(\Delta_2^L)$ . Furthermore, by Corollary 1 we get

$$\text{Leaf}^P(\Delta_2^L) \subseteq T \cdot \text{Leaf}^P(\mathcal{DD}_1) \subseteq T \cdot BC \cdot \Sigma_1^p = \Delta_2^p.$$

For the induction step, we get:

$$\begin{aligned} \text{Leaf}^P(\Delta_{k+1}^L) &= \text{Leaf}^P(\text{Upol}(\text{BPol}(\Delta_k^L))) \text{ (See Figure 1)} \\ &= T \cdot \exists \cdot \text{Leaf}^P(\Delta_k^L) \text{ (Lemma 5)} \\ &= T \cdot \exists \cdot \Delta_k^p = \Delta_{k+1}^p \text{ (from the induction hypothesis)} \end{aligned}$$

Note that we can indeed apply Lemma 5: by induction we have  $\text{Leaf}^P(\Delta_k^L) = \Delta_k^p$  which is closed under conjunctive reductions. We also get from Lemma 5 that  $\text{Leaf}^P(\Delta_{k+1}^L) = \text{Leaf}^P(W_{D_k}) = \text{Leaf}^P(D_{k+1})$ .  $\square$

Note that for each of the languages  $D_i$ , the letter 0 is *neutral* i.e. can be deleted or added at will in a word without affecting membership in  $D_i$ . Thus, the above result still holds when we further require as in e.g. [BCS92, BS97, HL\*93] that the leaf-language machines produce balanced computation trees. Meyer and Stockmeyer made the somewhat uncanonical choice of defining  $\Delta_{k+1}^p$  as  $T \cdot \Sigma_k^p$  rather than  $\Pi_{k+1}^p \cap \Sigma_{k+1}^p$ , probably because this makes  $\Delta_{k+1}^p$  a “syntactic” class [Pa94]. This has the advantage of allowing a nice completion of the correspondence between the dot-depth and polynomial hierarchies, in the line of Theorem 1.

The proof of Lemma 5 relativizes and so the equality of Theorem 3 holds relative to any oracle. So there exist oracles relative to which  $\text{Leaf}^P(\mathcal{DD}_{k-1}) = \text{BC}(\Sigma_{k-1}^p)$  is

strictly contained in  $\text{Leaf}^P(D_k) = \Delta_k^p$ . This shows that the  $D_k$  are a rare occurrence of languages lying in  $\Delta_k^L - \mathcal{DD}_{k-1}$ .

To obtain an analog of Theorem 3 for the levels of the Cohen-Brzozowski hierarchy, we use a result of Pin which forms a bridge between the two versions of the dot-depth hierarchy. For a word  $w = a_1 \dots a_n$  in  $\Sigma^*$  and an integer  $d \geq 1$ , let  $w^d$  denote the  $d$ -decomposition of  $w$ , that is the word of  $\Sigma^d$  obtained by moving a window of size  $d$  over  $w$ . More formally,  $w^d$  is  $w$  if  $d > n$  and otherwise

$$w^d = (a_1, \dots, a_d)(a_{d+1}, \dots, a_{2d}) \dots (a_{n-d+1}, \dots, a_n).$$

Similarly, for a language  $L \in \Sigma^*$ , let  $L^d = \{w^d | w \in L\}$ . Pin showed that  $L \in \Delta_k^B$  iff it is a finite union of languages of the form  $xKy$  for some finite words  $x, y$  and  $J^d$  for some  $d \geq 1$  where the languages  $J, K$  are in  $\Delta_k^L$  [Pi98]. In other words, this sliding window of size  $d$  precisely characterizes the nuance between the two hierarchies. We can now easily show

**Theorem 4** *For every  $k \geq 1$ ,  $\text{Leaf}^P(\Delta_k^B) = \Delta_k^p$ .*

**Proof.** From the definition of dot-depth one can see that for any language  $L \subseteq \Sigma^*$  lying in  $\Delta_k^L$ , we have  $L \cap \Sigma^+$  in  $\Delta_k^B$  so we clearly have  $\Delta_k^p = \text{Leaf}^P(\Delta_k^L) \subseteq \text{Leaf}^P(\Delta_k^B)$ . For the converse, we can easily transform a machine  $N$  yielding a leafstring  $w$  into a machine  $M$  that yields  $w^d$  for some fixed  $d$ . This allows us to transform a machine working with a leaf-language in  $\Delta_k^B$  into one that works with a language in  $\Delta_k^L$ .  $\square$

## 6. Leaf-Language Upper Bounds for BPP

Lautemann, improving on a result of Sipser, showed that the class BPP is contained in the intersection of  $\Sigma_2^p$  and  $\Pi_2^p$  (see e.g. [Pa94]) and this containment holds relative to any oracle. Hence, BPP is contained in  $\text{Leaf}^P(\Sigma_2^L) \cap \text{Leaf}^P(\Pi_2^L)$  and it is natural to ask whether it is contained in  $\text{Leaf}^P(\Sigma_2^L \cap \Pi_2^L) = \text{Leaf}^P(\Delta_2^L)$ . This cannot be ruled out – and many now believe that in fact  $\text{BPP} = \text{P}$  – but there are oracles relative to which  $\Delta_2^p$  is strictly contained in BPP (see e.g. [BT00]). One can also try to identify the positive varieties of languages  $\mathcal{V}$  such that  $\text{BPP} \subseteq \text{Leaf}^P(\mathcal{V})$  holds relative to any oracle.

In [ST03], Straubing and Thérien showed that the class of regular languages which can be defined by two-variable sentences using ordinary and modular quantifiers is exactly the class  $\mathcal{DA} * \mathcal{G}_{sol} = \text{UPol}(\text{BPol}(\mathcal{G}_{sol}))$  where  $\mathcal{G}_{sol}$  denotes the class of languages whose syntactic monoid is a solvable group. They noted that some of their methods were reminiscent of the proof of Toda's Theorem, leading them to conjecture that BPP sits in  $\text{Leaf}^P(\text{UPol}(\text{BPol}(\mathcal{G}_{sol})))$ . We cannot directly use our result of Lemma 5 to study this class of languages since the variety of languages  $\mathcal{G}_{sol}$  does not contain the subalphabets but the proof can be adapted to show:

**Theorem 5** *If  $\mathcal{H}$  is a  $*$ -variety of group languages such that  $\text{Leaf}^P(\mathcal{V})$  is closed under polynomial-time conjunctive reductions, then*

$$\text{Leaf}^P(\text{UPol}(\text{BPol}(\mathcal{V}))) = \text{T} \cdot \exists \cdot \text{Leaf}^P(\mathcal{V}).$$

**Proof.** As in Lemma 5, the left to right inclusion follows from our remarks following Lemma 1 and Corollary 1. To get the right to left inclusion, we were previously using the language  $W_L$  for some  $L \in \mathcal{H}$  but if  $\mathcal{H}$  does not contain the subalphabets, we cannot guarantee that this language lies in  $\text{Upol}(\text{BPol}(\mathcal{H}))$ . We choose the following closely related alternative: for  $L \in \Sigma_0^* \mathcal{H}$ , let  $U_L$  be the language over the larger alphabet  $\Sigma_0 \cup \{a, b\}$  consisting of words  $w = w_1 m_1 w_2 \dots m_n w_n$  where the  $m_i$  are markers  $a$  or  $b$  and the  $w_i$  words of  $\Sigma_0^*$  such that

- there exists  $i \leq n$  such that  $w_1 m_1 \dots m_{i-1} w_i \in L \sqcup \{a, b\}^*$ ;
- if  $i_{\min}$  is the smallest such  $i$  then  $m_{i_{\min}}$  then  $i_{\min} = a$ .

We claim  $U_L \in \text{Upol}(\text{BPol}(\mathcal{H}))$ . Indeed, let  $L' = L \sqcup \{a, b\}^*$ . If  $h : \Sigma \rightarrow \Sigma_0$  is the homomorphism which maps  $a$  and  $b$  to  $\epsilon$  and is the identity on  $\Sigma^0$  we have  $L' = h^{-1}(L)$  so  $L \in \Sigma \mathcal{H}$ . Therefore, the language  $K = \Sigma^* \{a, b\} L' \{a, b\} \Sigma^* \cup \Sigma^* \{a, b\} L' \cup L'$  is in  $\text{Pol}(\mathcal{H})$  and  $U_L = \bar{K} \{a, b\} L' a \Sigma^* \cup L' a \Sigma^*$  is an unambiguous product as we argued in Lemma 4.

Now, we need to simulate a polynomial-time deterministic machine  $D$  querying an oracle of the form  $\exists \cdot K$  with  $K \in \text{Leaf}^P(L_0)$  and  $L_0 \in \mathcal{H}$ . Again, since  $\text{Leaf}^P(\mathcal{H})$  is closed under polynomial-time conjunctive reductions, there is a language  $L \in \mathcal{H}$  and a NDTM  $M$  which on input  $\langle x_1, \dots, x_n \rangle$  produces a leafstring from  $L$  iff each  $x_i$  belongs to  $K$ . We will show that  $D$  can be simulated by an NDTM  $N$ , using the leaf-language  $U_L$ .

If the syntactic monoid of  $L$  is a group of order  $q$  then for any word  $u$ , we have  $u^q \equiv_L \epsilon$  in the syntactic congruence of  $L$ . In the construction of Lemma 5, the leafstring produced by  $N$  was of the form  $w = w_1 m_1 \dots w_n m_n$  with  $w_i \in \Sigma_0^*$  and  $m_i \in \{a, b\}$  and all we needed is to find the leftmost  $m_i$  with  $w_i$  lying in  $L$  to decide whether or not  $w$  lies in  $W_L$ . We modify our construction so that the leafstring produced by  $N$  has the form  $w_1 m_1 (w_1)^{q-1} w_2 m_2 (w_2)^{q-1} \dots w_n m_n (w_n)^{q-1}$ . In other words, once a branch of  $N$ 's simulation of  $D$  terminates it produces first a subtree whose leafstring lies in  $L$  if all the input-witness pairs on the corresponding branch lie in  $K$ , then a marker  $a$  or  $b$  depending on whether  $D$  has accepted or rejected on this path of assumed oracle answers and, finally, repeats  $q - 1$  copies of the first subtree. Now, the prefix  $w_1 m_1 (w_1)^{q-1} \dots w_{i-1} m_{i-1} (w_{i-1})^{q-1} w_i$  lying before the  $i^{\text{th}}$  marker lies in  $L \sqcup \{a, b\}^*$  iff  $w_i \in L$  because we have  $w_1^q \dots w_{i-1}^q \equiv_L \epsilon$ . Hence, the leafstring belongs to  $U_L$  if and only if  $D$  accepts its input.  $\square$

It is known that for any prime  $p$ , we have  $\text{Leaf}^P(\mathcal{G}_p) = \text{MOD}_p \text{P}$  where  $\mathcal{G}_p$  are the languages whose syntactic monoid is a  $p$ -group and  $\text{Leaf}^P(\mathcal{G}_{\text{sol}}) = \text{MOD}^* \text{P}$  where  $\text{MOD}^* \text{P}$  denotes the closure of  $\text{P}$  under the  $\text{Mod}_q$  operators [HL\*93]. Since these classes are closed under conjunctive reductions, we have

$$\text{Leaf}^P(\mathcal{DA} * \mathcal{G}_{\text{sol}}) = T \cdot \exists \cdot \text{Leaf}^P(\mathcal{G}_{\text{sol}}) = T \cdot \exists \cdot \text{MOD}^* \text{P}$$

and

$$\text{Leaf}^P(\text{UPol}(\text{BPol}(\mathcal{G}_q))) = T \cdot \exists \cdot \text{Leaf}^P(\mathcal{G}_q) = T \cdot \exists \cdot \text{MOD}_q \text{P}.$$

There exist relativized worlds in which  $T \cdot \exists \cdot \text{MOD}_2 \text{P}$  does not contain BPP [BT00] and it is perhaps possible to further show that in this world even  $T \cdot \exists \cdot \text{MOD}^* \text{P}$

does not contain BPP. Such a result would rule out any relativizable proof of the aforementioned conjecture of Straubing and Thérien.

## 7. Conclusion and Open Problems

We have shown that the  $\Delta_k$  classes of the dot-depth hierarchy and of the polynomial hierarchy correspond via leaf-languages and this result also holds if we consider the Cohen-Brzozowski definition of the dot-depth hierarchy. Furthermore, we have obtained general results relating the polynomial closure and unambiguous closure operators on  $*$ -varieties of languages with the  $\exists$ - and  $T$ -operators on complexity classes respectively.

For all varieties considered in this paper,  $\text{Leaf}^P(\mathcal{V})$  is closed under polynomial-time conjunctive reductions and this can perhaps be concluded simply using the closure properties of the varieties. This would make Lemma 5 “cleaner” because it would shift the technical requirements to the class of languages  $\mathcal{V}$ , with no requirements left for the complexity class  $\text{Leaf}^P(\mathcal{V})$ .

We already mentioned that we could not find an example of a  $*$ -variety of languages  $\mathcal{V}$  such that the opposite direction of Corollary 1 does not hold via oracles, i.e. such that  $T \cdot \text{Leaf}^P(\mathcal{V})$  is a proper subset of  $\text{Leaf}^P(\text{UPol}(\mathcal{V}))$ . Could it be the case that  $\text{UPol}$  and  $T$  generally correspond on  $*$ -varieties of languages via leaf languages, without any conditions on the  $*$ -varieties of languages?

When does a complexity class  $\text{Leaf}^P(\mathcal{V})$  for a positive  $*$ -variety of languages  $\mathcal{V}$  contain a language  $L$  such that  $\text{Leaf}^P(\mathcal{V}) = \text{Leaf}^P(L)$ ? The classes  $\text{Leaf}^P(\Sigma_k^L)$ ,  $\text{Leaf}^P(\Pi_k^L)$ , and  $\text{Leaf}^P(\Delta_k^L)$  do have one, the classes  $\text{Leaf}^P(\mathcal{DD}_k)$  and  $\text{Leaf}^P(\mathcal{SF})$  seem not to. Is there, for example, an algebraic property which corresponds to this distinction?

Selivanov and Wagner [SW04] defined a notion of reducibility for languages in the dot-depth hierarchy and, in light of our results, one would expect that the  $D_k$  languages which we defined are complete for the  $\Delta_k^L$ -classes with respect to such reductions.

**Acknowledgments.** We thank Jean-Eric Pin for his help with Table 1 and Klaus Wagner for comments and for pointing out Travers’ work [Tr02].

## References

- [BGS75] T. P. BAKER, J. GILL, R. SOLOVAY *Relativizations of the  $P = ? NP$  Question*, SIAM Journal on Computing **4**, 1975, pp. 431-442
- [BKS99] B. BORCHERT, D. KUSKE, F. STEPHAN: *On existentially first-order definable languages and their relation to NP*, Theoretical Informatics and Applications **33**, 1999, pp. 259–269
- [BSS99] B. BORCHERT, H. SCHMITZ, F. STEPHAN:  $\text{Leaf}^P(\Delta_2^B) = \Delta_2^P$ , unpublished manuscript, 1999
- [BS97] B. BORCHERT, R. SILVESTRI: *A characterization of the leaf language classes*, Information Processing Letters **63**, 1997, pp. 153-158

- [BCS92] D. P. BOVET, P. CRESCENZI, R. SILVESTRI: *A uniform approach to define complexity classes*, Theoretical Computer Science **104**, 1992, pp. 263–283.
- [BT00] H. BUHRMAN, L. TORENVLIET, *Randomness is Hard*. SIAM J. Comput. 30(5): 1485-1501 (2000)
- [BV98] H.-J. BURTSCHICK, H. VOLLMER: *Lindström Quantifiers and Leaf Language Definability*, International J. Foundations of Computer Science **9**, 1998, pp. 277-294.
- [He84] H. HELLER: *Relativized Polynomial Hierarchies Extending Two Levels* Mathematical Systems Theory **17**, 1984, pp. 71-84.
- [HL\*93] U. HERTRAMPF, C. LAUTEMANN, T. SCHWENTICK, H. VOLLMER, K. WAGNER: *On the power of polynomial-time bit-computations*, Proc. 8th Structure in Complexity Theory Conference, 1993, pp. 200–207.
- [Ka88] J. KADIN: *The Polynomial Time Hierarchy collapses if the Boolean Hierarchy collapses*, SIAM Journal of Computing **17**, 1988, pp. 1263–1282.
- [MS72] A. R. MEYER, L. J. STOCKMEYER: *The equivalence problem for regular expressions with squaring requires exponential space*, Proceedings 13th Annual IEEE Symposium on Switching and Automata Theory, 1972, pp. 125–129.
- [Pa94] C. PAPADIMITRIOU: *Computational Complexity*, Addison Wesley, 1990.
- [Pi97] J.-E. PIN: *Syntactic semigroups*, Chap. 10, Handbook of Language Theory, vol. 1, Springer Verlag, 1997, 679–746.
- [Pi98] J.-E. PIN: *Bridges for Concatenation Hierarchies*, Proc. Int. Conf. on Automata, Languages and Programming, 1998, pp. 431–442.
- [PST88] J.-E. PIN, H. STRAUBING, D. THÉRIEN: *Locally trivial categories and unambiguous concatenation*, Journal of Pure and Applied Algebra **52**, 1988, pp. 297–311.
- [PW97] J.-E. PIN, P. WEIL: *Polynomial closure and unambiguous product*, Theory of Computing Systems **30**, 1997, pp. 383–422.
- [Sch76] M. P. SCHÜTZENBERGER: *Sur le produit de concatenation non ambigu*, Semigroup Forum **13**, 1976, pp. 47–75.
- [Sel01] V. L. SELIVANOV: *Relating Automata-Theoretic Hierarchies to Complexity-Theoretic Hierarchies*, Proc. FCT, 2001, pp. 323–334
- [SW98] H. SCHMITZ, K. W. WAGNER, *The Boolean hierarchy over level 1/2 of the Straubing-Thérien hierarchy*, Technical report 201, Inst. für Informatik, Uni. Würzburg, 1998.
- [SW04] V. L. SELIVANOV, K. W. WAGNER, *A Reducibility for the Dot-Depth Hierarchy*, Proc. Mathematical Foundations of Computer Science, 2004, pp. 783–793.
- [St94] H. STRAUBING: *Finite Automata, Formal Logic, and Circuit Complexity*, Birkhäuser, Boston, 1994.
- [ST03] H. STRAUBING, D. THÉRIEN, *Regular Languages Defined by Generalized First-Order Formulas with a Bounded Number of Bound Variables*, Theory Comput. Syst. **36(1)**, 2003, pp. 29-69.
- [TT02] P. TESSON, D. THÉRIEN: *Diamonds are forever: the Variety DA*, in Semigroups, Algorithms, Automata and Languages, WSP, 2002, 475–499.
- [Tr02] S. TRAVERS: *Blattsprachen-Komplexitätsklassen: Über Turing-Abschluss und Counting-Operatoren*, Studienarbeit, Universität Würzburg, 2002
- [Wa90] K. W. WAGNER: *Bounded Query Classes*, SIAM Journal on Computing **19**, 1990, pp. 833–846
- [Yao85] A. C.C. YAO: *Separating the Polynomial Hierarchy by oracles*, Proc. 26th IEEE Symp. on the Foundations of Computer Science, 1985, pp. 1–10