The Dot-Depth and the Polynomial Hierarchy Corresspond on the Delta Levels

Bernd Borchert
Klaus-Jörn Lange
Frank Stephan
Pascal Tesson
Denis Thérien

WSI-2004-3

Universität Tübingen
Wilhelm-Schickard-Institut für Informatik
Arbeitsbereich Theoretische Informatik/Formale Sprachen
Sand 13
D-72076 Tübingen

borchert@informatik.uni-tuebingen.de

© WSI 2004
ISSN 0946-3852

The Dot-Depth and the Polynomial Hierarchy Correspond on the Delta Levels

Bernd Borchert[1], Klaus-Jörn Lange[1], Frank Stephan[2], Pascal Tesson[1], and Denis Thérien[3]

[1] Universität Tübingen, Germany
email: {borchert,lange,tesson}@informatik.uni-tuebingen.de
[2] NICTA, Sydney, Australia
email: fstephan@cse.unsw.edu.au
[3] McGill University, Montréal, Canada
email: denis@cs.mcgill.ca

Abstract. It is well-known that the Σ_k- and Π_k-levels of the dot-depth hierarchy and the polynomial hierarchy correspond via leaf languages. In this paper this correspondence will be extended to the Δ_k-levels of these hierarchies: $\mathrm{Leaf}^{\mathrm{P}}(\Delta_k^L) = \Delta_k^p$.

1 Introduction

It is well-known that the Σ_k- and Π_k-levels of the dot-depth hierarchy and the polynomial hierarchy correspond via leaf languages, i.e. for all $k \geq 1$ it holds:

$$\mathrm{Leaf}^{\mathrm{P}}(\Sigma_k^L) = \Sigma_k^p,$$

$$\mathrm{Leaf}^{\mathrm{P}}(\Pi_k^L) = \Pi_k^p.$$

This was shown by Burtschick & Vollmer [BV98]. 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 an earlier paper by Hertrampf et al. [HL*93]):

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

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

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

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

The aim of this paper is to extend the correspondence of the dot-depth hierarchy and the polynomial hierarchy to the Δ_k-levels of the two hierarchies which, for

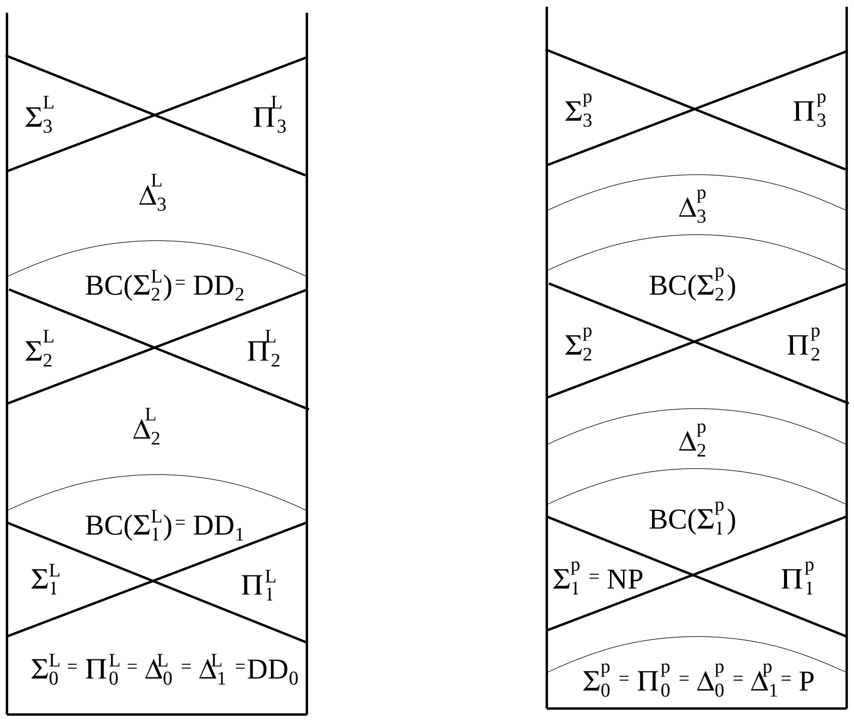

Fig. 1. The dot-depth and the polynomial hierarchy

the dot-depth hierarchy, are defined as the intersections of the corresponding Σ_k- and Π_k-levels and for the polynomial hierarchy are defined as the polynomial-time Turing reducibility closure of the Σ_{k-1}-class (see Figure 1). For level 2 this correspondence was already shown in the unpublished manuscript [BSS99]:

$$\mathrm{Leaf}^{\mathrm{P}}(\Delta_2^L) = \Delta_2^p.$$

The proof of this result used for the direction from left to right the Schützenberger characterization of Δ_L^2 as unambiguous products [Sch76], and for the direction from right to left a method from Wagner [Wa90] showing that the ODD MAX SAT problem is polynomial-time many-one complete for Δ_2^p. The main result of this paper is that for all $k \geq 2$:

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

This result will be a consequence of the following correspondence of the combined operator UPol $\circ$ BPol on certain $*$-varieties of regular languages $\mathcal{V}$, and the combination of polynomial-time Turing reducibility closure (denoted in dot operator notation $\mathrm{T}\,\cdot$ throughout this paper) applied to the projection operation $\exists\,\cdot$ on complexity classes:

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

Crucial ingredients of the proof of this operator correspondence are for the direction from left to right a characterization of the 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 Δ levels of the dot-depth hierarchy, with the notable exception of Δ_L^2: the survey [TT02] showed that this class is of computational relevance and has many nice properties and characterizations. On the polynomial hierarchy side it is fair to say that the class Δ_2^p is a "nobody" compared with the "celebrity" NP. But Δ_2^p should deserve more attention since problems in Δ_2^p can be considered as the problems of polynomial-time complexity less or equal to that of NP-complete problems because the polynomial-time Turing reducibility is a very natural way to compare polynomial-time complexities of computational problems ("given an algorithm for one problem for free, can you do the other problem in polynomial-time?").

The paper is organized as follows. In Section 2 we recall the definition of the dot-depth hierarchy, and of the related operators Pol and UPol. In Section 3 the concept of leaf languages and the polynomial hierarchy is presented. In Section 4 the correspondence of the Pol operator on the dot-depth hierarchy and the $\exists\cdot$ operator on the polynomial hierarchy is proven. In Section 5 basic properties of the Δ_k classes are presented, and the main result about the correspondence of the Δ_k levels is proven.

2 The Dot-Depth Hierachy

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

Let a class of languages $\mathcal{C}$ be given. The *polynomial closure* $\mathrm{Pol}(\mathcal{C})$ is the class of languages consisting, for every alphabet Σ, 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 Σ. 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 $\mathrm{UPol}(\mathcal{C})$ the class of finite disjoint unions of unambiguous marked products of languages of $\mathcal{C}$. The classes of languages Co-$\mathcal{C}$ and B$\mathcal{C}$ are defined, for every alphabet Σ, as the set of complements of languages from $\Sigma^*\mathcal{C}$ and as the Boolean closure of $\Sigma^*\mathcal{C}$, respectively. Co-Pol and BPol are defined as the combined operators Co- $\circ$ Pol and B $\circ$ Pol, respectively.

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

Definition 1 (dot-depth hierarchy).

$$\begin{array}{llll}
(a)\ \varSigma_0^L := \varPi_0^L := \varDelta_0^L := \mathcal{DD}_0 := \mathcal{I} & & (d)\ \mathcal{DD}_{k+1} := \mathrm{BPol}(\mathcal{DD}_k) \\
(b)\ \varSigma_{k+1}^L := \mathrm{Pol}(\mathcal{DD}_k) & & (e)\ \varDelta_{k+1}^L := \varSigma_{k+1}^L \cap \varPi_{k+1}^L \\
(c)\ \varPi_{k+1}^L := \text{Co-Pol}(\mathcal{DD}_k) & & (f)\ \mathcal{SF} := \bigcup_{i\geq 0} \varSigma_i^L
\end{array}$$

It should be noted that the dot-depth hierarchy presented here is sometimes known as the Straubing-Thérien hierarchy. Other papers use the term refering to the closely related Brzozowski hierarchy which is defined analogously: level 0 of the Brzozowski hierarchy further includes the so-called generalized definite languages. Although the hierarchies thus defined do not coincide, our results can also be obtained for the levels of the Brzozowski hierarchy as we will note in our conclusion.

Results of Pin and Weil [PW97] show that $\varSigma_{k+1}^L \cap \varPi_{k+1}^L = \mathrm{UPol}(\mathcal{DD}_k)$ 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 $\varSigma_k^L, \varPi_k^L, \mathcal{DD}_k, \varDelta_k^L$ 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
$\varSigma_k^L$	$\varSigma_k^L$	$\varPi_k^L$	$\mathcal{DD}_k$	$\varSigma_k^L$
$\varPi_k^L$	$\varSigma_{k+1}^L$	$\varSigma_k^L$	$\mathcal{DD}_k$	$\varDelta_{k+1}^L$
$\mathcal{DD}_k$	$\varSigma_{k+1}^L$	$\mathcal{DD}_k$	$\mathcal{DD}_k$	$\varDelta_{k+1}^L$
$\varDelta_{k+1}^L$	$\varSigma_{k+1}^L$	$\varDelta_{k+1}^L$	$\varDelta_{k+1}^L$	$\varDelta_{k+1}^L$

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

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

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

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

inverse homomorphic images: If $\varGamma, \varSigma$ are alphabets and h is a homomorphism from $\varGamma^*$ to $\varSigma^*$ and L is a language from $\varSigma^*\mathcal{V}$ then $h^{-1}(L) = \{x \in \varGamma^* \mid h(x) \in L\}$ is in $\varSigma_0^*\mathcal{V}$.

A *positive $*$-variety of languages* is defined the same way but the closure under complementation is not necessary.

Examples of $*$-varieties of languages include $\mathcal{DD}_k$ and Δ_k^L for $k \geq 0$, and $\mathcal{SF}$ while Σ_k^L and Π_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 Σ and each $\Sigma_0 \subseteq \Sigma$, we have $\Sigma_0^* \in \Sigma^*\mathcal{V}$. This is equivalent to having a non-trivial intersection with Π_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 contain 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 Hierachy

Let some language L over some alphabet Σ be given. The *leaf language* approach [BCS92,HL*93,BKS99] assigns to the language L a class $\mathrm{Leaf}^{\mathrm{P}}(L)$ of languages on the alphabet $\{0,1\}$ the following way. Let some nondeterministic polynomial-time Turing machine N be given. Assume that it not only accepts or rejects on every computation path (that would correspond to a 2-letter alphabet like $\{0,1\}$) but outputs a letter from the alphabet Σ on each computation path when it terminates. N produces for every input $x \in \{0,1\}^*$ a computation tree (not necessary balanced) whose paths are ordered in the natural way and whose leaves are labeled by letters from Σ. Therefore the letters on the leaves form a word over the alphabet Σ which we call the leafstring(N, x) or the yield of the computation tree. Let for each N the language $\mathrm{Leaf}^N(L) \in \{0,1\}^*$ be the set of inputs x such that leafstring(N, x) is in L, and let $\mathrm{Leaf}^{\mathrm{P}}(L)$ be the set of languages $\mathrm{Leaf}^N(L)$ for some N. As an example note that for the language $S_1 := \{0,1\}^*1\{0,1\}^*$ over alphabet $\{0,1\}$ it holds $\mathrm{Leaf}^{\mathrm{P}}(S_1) = \mathrm{NP}$. For a class $\mathcal{C}$ of languages let $\mathrm{Leaf}^{\mathrm{P}}(\mathcal{C})$ be the union of the classes $\mathrm{Leaf}^{\mathrm{P}}(L)$ for $L \in \mathcal{C}$. Note that all classes $\mathrm{Leaf}^{\mathrm{P}}(L)$ and $\mathrm{Leaf}^{\mathrm{P}}(\mathcal{C})$ are by definition subsets of the set of languages over the alphabet $\{0,1\}$. We will call such classes *complexity classes*. We 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 languages.

Let P (NP) be the set of languages computable by a Turing machine in deterministic (nondeterministic) polynomial time. For a complexity class $\mathcal{C}$ let $\mathrm{T}\cdot\mathcal{C}$ be the set of languages computable in deterministic polynomial time via an oracle Turing machine which uses a language from $\mathcal{C}$ as an oracle[1]. Let $\exists \cdot \mathcal{C}$ be the set of all languages L such that

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

[1] This class is often denoted $\mathrm{P}(\mathcal{C})$, $\mathrm{P}^{\mathcal{C}}$ or $\leq_T^p(\mathcal{C})$.

for some language $A \in \mathcal{C}$ and some polynomial q. Let $\mathrm{co} \cdot \mathcal{C}$ be the set of complements of $\mathcal{C}$ and $\mathrm{BC} \cdot \mathcal{C}$ 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 $\mathrm{P} = \mathrm{T} \cdot \emptyset$, $\mathrm{NP} = \exists \cdot \mathrm{P}$, $\text{co-NP} = \mathrm{co} \cdot \mathrm{NP}$, $\mathrm{BC(NP)} = \mathrm{BC} \cdot \mathrm{NP}$, and $\Delta_2^p = \mathrm{T} \cdot \mathrm{NP}$. The classes of the polynomial hierarchy are defined as follows [MS72,Pa94], (see Figure 1):

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

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

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

class/operator	$\exists \cdot$	$\mathrm{co} \cdot$	$\mathrm{BC} \cdot$	$\mathrm{T} \cdot$
Σ_k^p	Σ_k^p	Π_k^p	$\mathrm{BC}(\Sigma_k^p)$	Δ_{k+1}^p
Π_k^p	Σ_{k+1}^p	Σ_k^p	$\mathrm{BC}(\Sigma_k^p)$	Δ_{k+1}^p
$\mathrm{BC}(\Sigma_k^p)$	Σ_{k+1}^p	$\mathrm{BC}(\Sigma_k^p)$	$\mathrm{BC}(\Sigma_k^p)$	Δ_{k+1}^p
Δ_{k+1}^p	Σ_{k+1}^p	Δ_{k+1}^p	Δ_{k+1}^p	Δ_{k+1}^p

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

It is not known whether the polynomial hierarchy is in fact infinite. Nevertheless, there exists a relativized world in which all classes Σ_k^{p}, Π_k^{p} and Δ_k for $k \geq 1$ are different from one another [Yao85]. Moreover, in this world, Δ_k^p is different from $\mathrm{BC}(\Sigma_{k-1}^p)$ for any $k \geq 2$ because Δ_k^p still has, for every relativization, a $\leq_m^p$-complete language while by the (relativizable) results of Kadin [Ka88] the class $\mathrm{BC}(\Sigma_{k-1}^p)$ does not. For $k = 1$ and $k = 2$ oracles separating Δ_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 litterature. 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.

4 The Sigma and Pi Levels of the Hierarchies

We will show that the Pol operator on positive varieties of regular languages corresponds, under certain conditions, via leaf languages to the dot operator $\exists \cdot$ on the polynomial-time complexity classes. This was already mentioned but not proven explicitly in the paper of Hertrampf et al. [HL*93]. Let some language L over an alphabet Σ_0 be given. Let E_L be the language $\Sigma^* bLb\Sigma^*$ over the alphabet $\Sigma = \Sigma_0 \cup \{b\}$ where b is some letter (called *marker*) not in Σ_0. Note that if L is in a positive variety $\mathcal{V}$ which contains the sub-alphabets then E_L is in $\mathrm{Pol}(\mathcal{V})$.

Lemma 3 ([HL*93]). *Let $\mathcal{V}$ be a positive $*$-variety of languages which contains the sub-alphabets.*
(a) $\mathrm{Leaf}^{\mathrm{P}}(\mathrm{Pol}(\mathcal{V})) = \exists \cdot \mathrm{Leaf}^{\mathrm{P}}(\mathcal{V})$.
(b) If $\mathrm{Leaf}(L) = \mathrm{Leaf}(\mathcal{V})$ *for a single language* $L \in \mathcal{V}$ *then* $\mathrm{Leaf}(E_L)$ *equals the two classes from (a).*

Proof. Direction (a) $\subseteq$: Let L be a leaf language $L = L_0 a_1 L_1 \cdots a_n L_n$ from $\Sigma^*\mathrm{Pol}(\mathcal{V})$, i.e. each L_i is a language from $\Sigma^*\mathcal{V}$ and all markers a_i are from Σ, too. 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 $L_i \sqcup\!\sqcup\, 0^*$ (the language L_i shuffled with 0's) is also in $\mathcal{V}$. We define the language J over the $(n+1)$-fold product $\Sigma_0 \times \ldots \times \Sigma_0$

$$J = \left\{ \begin{pmatrix} a_{10} \\ a_{11} \\ \vdots \\ a_{1n} \end{pmatrix} \begin{pmatrix} a_{20} \\ a_{21} \\ \vdots \\ a_{2n} \end{pmatrix} \cdots \begin{pmatrix} a_{s0} \\ a_{s1} \\ \vdots \\ a_{sn} \end{pmatrix} \middle| \text{ for each } i,\ a_{1i} \ldots a_{si} \in L_i \sqcup\!\sqcup\, 0^* \right\}.$$

By further using closure under finite intersections, we get that J also belongs to $\mathcal{V}$. Let some nondeterministic polynomial-time Turing machine M working with leaf language L be given. An NDTM working with the leaf language J can be seen as writing its leafstring on $n+1$ different, independent tracks corresponding to the $n+1$ copies of the alphabet Σ_0. Let M' be such an NDTM which, on input $\langle x, y\rangle$ checks whether y encodes n paths $p_1 < \ldots < p_n$ of the computation produced by M on input x and first checks if the letters written at these positions by M are $a_1, \ldots, a_n$. If this is not the case M' rejects, i.e. writes a computation tree whose leafstring is not in J (this word exists because $\mathcal{V}$ is not trivial since it contains the sub-alphabets). Otherwise, let $p_0 = 0$ and $p_{n+1} = \max + 1$ where 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 $(0, \ldots 0, b, 0 \ldots 0)^T$ where b is in the i-th track of the tuple and is the letter which was written by M on path p. On the guessed paths $p_1 < \ldots < p_n$ it produces an all-0 tuple. The i-th track of the resulting leafstring is then a word of the form $0^{n_i} w_i 0^{m_i}$ (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 $0^{n_i} w_i 0^{m_i}$ belong to $L_i \sqcup\!\sqcup 0^*$ if and only if the whole leafstring belongs to J if and only if M' accepts on input $\langle x, y\rangle$. Thus, M accepts x if and only if there exists y for which M' accepts on input $\langle x, y\rangle$. Note that in general L could be a finite union of such marked products in which case M' would further need to guess which marked product to work with.

Direction (a) $\supseteq$: Let L from $\Sigma^*\mathcal{V}$, q a polynomial and K a language of $\exists \cdot \mathrm{Leaf}^{\mathrm{P}}(\mathcal{V})$:

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

We will construct a NDTM M' using E_L as a leaf language to accept the language K. 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 via a simulation of 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). *q.e.d.*

We will later need the above lemma to prove Lemma 10. As a byproduct, we can also reprove a theorem due partly to Burtschick and Vollmer [BV98] and partly to Hetrampf et al. [HL*93]. We define for $k \geq 1$ *canonical languages* S_k and P_k for the classes Σ_k^L and Π_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. The canonical Π_2^L language P_2 is for example 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 Σ_k^L and P_k is an element of Π_k^L.

Theorem 4 ([BV98,HL*93]). *Let* $k \geq 1$.
(a) $\mathrm{Leaf}^{\mathrm{P}}(S_k) = \mathrm{Leaf}^{\mathrm{P}}(\Sigma_k^L) = \Sigma_k^p$,
(b) $\mathrm{Leaf}^{\mathrm{P}}(P_k) = \mathrm{Leaf}^{\mathrm{P}}(\Pi_k^L) = \Pi_k^p$.
(c) $\mathrm{Leaf}^{\mathrm{P}}(\mathcal{DD}_k) = \mathrm{BC}(\Sigma_k^p)$.
(d) $\mathrm{Leaf}^{\mathrm{P}}(\mathcal{SF}) = \mathrm{PH}$

Proof. We use a simultaneous induction to prove parts (a), (b) and (c). The induction base $k = 1$ was proven in [BKS99]. For $k \geq 2$, we have

$$\begin{aligned}
\mathrm{Leaf}^{\mathrm{P}}(\Sigma_k^L) &= \mathrm{Leaf}^{\mathrm{P}}(\mathrm{Pol}(\mathcal{DD}_{k-1})) \text{ (Definition 1(b))}\\
&= \mathrm{Leaf}^{\mathrm{P}}(\mathrm{Pol}(\Pi_{k-1}^L)) \text{ (see Table 1)}\\
&= \exists \cdot \mathrm{Leaf}^{\mathrm{P}}(\Pi_{k-1}^L) \text{ (by Lemma 3)}\\
&= \exists \cdot (\Pi_{k-1}^p) \text{ (by induction hypothesis)}\\
&= \Sigma_k^{\mathrm{P}}
\end{aligned}$$

The equalities $\mathrm{Leaf}^{\mathrm{P}}(\Pi_k^L) = \Pi_k^{\mathrm{p}}$ and $\mathrm{Leaf}^{\mathrm{P}}(\mathcal{DD}_k) = \mathrm{BC}(\Sigma_k^p)$ readily follow. The existence of single leaf languages S_k, P_k characterizing the Σ_k^p and Π_k^p levels, respectively, follows by induction via part (b) of Lemma 3.

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

5 The Delta Levels of the Hierarchies

In this section we are going to prove the correspondence of the Delta levels of the dot-depth and the polynomial hierarchy. This will be a corollary of Lemma 10 which states that, under certain technical conditions, the unambiguous polynomial closure operator UPol on $*$-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 $\mathrm{T}\cdot$ and $\mathrm{BC}\cdot$:

Proposition 5. *Let $\mathcal{C}$ be a complexity class and $\mathcal{V}$ be a (positive) variety of languages.*
(a) $\mathrm{T} \cdot \mathrm{T} \cdot \mathcal{C} = \mathrm{T} \cdot \mathcal{C}$.
(b) *If* $\mathcal{C} \subseteq \mathrm{T} \cdot \mathrm{Leaf}^{\mathrm{P}}(\mathcal{V})$ *then* $\mathrm{BC} \cdot \mathcal{C} \subseteq \mathrm{T} \cdot \mathrm{Leaf}^{\mathrm{P}}(\mathcal{V})$,
(c) $\mathrm{BC} \cdot \mathrm{Leaf}^{\mathrm{P}}(\mathcal{V}) \subseteq \mathrm{T} \cdot \mathrm{Leaf}^{\mathrm{P}}(\mathcal{V})$,
(d) $\mathrm{T} \cdot \mathrm{BC} \cdot \mathrm{Leaf}^{\mathrm{P}}(\mathcal{V}) = \mathrm{T} \cdot \mathrm{Leaf}^{\mathrm{P}}(\mathcal{V})$.

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 $L_1 \in T \cdot \mathrm{Leaf}^{\mathrm{P}}(K_1)$ and $L_2 \in T \cdot \mathrm{Leaf}^{\mathrm{P}}(K_2)$ with $K_1, K_2 \in \mathcal{V}$ then we can find some $K \in \mathcal{V}$ such that $L_1 \cap L_2 \in T \cdot \mathrm{Leaf}^{\mathrm{P}}(K)$. For instance, we can take $K = K_1 \sqcup\!\sqcup K_2$ where K_1 and K_2 are assumed (w.l.o.g.) to be over disjoint alphabets. It can then be shown easily that $K \in \mathcal{V}$ and that $\mathrm{Leaf}^{\mathrm{P}}(K_i) \subseteq \mathrm{Leaf}^{\mathrm{P}}(K)$. *q.e.d.*

We will use the following characterization of the UPol operator. Let a $*$-variety of languages $\mathcal{V}$ be given: $\ell_1 * \mathcal{V}$ ($\mathcal{V} *_r \ell_1$) is the Boolean closure of languages[2] of *unambiguous* products $La\Sigma^*$ ($\Sigma^* aL$) where $L \in \Sigma^*\mathcal{V}$ and $a \in \Sigma$. By [PST88], $\ell_1 * \mathcal{V}$ and $\mathcal{V} *_r \ell_1$ are themselves $*$-varieties of languages. Moreover, $\mathrm{UPol}(\mathcal{V})$ can be characterized by these two operators:

[2] The symbols $*$, $*_r$ and ℓ_1 actually have a meaning as wreath product, reversed wreath product and the set of locally trivial categories, respectively, see for example [PST88].

Theorem 6 ([PST88]). *Let $\mathcal{V}$ be a $*$-variety of languages.* $\mathrm{UPol}(\mathcal{V})$ *equals the class of languages obtainable from $\mathcal{V}$ by finitely many applications of the operators $\ell_1 *$ and $*_r \ell_1$.*

We first show that one application of these two operators can basically be simulated by a polynomial-time Turing reduction.

Lemma 7. *Let $\mathcal{V}$ be a $*$-variety of languages.*
(a) $\mathrm{Leaf}^{\mathrm{P}}(\ell_1 * \mathcal{V}) \subseteq \mathrm{T} \cdot \mathrm{Leaf}^{\mathrm{P}}(\mathcal{V})$,
(b) $\mathrm{Leaf}^{\mathrm{P}}(\mathcal{V} *_r \ell_1) \subseteq \mathrm{T} \cdot \mathrm{Leaf}^{\mathrm{P}}(\mathcal{V})$.

Proof. By symmetry, we need only prove (a). Assume we have an unambiguous concatenation of the form $L_0 = La\Sigma^*$ with $L \in \mathcal{V}$. We have to show that $\mathrm{Leaf}^{\mathrm{P}}(L_0) \subseteq \mathrm{T} \cdot \mathrm{Leaf}^{\mathrm{P}}(\mathcal{V})$. For this we use the following property of L_0. Let w be a word of Σ^* and x be the longest prefix of w which is also prefix of some word in L. If $x \neq w$, i.e. $w = xby$ with a letter $b \in \Sigma$ and word $y \in \Sigma^*$ then:

$$(x \in L \text{ and } b = a) \Leftrightarrow w \in L_0.$$

The direction $\Rightarrow$ of course holds because in that case $w = xby$ matches the pattern $L_0 = La\Sigma^*$. For the other direction, assume that $x \notin L$ or $b \neq a$, i.e. the word w does not match the pattern $La\Sigma^*$ via an a at the position right of x. In order to let $w = xby$ match the pattern $La\Sigma^*$ the position of the a in this pattern has to be either within x or within y. It cannot be within y since this would mean that xb is the prefix of a word in L, contradicting the maximality assumption for x. But it can not be within x either for if $x = vaz$ with $v \in L$, then let e be such that $xe \in L$ (this e exists since x is a prefix of a word in L): the word $xea = vazea$ has two factorizations $v\underline{a}zea$ and $vaze\underline{a}$ showing membership in the marked product $L_0 = La\Sigma^*$, contradicting the unambiguity of L_0. For the case $x = w$ the same argument shows that w cannot be from L_0.

The following algorithm will use the above property: First it searches for the longest prefix x in the leafstring of the computation tree which is also a prefix of a word in L and then it checks whether (i) this prefix is in L and (ii) the path right next to it exists and has letter a. More precisely, let L_0 be the unabmiguous concatenation $La\Sigma^*$. Because $\mathcal{V}$ is a $*$-variety of languages, the set L_p of prefixes of words in L is in $\mathcal{V}$. Indeed, L_p is by definition the infinite union of the languages Lw^{-1} over all $w \in \Sigma^*$. But if $v \equiv_L w$ for the congruence relation $\equiv_L$ on Σ^* (see for example [St94]) it holds $Lv^{-1} = Lw^{-1}$. Because L is regular there exist only a finite number of equivalence classes w.r.t. the $\equiv_L$ congruence relation. Therefore L_p is the union of the finitely many right quotients $Lu_1^{-1}, \ldots Lu_j^{-1}$ where $u_1, \ldots u_j$ are representatives of the equivalence classes of the $\equiv_L$ congruence relation on Σ^*. By the closure under right quotients and under finite unions, L_p is in $\mathcal{V}$.

Our algorithm will have to ask questions to oracle $\mathrm{Leaf}^{\mathrm{P}}(L)$ and to oracle $\mathrm{Leaf}^{\mathrm{P}}(L_p)$, therefore we have to combine them into one oracle. This is possible since $\mathcal{V}$ is a variety: let Σ_0 be an alphabet with the same cardinality as Σ

but disjoint from Σ, and let π be a bijection from Σ_0 to Σ. Let J be the shuffle language $L \sqcup\!\sqcup \pi(L_p)$ on alphabet $\Sigma \cup \Sigma_0$. By the closure under inverse homomorphic images J belongs to $\mathcal{V}$. Let U be the the "double universal" NTDM for $\mathrm{Leaf}^{\mathrm{P}}(J)$, i.e. let U be the NDTM which on input $0\langle x, M, 0^k\rangle$ simulates M on input x for k steps printing on a leaf the letter a in case M prints letter a, and which on input $1\langle x, M, 0^k\rangle$ simulates M on input x for k steps printing on a leaf the letter $\pi(a)$ in case M prints letter a. Given a NDTM M working with leaf language L_0 a polynomial-time oracle Turing machine D and a NDTM U working with leaf language J will be constructed such that $\mathrm{Leaf}^M(L_0)$ equals the language accepted by D with oracle $\mathrm{Leaf}^U(J)$. The oracle DTM D will work the following way: On an input x, D will find the longest prefix of the leafstring of the computation of M on input x belonging to L. This will be done via binary search and the help of the oracle $\mathrm{Leaf}^U(J)$ (actually only using the L_p part of J): First D will ask the oracle whether the leftmost part of the leafstring is in L_p, i.e. it will ask the query $1\langle x, M_1, 0^{q(|x|)}\rangle$ where q is the run time bound of M and M_1 is the machine which works like M but just evaluates the leftmost subtree of the computation tree. If the answer is 'yes' then the leafstring of the leftmost subtree of the computation tree of M on input x is a prefix of L, and D will continue the binary search in the rightmost subtree. If the answer is 'no' then D continue the binary search for the longest prefix in the leftmost subtree. Thus D will find after a polynomially many questions the path i such that the prefix $w_0 \dots w_i$ of the leafstring of the computation tree of M working on input x is the longest prefix of the leafstring which belongs to L_p. Now D will ask the oracle (only using the L part of J) whether $w_0 \dots w_i$ is in L by giving it $0\langle x, M', 0^{q(|x|)}\rangle$ where M' is the NDTM which works like M but only evaluates the paths up to path i. If the answer is no, D rejects. If it is 'yes' then it checks in deterministic polynomial-time whether the path $i+1$ exist in the computation tree (i may be its rightmost path) and has letter a written on the leaf of path $i+1$. If this is the case D accepts, otherwise it rejects. This behaviour is correct by our first argument, i.e. D accepts an input iff the leafstring produced by M is in $La\Sigma^*$.

We have shown $\mathrm{Leaf}^{\mathrm{P}}(\mathcal{U}) \subseteq \mathrm{T} \cdot \mathrm{Leaf}^{\mathrm{P}}(\mathcal{V})$ for the set $\mathcal{U}$ of unambiguous concatenations of the form $L_0 = La\Sigma^*$ with $L \in \mathcal{V}$. Therefore:

$$\mathrm{Leaf}^{\mathrm{P}}(\ell_1 * \mathcal{V}) = \mathrm{Leaf}^{\mathrm{P}}(\mathrm{B}\mathcal{U}) \subseteq \mathrm{BC} \cdot \mathrm{Leaf}^{\mathrm{P}}(\mathcal{U}) \subseteq \mathrm{T} \cdot \mathrm{Leaf}^{\mathrm{P}}(\mathcal{V})$$

by Proposition 5 (b). *q.e.d.*

Corollary 8. *Let $\mathcal{V}$ be a $*$-variety of languages.*
$\mathrm{Leaf}^{\mathrm{P}}(\mathrm{UPol}(\mathcal{V})) \subseteq \mathrm{T} \cdot \mathrm{Leaf}^{\mathrm{P}}(\mathcal{V})$.

Proof. According to Theorem 6 $\mathrm{UPol}(\mathcal{V})$ is generated by finitely many applications of the two operators $\ell_1 *$ and $*_r \ell_1$, 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 two operators the statement holds by Lemma 7 together with the idempotency of the Turing closure operator $\mathrm{T}\cdot$ stated in Proposition 5(a). *q.e.d.*

The authors do not know whether the reverse inclusion in the above Corollary 8 holds in general, i.e. whether the UPol operator on varieties corresponds to the Turing reducibility closure on the polynomial-time degrees. Nevertheless, Lemma 10 below will show that, under certain conditions, a correspondence holds at least for the combined operator UPol $\circ$ BPol on language classes and the combined operator $\mathrm{T} \cdot \exists \cdot$ on complexity classes.

For a language L over alphabet Σ_0 the language, 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 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 Σ_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 9. *Let L be a language from a $*$-variety of languages $\mathcal{V}$ which contains the sub-alphabets. Then W_L is a language from* UPol(BPol($\mathcal{V}$)).

Proof. Let $L \in \mathcal{V}$ over alphabet Σ_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

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

It is in Pol($\mathcal{V}$), therefore its complement $\overline{K}$ is from BPol($\mathcal{V}$). Now we can write

$$W_L = \overline{K}\{a, b\}L_0 a\Sigma^* \;\cup\; L_0 a\Sigma^*$$

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 in the language $\overline{K}\{a, b\}$ a subword from $\{a, b\}L_0 a$ is forbidden, and in the second expression the position of the a is, as the first marker occurring, uniquely determined. This concatenation representation of W_L shows $W_L \in$ UPol(BPol($\mathcal{V}$)). *q.e.d.*

We say that a language A *polynomial-time conjunctively reducible* to a language B if there is a polynomial-time oracle Turing machine for A which on input x asks membership questions to 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. Call a class $\mathcal{C}$ of languages *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 Δ_k^p classes have this property. Other classes have this property and in particular Σ_k^p and Π_k^p (via two different techniques).

Lemma 10. *Let $\mathcal{V}$ be a $*$-variety of languages which contains the sub-alphabets, and let* $\mathrm{Leaf}^{\mathrm{P}}(\mathcal{V})$ *be closed under polynomial-time conjunctive reductions.*
(a) $\mathrm{Leaf}^{\mathrm{P}}(\mathrm{UPol}(\mathrm{BPol}(\mathcal{V}))) = \mathrm{T} \cdot \exists \cdot \mathrm{Leaf}^{\mathrm{P}}(\mathcal{V})$*,*
(b) If $L \in \mathcal{V}$ is a single language such that $\mathrm{Leaf}^{\mathrm{P}}(L) = \mathrm{Leaf}^{\mathrm{P}}(\mathcal{V})$ *then* $\mathrm{Leaf}^{\mathrm{P}}(W_L)$ *equals the classes from (a).*

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

$$\begin{aligned}\mathrm{Leaf}^{\mathrm{P}}(\mathrm{UPol}(\mathrm{BPol}(\mathcal{V}))) &\subseteq \mathrm{T} \cdot \mathrm{Leaf}^{\mathrm{P}}(\mathrm{BPol}(\mathcal{V})) \text{ (Lemma 8)}\\ &\subseteq \mathrm{T} \cdot \mathrm{BC} \cdot \mathrm{Leaf}^{\mathrm{P}}(\mathrm{Pol}(\mathcal{V}))\\ &= \mathrm{T} \cdot \mathrm{BC} \cdot \exists \cdot \mathrm{Leaf}^{\mathrm{P}}(\mathcal{V}) \text{ (Lemma 3)}\\ &= \mathrm{T} \cdot \exists \cdot \mathrm{Leaf}^{\mathrm{P}}(\mathcal{V}) \text{ (Proposition 5(d))}.\end{aligned}$$

The proof of the other direction generalizes an idea of Wagner [Wa90] whose result can be interpreted as showing that $\Delta_2^p \in \mathrm{Leaf}^{\mathrm{P}}(0^*a\{0,a,b\}^*)$. Since this fact will later be needed in the proof of Theorem 11, 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 there exists a $\mathrm{Leaf}^{\mathrm{P}}(0^*a\{0,a,b\}^*)$ N that can simulate any deterministic polynomial time machine D querying, say, a SAT oracle. 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, if 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, $\mathrm{leafstring}(N,x) \in 0^*a\{0,a,b\}^*$ iff x is accepted by D. This shows $\Delta_2^p = \mathrm{P}^{\mathrm{SAT}} \subseteq \mathrm{Leaf}^{\mathrm{P}}(0^*a\{0,a,b\}^*)$.

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

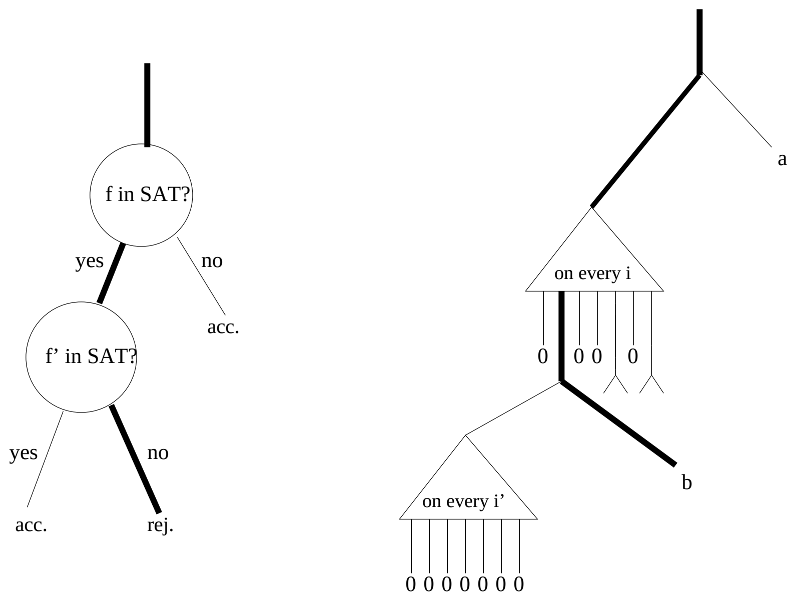

Fig. 2. Building a computation tree for the leaf language $0^*a\{0, a, b\}^*$

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 $\mathrm{T} \cdot \exists \cdot \mathrm{Leaf}^{\mathrm{P}}(L_0) \subseteq \mathrm{Leaf}^{\mathrm{P}}(W_L)$ 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:

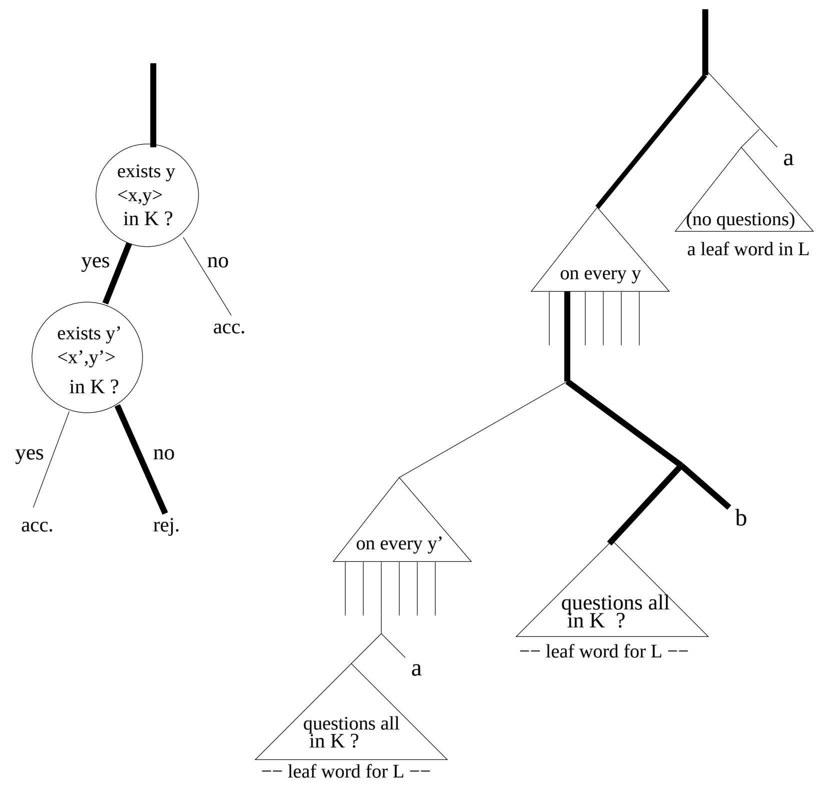

Fig. 3. Building a computation tree for the leaf language W_L

- 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 $\text{Leaf}^{\text{P}}(\mathcal{V})$.
- 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 $\text{T} \cdot \exists \cdot \text{Leaf}^{\text{P}}(L_0) \subseteq \text{Leaf}^{\text{P}}(W_L)$ and so, by Lemma 9 we conlude $\text{T} \cdot \exists \cdot \text{Leaf}^{\text{P}}(L_0) \subseteq \text{Leaf}^{\text{P}}(\text{UPol}(\text{BPol}(\mathcal{V})))$.

(b) follows from the construction in part $\supseteq$ above. *q.e.d.*

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 Δ_2^L and so we have $D_k \in \Delta_k$ using Lemma 9.

Theorem 11 (Main). *For every $k \geq 2$:*
$\mathrm{Leaf}^{\mathrm{P}}(D_k) = \mathrm{Leaf}^{\mathrm{P}}(\Delta_k^L) = \Delta_k^p$.

Proof. We establish $\mathrm{Leaf}^{\mathrm{P}}(D_k) = \mathrm{Leaf}^{\mathrm{P}}(\Delta_k^L) = \Delta_k^p$ by induction on k. For $k = 2$, we have already shown in the previous proof that $\Delta_2^p \subseteq \mathrm{Leaf}^{\mathrm{P}}(D_2) \subseteq \mathrm{Leaf}(\Delta_2^L)$. Furthermore, by Corollary 8 we get

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

For the induction step, we get:

$$\begin{aligned}\mathrm{Leaf}^{\mathrm{P}}(\Delta_{k+1}^L) &= \mathrm{Leaf}^{\mathrm{P}}(\mathrm{Upol}(\mathrm{BPol}(\Delta_k^L))) \text{ (See Figure 1)}\\ &= T \cdot \exists \cdot \mathrm{Leaf}^{\mathrm{P}}(\Delta_k^L) \text{ (Lemma 10)}\\ &= T \cdot \exists \cdot \Delta_k^p = \Delta_{k+1}^p \text{ (from the induction hypothesis)}\end{aligned}$$

Note that we can indeed apply Lemma 10: by induction we have $\mathrm{Leaf}^{\mathrm{P}}(\Delta_k^L) = \Delta_k^p$ which is closed under conjunctive reductions. We also get from Lemma 10 that $\mathrm{Leaf}^{\mathrm{P}}(\Delta_{k+1}^L) = \mathrm{Leaf}^{\mathrm{P}}(W_{D_k}) = \mathrm{Leaf}^{\mathrm{P}}(D_{k+1})$. *q.e.d.*

We should note that Meyer and Stockmeyer made the somewhat uncanonical choice of defining Δ_{k+1}^p as $\mathrm{T} \cdot \Sigma_k^p$ rather than $\Pi_{k+1}^p \cap \Sigma_{k+1}^p$, probably because this makes Δ_{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 4.

In Lemma 10, the technical condition that $\mathcal{V}$ contains the sub-alphabets is in fact superfluous: if $\mathcal{V}$ is a variety of languages whose syntactic monoids are groups and such that $\mathrm{Leaf}^{\mathrm{P}}(\mathcal{V})$ is closed under conjunctive reductions then we indeed have $\mathrm{Leaf}^{\mathrm{P}}(\mathrm{UPol}(\mathrm{BPol}(\mathcal{V})))) = T \cdot \exists \cdot \mathrm{Leaf}^{\mathrm{P}}(\mathcal{V})$. Handling of that case requires only a slightly different construction both for W_L and for the simulation of the $T \cdot \exists \cdot \mathrm{Leaf}^{\mathrm{P}}(L)$-machine as well as extra algebraic considerations. In [ST03], it was shown that the class of regular languages which can be defined by two-variable sentences using ordinary and modular quantifiers is exactly the class $\mathcal{DA} * \mathcal{G}_{sol} = \mathrm{UPol}(\mathrm{BPol}(\mathcal{G}_{\mathrm{sol}}))$ where $\mathcal{G}_{sol}$ denotes the class of languages whose syntactic monoid is a solvable group. Our results combined with [HL*93] show

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

where $\mathrm{MOD}^*\mathrm{P}$ denotes the closure of P under the Mod_q operators. Similarly, for any prime q, let $\mathcal{G}_q$ be the class of languages whose syntactic monoids are q-groups: we can show

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

All these results relativize and this allows us to shed new light on the problem of finding a leaf-language upper bound for BPP: the classical result of Lautemann and Sipser (see [Pa94]) shows that BPP is contained in $\Sigma_2^p \cap \Pi_2^p$ and it is natural to ask whether there is a language L in $\Sigma_2^L \cap \Pi_2^L$ with BPP $\subseteq$ $\mathrm{Leaf}^{\mathrm{P}}(L)$. This we now know cannot be the case with respect to all oracles since we will have $\mathrm{Leaf}^{\mathrm{P}}(L) \subseteq \Delta_2^p$ whereas relativized worlds exist in which Δ_2^p is strictly contained in BPP (see e.g. [BT00]). Straubing and Thérien have conjectured that BPP sits in $\mathrm{Leaf}^{\mathrm{P}}(\mathrm{UPol}(\mathrm{BPol}(\mathcal{G}_{\mathrm{sol}})))$ but there exist relativized worlds in which $T \cdot \exists \cdot \oplus \mathrm{P}$ does not contain BPP [BT00] and it is perhaps possible to further show that in this world even $T \cdot \exists \cdot \mathrm{MOD}^*\mathrm{P}$ does not contain BPP. Such a result would rule out any relativizable proof of the aforementioned conjecture.

6 Conclusion

We have shown that the Delta classes of the dot-depth hierarchy and of the polynomial hierarchy correspond via leaf-languages. This result also holds if we consider the Brzozowski definition of the dot-depth hierarchy. This extension of our result can be obtained using the "bridging" method outlined e.g. in [Pi98] which relates the two hierarchies in a straightforward way.

We know by Theorem 11 that there are languages in Δ_2^L, for example D_2, that capture the class Δ_2^{p}. Using algebraic methods, one can prove that if L is in Δ_2^L then either $\mathrm{Leaf}^{\mathrm{P}}(L) \subseteq \mathrm{BC(NP)}$ or $\mathrm{Leaf}^{\mathrm{P}}(L) = \Delta_2^{\mathrm{p}}$ but we do not know if a similar phenomenon occurs for $k \geq 3$. A related question is whether the D_k languages which we defined are complete for the Δ_k^L-classes with respect to the reductions defined in [SW03].

For all varieties considered in this paper, we do have that $\mathrm{Leaf}^{\mathrm{P}}(\mathcal{V})$ 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 10 "cleaner" because it would shift the technical requirements to the class of languages $\mathcal{V}$, with no requirements left for the complexity class $\mathrm{Leaf}^{\mathrm{P}}(\mathcal{V})$.

We already mentioned that we could not find an example of a $*$-variety of languages $\mathcal{V}$ such that the opposite direction of Corollary 8 does not hold via oracles, i.e. such that $\mathrm{T} \cdot \mathrm{Leaf}^{\mathrm{P}}(\mathcal{V})$ is a proper subset of $\mathrm{Leaf}^{\mathrm{P}}(\mathrm{UPol}(\mathcal{V}))$. Could it be the case that UPol and T$\cdot$ generally correspond on $*$-varieties of languages via leaf languages, without any conditions on the $*$-varieties of languages?

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

Acknowledgments. We thank Jean-Eric Pin for his help with Table 1. The fourth and fifth authors were supported by the Alexander von Humboldt foundation.

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, Theoret. Informatics Appl. **33**, 1999, pp. 259–269

[BSS99] B. Borchert, H. Schmitz, F. Stephan: $\mathrm{Leaf}^P(\Delta_2^B) = \Delta_2^p$, 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 Journal of 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, Reading MA, 1990.

[Pi98] J.-E. Pin: *Bridges for Concatenation Hierarchies*, Proc. ICALP, 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.

[SW03] V. L. Selivanov, K. W. Wagner, *A Reducibility for the Dot-Depth Hierarchy*, Technical Report 313, CS Department, University of Würzburg, December 2003

[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.

[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*, Poc. 26th IEEE Symp. on the Foundations of Computer Science, 1985, pp. 1–10