On Existentially First-Order Definable Languages and their Relation to NP

Bernd Borchert
Universität Heidelberg
bb@math.uni-heidelberg.de

Dietrich Kuske*
Technische Universität Dresden
kuske@math.tu-dresden.de

Frank Stephan*
Universität Heidelberg
fstephan@math.uni-heidelberg.de

Abstract

Pin & Weil [PW95] characterized the automata of existentially first-order definable languages. We will use this result for the following characterization of the complexity class NP. Assume that the Polynomial-Time Hierarchy does not collapse. Then a regular language L characterizes NP as an unbalanced polynomial-time leaf language if and only if L is existentially but not quantifierfree definable in $\mathrm{FO}[<, \min, \max, -1, +1]$.

1 Introduction

The following characterization of the first-order definable languages is a classical result of McNaughton & Papert [MP71].

For a language L it is equivalent:

(a) L is definable in first-order logic,
(b) L can be described by a starfree regular expression,
(c) L is accepted by a counterfree finite automaton.

Let first-order-definability in this paper refer to the usual model of defining languages by logical formulas where in addition to the $<$-relation we allow the constants min and max, and the predecessor -1 and successor $+1$ functions. We will call this first order logic $\mathrm{FO}[<, \min, \max, -1, +1]$ *extended first-order logic* FOX. In an analogous fashion like above the subset of the first-order *existentially* definable languages can be characterized. For this we need the following two definitions. According to the terminology in [PW95] a language has *dot-depth*

*Supported by the German Research Foundation (DFG).

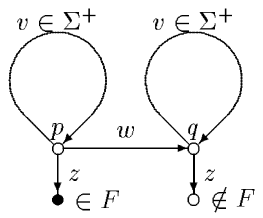

Figure 1: co-UP-pattern

1/2 if it is the finite union of languages of the form $w_1\Sigma^* w_2\Sigma^* w_3 \ldots \Sigma^* w_n$ where $w_1, \ldots, w_n$ are words. The second definition is the following: Say that a finite automaton *contains the* co-UP*-pattern* if there are two reachable states p, q, a nonempty word $v \in \Sigma^+$ and two words $w, z \in \Sigma^*$ such that $p.v = p$, $q.v = q$, $p.w = q$, and $p.z$ is an accepting state and $q.z$ is not an accepting state, see Figure 1 (we will explain the strange name for this pattern later, see Lemma 3.2).

The following equivalence (a) $\Longleftrightarrow$ (b) is due to Thomas [Th82], and (b) $\Longleftrightarrow$ (c) is due to Pin & Weil [PW95].

For a language L it is equivalent:

(a) L is existentially definable in FOX,
(b) L has dot-depth 1/2,
(c) L is accepted by a finite automaton which does not contain the co-UP-pattern.

In a similar fashion, the universally and the quantifierfree definable languages can be characterized.

We will use the known result above for the following application on the characterization of the complexity class NP. Remember that NP is the set of languages L for which there is a nondeterministic polynomial-time Turing machine (NPTM) M such that a word x is in L iff some computation path of the computation tree of $M(x)$ accepts. It is easy to see that for example the following definition also yields the class NP (note that an NPTM defines in an obvious way an order on the computation paths): NP is the set of languages A for which there is an NPTM M such that a word x is in A iff there is an accepting path p of the computation tree of $M(x)$ and the next path $p+1$ in that computation tree does not accept. Yet another example of a characterization of NP is the following: NP is the set of languages A for which there is an NPTM M such that a word x is in A iff there is an accepting path p of the computation tree of $M(x)$ and some later path $p' > p$ in that computation tree of $M(x)$ does not accept.

The reader will notice that by writing a 1 for acceptance and a 0 for rejection the above three examples of definitions of NP can easily be described by languages: the language corresponding to the first standard definition is $\Sigma^*1\Sigma^*$, the language corresponding to the second example is $\Sigma^*10\Sigma^*$, and the language corresponding to the third is $\Sigma^*1\Sigma^*0\Sigma^*$. This concept is the so-called *leaf language approach* of characterizing complexity classes, more precisely: the polynomial-time unbalanced one, see Borchert [Bo95] (the first paper about leaf languages by Bovet et al. [BCS92] used the balanced approach).

We had three examples of languages such that the complexity class characterized by it equals NP. Now the obvious questions is of course: which are exactly the languages which characterize NP? – at least we would like to know which *regular* languages characterize NP. Because the regular language $1\Sigma^*$ characterizes the complexity class P we would, with an answer to that question, solve the P=NP? question. Therefore, we cannot expect a perfect answer. But under the assumption that the Polynomial-Time Hierarchy (PH) does not collapse we are able to give the following answer. The equivalence (a) $\Longleftrightarrow$ (c) from above will be the key for this result.

> Assume that PH does not collapse. Then a regular language L characterizes NP as an unbalanced polynomial-time leaf language if and only if L is existentially but not quantifierfree definable in FOX.

Borchert [Bo95] showed that under the assumption that PH does not collapse there is no class characterized by a regular leaf language properly between P and NP and neither between P and co-NP. In this paper we will prove such an emptiness result for the intervals between NP and co-1-NP and between NP and NP$\oplus$co-NP (see Theorem 3.13).

Finally, we will show a *union-style* result for a correlation of the Boolean Hierarchy over NP and the Boolean Hierachy over the set of first-order existentially definable languages (see Theorem 4.4).

2 Language-Theoretic Results

Let in this paper languages be over the alphabet $\Sigma = \{0, 1\}$. Consider the usual model of defining languages by formulas, see for example [MP71, St94]. In addition to the usual predicate $<$ we allow the constants min and max and the successor and predecessor functions $+1$ and -1. These additional functions are first-order definable in FO[<] but on the level of existential definability the extended logic is strictly more powerful. We will call this extended first-order logic FOX. Finite automata are defined as usually, see for example [MP71, St94], we consider them to be deterministic. The notions *a language has dot-depth 1/2* and *an automaton contains the* co-UP-*pattern* were already defined in the introduction. We already cited the following equivalence, the part (a) $\Longleftrightarrow$ (b) is due to Thomas [Th82]. The equivalence (b) $\Longleftrightarrow$ (c) is not stated by Pin & Weil [PW95] the way we present it but easily follows from their Theorem 5.2.

Theorem 2.1 (Thomas [Th82], Pin & Weil [PW95]) *For a language L it is equivalent:*
(a) *L is existentially definable in* FOX*,*
(b) *L has dot-depth 1/2,*
(c) *L is accepted by a finite automaton which does not contain the* co-UP-*pattern.*

Clearly, a language L is existentially definable iff its complement is universally definable. Let $\mathcal{A}$ be an automaton accepting L. Exchanging the accepting and the rejecting states, one obtains an automaton that accepts the complement of L. Therefore, the following characterization of the universally definable languages is a direct corollary of Theorem 2.1. The UP-pattern is defined like the co-UP-pattern by exchanging "$\in F$" and "$\notin F$".

Theorem 2.2 (Thomas [Th82], Pin & Weil [PW95]) *For a language L it is equivalent:*
(a) *L is universally definable in* FOX*,*
(b) *the complement of L has dot-depth 1/2,*
(c) *L is accepted by a finite automaton which does not contain the* UP-*pattern.*

Another result of this type can be given for quantifierfree definable languages. A language L is *generalized definite* iff there exists a natural number n such that $xyz \in L \iff xy'z \in L$ for any words x, y, y' and z with $|x|, |z| = n$, i.e. the membership of a word in L depends only on the n first and the n last letters of w.

Theorem 2.3 (Thomas [Th82], Pin & Weil [PW95], Borchert [Bo95]) *For a language L it is equivalent:*
(a) *L is quantifierfree definable in* FOX*,*
(b) *L is generalized definite,*
(c) *L is accepted by a finite automaton which does neither contain the* co-UP-*pattern nor the* UP-*pattern.*

Proof. The equivalence $(a) \iff (b)$ is again due to Thomas [Th82, Lemma 2.9]. Since any quantifierfree definable language is existentially and universally definable, the implication $(a) \Rightarrow (c)$ follows from Theorems 2.1 and 2.2. In [Bo95] the proof of Lemma 12 implicitly shows how one can find one of the two patterns (co-UP or UP) in an automaton for a language which is not generalized definite. This proves the implication $(c) \Rightarrow (b)$. $\square$

Remark. For the logic FO[<] it is known that a language is existentially and universally definable iff it is quantifierfree definable (cf. [PW95]). We did not find a proof for the analoguous fact for FOX. Otherwise, Theorem 2.3 would be a direct corollary from Theorems 2.1 and 2.2, the proof of Lemma 12 from

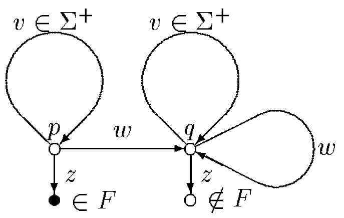

Figure 2: co-NP-pattern

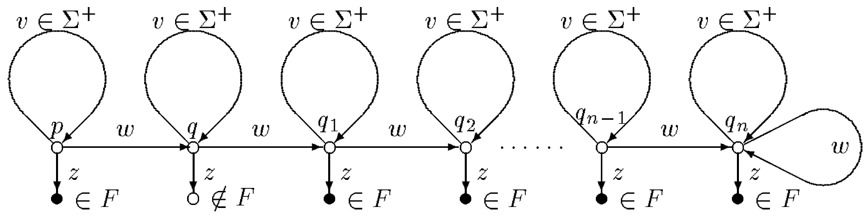

Figure 3: co-1-NP-pattern

[Bo95] would not be needed. Conversally, from Theorems 2.1, 2.2 and 2.3, we get that a language L is quantifierfree definable in FOX iff it is existentially and universally definable in FOX.

We define more patterns: the co-NP-pattern, the co-1-NP-pattern and the counting pattern (Figures 2, 3 and 4). The co-NP-*pattern* looks like the co-UP-pattern, with an additional w-loop at q, i.e. $q.w = q$. In the co-1-NP-*pattern*, $v \neq \epsilon$ and $p.v = p$, $q.v = q$ and $q_i.v = q_i$. Furthermore, $p.w = q$, $q.w = q_1$ and $q_i.w = q_{i+1}$ for $i = 1, 2, \ldots, n-1$ and $q_n.w = q_n$. Finally, $p.z \in F$, $q.z \notin F$ and $q_i.z \in F$ for $i = 1, 2, \ldots, n$. An automaton contains the *counting pattern (for the number n)* if there are two reachable states p, q and two words w, z such that $p.w = q, q.w^{n-1} = p$ and $p.z \in F$ and $q.z \notin F$. Note that a minimal automaton contains the counting pattern iff it is not counterfree. Finally, the NP-pattern and the 1-NP-pattern are defined like the co-NP-pattern and the co-1-NP-pattern, respectively, by exchanging "$\in F$" and "$\notin F$". Note that all

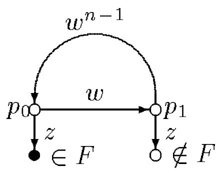

Figure 4: n-counting pattern

patterns besides the UP-pattern and NP-pattern contain the co-UP-pattern as a subpattern (with $v := w^n$ in the counting pattern for the number n). In a straightforward way it is possible to show that besides for the co-1-NP- and the 1-NP-pattern it holds that a finite automaton contains the X-pattern iff the minimal automaton for the language contains the X-pattern.

The following Corollary 2.4 and its dual version Corollary 2.5 will be the bridge from Theorem 2.1 to the results about complexity classes presented in the next section.

Corollary 2.4 *Let $\mathcal{A}$ be the minimal automaton accepting the regular language L. Then L is not existentially definable in* FOX *iff $\mathcal{A}$ contains one of the following patterns:*
(1) the counting pattern,
(2) the co-NP-*pattern,*
(3) the co-1-NP-*pattern.*

Proof. Suppose the automaton $\mathcal{A}$ contains one of these patterns. Then it contains the co-UP-pattern, too. Hence, by Theorem 2.1, L is not existentially definable in FOX.

For the other direction let L be a regular language which is not existentially definable in FOX. It holds that L is not first-order definable at all iff $\mathcal{A}$ is not counterfree iff $\mathcal{A}$ contains the counting pattern. Therefore, for the counting case we have also the other direction. From now on assume that $\mathcal{A}$ is counterfree.

Because L is not existentially definable in FOX the automaton $\mathcal{A}$ contains the co-UP-pattern, i.e. there is a state p and words v, w', z with $v \in \Sigma^+$ such that $p.v = p$, $p.w'v = p.w'$, $p.z \in F$ and $p.w'z \notin F$. Now let $x := w'v'^n$ where n is the number of states of the minimal automaton $\mathcal{A}$ and $p_i = p.x^i$ for $i = 0, 1, 2, \ldots$ Then $p_1 = p.w'v'^n = p.w'$. Furthermore, since $\mathcal{A}$ is counterfree, $p_n = p_n.x$ and $p_i.v = p.(w'v'^n)^i v = p_i$.

Suppose $p_n.z \notin F$ and let $w := x^n$ and $q := p_n$. Then $p.v = p$, $q.v = q$, $p.w = q$, $q.w = q$, $p.z \in F$ and $q.z \notin F$, i.e. we found the co-NP-pattern.

In case $p_n.z \in F$, let i be the maximal natural number with $p_i.z \notin F$. Since $p_1.z \notin F$ we have $i \geq 1$. With $w := x^i$, $q = p.w$ and $q_j := q.w^j$ for $j = 1, 2, \ldots, n$, we obtain the co-1-NP-pattern. □

Note that exchanging the accepting and the rejecting states in the counting pattern results in a counting pattern, again. Therefore, the following is a direct consequence of the corollary above.

Corollary 2.5 *Let $\mathcal{A}$ be the minimal automaton accepting the regular language L. Then L is not universally definable in* FOX *iff $\mathcal{A}$ contains one of the following patterns:*
(1) the counting pattern,
(2) the NP-*pattern,*
(3) the 1-NP-*pattern.*

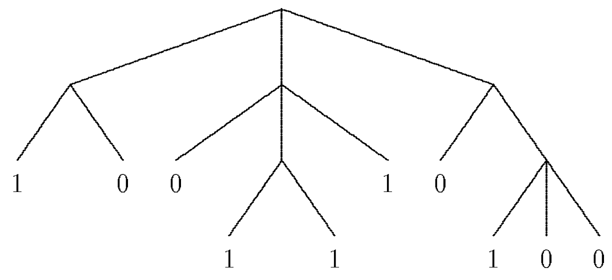

Figure 5: Computation tree (whose yield is 1001110100)

3 An Application to the Leaf Language Characterization of NP

In this section we will give an application of the results from the previous section to the question which leaf languages characterize the complexity class NP. An informal idea of the leaf language concept was already given in the introduction. Let us be a bit more formal (for a detailed definition and more examples and motivation see [Bo95]). Consider the computation tree given by a nondeterministic polynomial-time Turing machine (NPTM) M which runs on an input x. Note that by the original definition of nondeterminism the tree is not necessarily balanced. Also note that there is a natural order on the paths of the tree: if at some configuration there appears nondeterminism the Turing machine M gives a natural linear order on the different possible following configurations: this order is given by the order of the commands in the list representing the transition table. Given the computation tree, label every accepting final configuration with 1 and each rejecting final configuration with a 0. This way we get an ordered tree in which the leaves are labeled by 0 and 1, see Figure 5. The word consisting of the leaf labels read from left to right is called the *yield* of the tree. Let any language $L \in \Sigma^+$ over the alphabet $\Sigma = \{0, 1\}$ be given (because the yield has always positive length we ignore the empty word). The *(unbalanced polynomial-time) leaf language class* $L-$ P is the set of all languages A for which there is an NPTM M such that a word x is in A iff the yield of the computation tree is in L.

Examples. In the introduction we already presented three languages L_1, L_2, L_3 such that $L_1-\mathrm{P} = L_2-\mathrm{P} = L_3-\mathrm{P} = \mathrm{NP}$. As another example, $0^*-\mathrm{P} =$ co-NP. It is easy to see that $1\Sigma^*-\mathrm{P} = \mathrm{P}$. The classes $\mathrm{MOD}_n\mathrm{P}$ for each $n \geq 2$ are defined as $C_n - \mathrm{P}$ where C_n is the set of words w such that the number of 1's in w is not equal 0 modulo n. The two *trivial* leaf languages are $\emptyset$ and Σ^*, it holds $\emptyset-\mathrm{P} = \{\emptyset\}$ and $\Sigma^*-\mathrm{P} = \{\Sigma^*\}$. The *join of* NP *and* co-NP, NP $\oplus$ co-NP, is characterized as $J-\mathrm{P}$ where $J = 0\overline{0^*} \cup 10^*$, it is the smallest

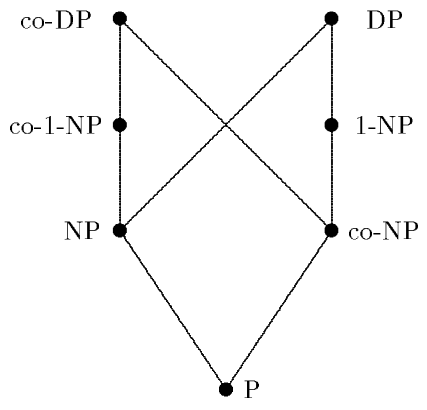

Figure 6: cf. Theorem 3.1

class among all classes $L-$ P containing both NP and co-NP, see [Bo95, Proof of Prop. 2(c)]. Another example is the class UP, a so-called *promise class*: It is the set of all languages A for which there is an NPTM M such that the yield of the computation tree $M(x)$ is in 0^* or 0^*10^* for every x, and a word x is in A iff the yield of the computation tree is in 0^*10^*. We give the definition of UP and its set of complements co-UP just in order to explain the name of the corresponding pattern (see Lemma 3.2); we do not use them for our results. The class 1-NP $= 0^*10^* -$ P and its class of complements co-1-NP $= \overline{0^*10^*} -$ P will be crucial for our main result. The following result gives an idea about their location, see Figure 6. DP is the class consisting of the intersections of classes in NP and co-NP. DP and its set of complements co-DP represent the two classes of the second level in the Boolean Hierarchy over NP, see the following section.

Theorem 3.1 (Blass & Gurevich [BG82], Kadin [Ka88], Chang, Kadin & Rohatgi [CKR95]) *Assume that PH does not collapse. Then* NP, co-NP, 1-NP, co-1-NP, DP, *and* co-DP *are six different classes and the only inclusions which hold are*

$$\begin{array}{rcl} \text{NP} \subset \text{co-1-NP} \subset \text{co-DP}, & \qquad & \text{NP} \subset \text{DP}, \\ \text{co-NP} \subset \text{1-NP} \subset \text{DP}, & \qquad & \text{co-NP} \subset \text{co-DP}. \end{array}$$

The following Lemma 3.2 gives the main connex between patterns in automata and complexity classes, it also explains the names of the patterns.

Lemma 3.2 *Let L be the language accepted by a finite automaton $\mathcal{A}$ where any state is reachable from the initial state.*

(a) *Let* X *be in* {NP,co-NP,1-NP,co-1-NP,UP,co-UP}. *If $\mathcal{A}$ contains the* X-*pattern then* X *is a subset of $L-$* P.

(b) *If $\mathcal{A}$ contains a counting pattern, then* $\mathrm{MOD}_p\mathrm{P}$ *is a subset of* $L-\mathrm{P}$ *for some prime* p.
(c) *If $\mathcal{A}$ contains both the* NP- *and the* co-NP*-pattern, then* $\mathrm{NP}\oplus\text{co-NP}$ *is a subset of* $L-\mathrm{P}$.

Proof. (a) Let the automaton $\mathcal{A}$ contain the co-NP-pattern, see Figure 2. Let the state p be reachable from the initial state by the word a. We want to show that co-NP is included in $L-\mathrm{P}$. It suffices to construct for every NPTM M an NPTM M' such that an input x is accepted by M via the leaf language 0^* iff x is accepted by M via the leaf language L. Given M, let M' be the following machine. On input x M' produces first of all $|a|$ leftmost computation paths with a written on them, this way the yield of the computation tree $M'(x)$ will have the prefix a. Then it simulates the computation of M including the nondeterministic branchings. Everytime M accepts (rejects) it produces the word w (the word v) by extending that computation path of M by $|w|$ (by $|v|$) new computation paths. Finally it produces $|z|$ rightmost computation paths for z. By this construction, $M'(x)$ has a similar computation tree as $M(x)$, besides that every 0 is replaced by a tree for v and every 1 is replaced by a tree for w, and at the leftmost and rightmost part there are computation paths for a and z, respectively. Looking at Figure 2 one can verify: if $M(x)$ does not accept on any computation path then the yield of the computation tree of $M'(x)$ reaches the state $p.z$, i.e. the yield is in L, and if $M(x)$ does accept on at least one computation path then the yield of the computation tree of $M'(x)$ reaches $q.z$, i.e. the yield is not in L. In other words: The yield of the computation tree of $M'(x)$ is in L iff the yield of the computation tree of $M(x)$ is in 0^*. This means that we have the desired inclusion: Let A be a language in co-NP witnessed by a machine M. Then it is witnessed by the machine M' that $A\in L-\mathrm{P}$. In other words, co-NP $\subseteq L-\mathrm{P}$. For the other patterns the proof including the construction of M' is the same.

(b) If the automaton for a language contains a counting pattern for the number n then it contains a counting pattern for a number p which is prime, this was shown in [Bo95, Lemma 6]. And [Bo95, Lemma 7] (using the methods of [BG92]) shows together with the construction from (a) that $\mathrm{MOD}_p\mathrm{P}\subseteq L-\mathrm{P}$ for every language L containing the counting pattern for a prime p.

(c) This follows immediately from (a) and the fact that $\mathrm{NP}\oplus\text{co-NP}$ is the smallest class $L-\mathrm{P}$ containing both NP and co-NP. □

Remark. It seems that Lemma 3.2 above is not possible for the polynomial-time *balanced* leaf language classes. Because this connection between the automata patterns and complexity classes is crucial we can state our main results for unbalanced leaf language classes only.

For the logic FO[<] and the balanced polynomial-time leaf languages, the following Lemma 3.3 and Corollary 3.4 are due to Burtschick & Vollmer (see

[BV96]). They can be considered to be the first results about the close relation of leaf languages for NP and existentially first-order definable languages. Here, we prove it for the slightly different case of unbalanced computation trees and the logic FOX.

Lemma 3.3 (cf. Burtschick & Vollmer [BV96]) *If L is existentially first-order definable in* FOX *then $L-$* P *is a subset of* NP*.*

Proof. Let L be defined by an existential sentence f in FOX, i.e.

$$f = \exists x_1 \ldots \exists x_m \phi(x_1, \ldots, x_m)$$

where ϕ is quantifierfree and contains no more variables than $x_1, \ldots, x_m$. We have to show that for every NPTM M there is a NPTM M' such that the language A accepted by M via the leaf language L is accepted by M' via the leaf language $\overline{0^*}$. M' is defined the following way. Given an input x simulate $M(x)$ including all nondeterministic branchings but do not terminate when the end of a computation path is reached. Instead, memorize the nondeterministic choices made on that computation path as a tupel $p_1 = (a_1, \ldots, a_l)$ where a number a_j denotes that in the j-th nondeterministic situation on that computation path the a_j-th possibility was chosen. After that simulate M including all nondeterministic branchings once more. Coming to the leaf of a computation, again do not stop but memorize the nondeterministic choices as a tupel p_2, and simulate M again. Iterate this procedure k times (k was the number of quantifiers in f). This way a computation tree with k layers of the simulated computation tree for $M(x)$ is obtained. Now a computation path of the whole computation tree $M'(x)$ basically represents a k-tupel $(p_1, \ldots, p_k)$ of computation paths of $M(x)$, each path p_i is represented as a tupel of numbers. Let this k-tupel $(p_1, \ldots, p_k)$ represent in the sentence f above a choice of the positions $(x_1, \ldots, x_k)$. Let $M(x)(p)$ denote the result (1 for acceptance or 0 for rejection) of the computation path p in the computation tree $M(x)$. Note that it is possible to compute the truth value of $\phi(M(x)(p_1), \ldots, M(x)(p_k))$ in polynomial time the following way. To get the value of $M(x)(p)$ for a path p simulate $M(x)$ on p. The constants max and min stand of course for the paths $(1, \ldots, 1)$ and $(m_1, \ldots, m_l)$, respectively, where the m_j are the choices for the rightmost path. Given a path $p = (a_1, \ldots, a_l)$ it is possible to compute the path $p+1$: when p is the rightmost path then $p+1 = p$, otherwise, starting with $a_l, a_{l-1}, \ldots$, check which a_j is the first non-maximal nondeterministic choice, and $p+1$ will be the path $(a_1, \ldots, a_j + 1, 1, \ldots, 1)$. Likewise, given p, one can compute $p-1$. Finally note that it is possible to compute the predicate $p < p'$ just by lexicographic comparison. After the computation of the truth value $\phi(M(x)(p_1), \ldots, M(x)(p_1))$ $M'(x)$ accepts iff $\phi(M(x)(p_1), \ldots, M(x)(p_1))$ is true.

Now that we gave the construction of M' we have to verify that it has the desired properties, i.e. we have to show the language A accepted by M via the leaf language L (where L is given by the sentence f) is accepted by M' via the leaf language $\overline{0^*}$. If $x \in A$ then the yield of $M(x)$ is in L, i.e. there are paths $p_1, \ldots, p_k$ such that $\phi(M(x)(p_1), \ldots, M(x)(p_k))$ evaluates to true. Consider the computation path in $M'(x)$ describing the tupel $(p_1, \ldots, p_k)$. Because $\phi(M(x)(p_1), \ldots, M(x)(p_k))$ evaluates to true, this path in $M'(x)$ is accepting. Therefore, x is in the language accepted by M' via the leaf language $\overline{0^*}$. It remains the case that $x \notin A$, i.e. for all paths $p_1, \ldots, p_k$ the expression $\phi(M(x)(p_1), \ldots, M(x)(p_k))$ is false. But this means that also $M'(x)$ will reject on all paths, in other words: the yield of the computation tree of $M(x)$ is not in $\overline{0^*}$. This finishes the proof that $L\text{-}\,\mathrm{P}$ is a subset of NP if L is existentially definable. □

The language $\overline{0^*}$ is existentially definable even in FO[] by the sentence $\exists x P_1(x)$. Because by definition $\mathrm{NP} = \overline{0^*}\text{-}\,\mathrm{P}$ we have the following corollary. Note that it is weaker than Theorem 3.9 below: even under the assumption that PH does not collapse it still allows other regular languages than the in FOX existentially definable ones to characterize NP.

Corollary 3.4 (cf. Burtschick & Vollmer [BV96]) *The union of all classes $L\text{-}\,\mathrm{P}$ over all existentially first-order definable languages L equals* NP*.*

The following Lemma 3.6 represents one direction of Theorem 3.9; note that it does not use the assumption that PH does not collapse. First we state the following easy result from [Bo95, Lemma 11].

Proposition 3.5 (Borchert [Bo95]) *If L is generalized definite but not trivial then $L\text{-}\,\mathrm{P} = \mathrm{P}$.*

Lemma 3.6 *Let L be existentially definable in* FOX*. If L is quantifierfree definable in* FOX *then $L\text{-}\,\mathrm{P}$ equals one of the classes $\{\emptyset\}$, $\{\Sigma^*\}$, or* P*, otherwise $L\text{-}\,\mathrm{P} = \mathrm{NP}$.*

Proof. If L is quantifierfree definable then it is generalized definite by Theorem 2.3, and we have that $\emptyset\text{-}\,\mathrm{P} = \{\emptyset\}$, $\Sigma^*\text{-}\,\mathrm{P} = \{\Sigma^*\}$ and (by the above Proposition 3.5) $L\text{-}\,\mathrm{P} = \mathrm{P}$ otherwise. Let L be existentially but not quantifierfree definable in FOX. Then it is not universally definable and by Theorem 2.2 its minimal automaton $\mathcal{A}$ does contain the counting, the NP-, or the 1-NP-pattern. But by Corollary 2.4 $\mathcal{A}$ contains neither the counting, the co-NP-, nor the co-1-NP-pattern. Because the co-NP-pattern is a subpattern of the 1-NP-pattern the automaton $\mathcal{A}$ has to contain the NP-pattern. Therefore, by Lemma 3.2, the class NP is contained in $L\text{-}\,\mathrm{P}$. Because L is existentially definable in FOX, $L\text{-}\,\mathrm{P}$ is contained in NP by Lemma 3.3. Therefore, $L\text{-}\,\mathrm{P} = \mathrm{NP}$. □

Now we turn to state the lemma for the other direction of our main Theorem 3.9 below. The lemma relies mainly on Corollary 2.4 which itself is a consequence

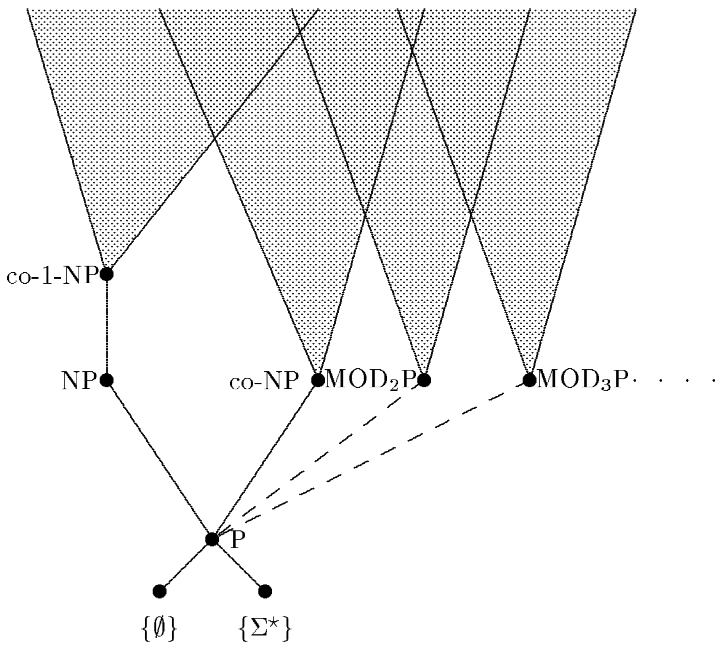

Figure 7: cf. Theorem 3.9

of the characterization 2.1 by Pin & Weil [PW95]. Note that like in Lemma 3.6 (for the other direction for the main theorem) we do not need the assumption that PH does not collapse.

Lemma 3.7 *Let L not be existentially definable in* FOX. *Then $L-$* P *contains at least one of the classes* co-NP, co-1-NP, *or* $\mathrm{MOD}_p\mathrm{P}$ *for some prime p.*

Proof. Let L not be existentially definable in FOX. By Corollary 2.4 the automaton for L contains the counting pattern, the co-NP-pattern, or the co-1-NP-pattern. Therefore, by Lemma 3.2, $L-$ P contains at least one of the classes co-NP, co-1-NP, or $\mathrm{MOD}_p\mathrm{P}$ for some prime p. □

Figure 7 depicts the situation. All the classes $L-$ P with L regular but not existentially definable in FOX are situated somewhere in the gray area.

For the proof of our main result (Theorem 3.9) and also later on we need the following result due to Toda [To91] (see also [Bo95, Lemma 15]).

Theorem 3.8 (cf. Toda [To91]) *Assume that* PH *does not collapse. Then* $\mathrm{MOD}_p\mathrm{P}$ *for a prime p is not contained in* PH.

Now, finally, we can state the following theorem about the close relation of existentially definable languages and NP.

Theorem 3.9 (Main Result) *Assume that* PH *does not collapse and let L be a regular language. Then L* – P = NP *iff L is existentially but not quantifier-free definable in* FOX.

Proof. Let L be existentially but not quantifier-free definable in FOX. Then, by Lemma 3.6 L – P = NP. If L is quantifierfree definable, then L – P $\subseteq$ P, and by the assumption that PH does not collapse we have L – P $\neq$ NP. Finally, let L be not existentially definable in FOX. Then, by Lemma 3.7, L – P contains at least one of the classes co-NP, co-1-NP, or MOD$_p$P for some prime p. If PH does not collapse none of these classes is a subset of NP, see Lemma 3.1 and Theorem 3.8. Therefore, also L – P is not a subset of NP. It follows that L – P $\neq$ NP. □

By duality we have the following result.

Theorem 3.10 *Assume that* PH *does not collapse and let L be a regular language. Then L* – P = co-NP *iff L is universally but not quantifier-free definable in* FOX.

Under the assumption that PH does not collapse we characterized in Theorems 3.9 and 3.10 the regular languages which characterize NP and co-NP respectively, as leaf languages. The following result solves this question for the class P. It follows from the previous results of this paper but actually it could be concluded already by the results of [Bo95], without the characterization by Pin & Weil [PW95] (cf. Theorem 2.1).

Theorem 3.11 *Assume that* PH *does not collapse and let L be a regular language. Then L* – P = P *iff L is quantifier-free definable but not trivial.*

Furthermore, we get the following decidability result as a simple corollary from Theorem 3.9 and the polynomial-time algorithm given by Pin & Weil [PW95].

Corollary 3.12 *Assume that* PH *does not collapse and let L be a regular language. Then the question whether L* – P = NP *is decidable in polynomial time, given the description of an automaton for L.*

In [Bo95] it was shown that under the assumption that PH does not collapse there are no classes L – P properly between P and NP and neither between P and co-NP. Here we have the following extension, the situation is shown in Figure 8.

Theorem 3.13 *Assume that* PH *does not collapse. Then* $\{\emptyset\}$, $\{\Sigma^*\}$, P, NP, co-NP, 1-NP, co-1-NP *and* NP $\oplus$ co-NP *are eight different classes, and the only inclusions which hold are the following ones and their transitive closures. Moreover, each of the following intervals represents a nondensity in the sense that both classes are a class L* – P *with L regular but no class of that kind is located properly between them.*

(a) $\{\emptyset\} \subset$ P,	(b) $\{\Sigma^*\} \subset$ P,
(c) P $\subset$ NP,	(d) P $\subset$ co-NP,
(e) NP $\subset$ co-1-NP,	(f) co-NP $\subset$ 1-NP,
(g) NP $\subset$ NP $\oplus$ co-NP,	(h) co-NP $\subset$ NP $\oplus$ co-NP.

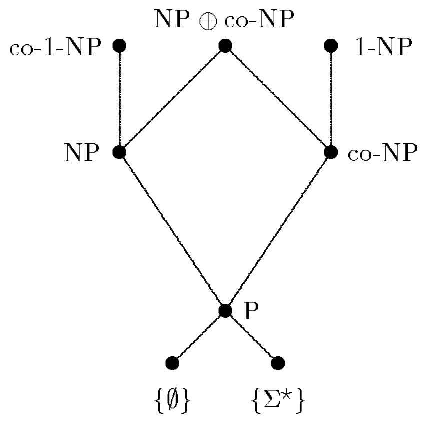

Figure 8: Nondensities in the inclusion order $\{L - P \mid L \text{ regular}\}$.

Proof. From Theorem 3.1 it follows that we have eight different classes and that the only inclusions which hold are the given ones and their transitive closure (remember that the class NP$\oplus$co-NP is contained both in DP and co-DP). Because all classes are contained in PH, which by assumption does not collapse, according to Theorem 3.8 none of them can contain a class MOD$_p$P. The emptiness of the intervals (a) and (b) follows from Theorem 3.11, and the emptiness for (c) and (d) was shown in [Bo95], it also follows from the proofs of Theorems 3.9 and 3.10. For the emptiness of (e) let L be a regular language with NP $\subset L-$ P $\subseteq$ co-1-NP. Then, by Theorems 3.9 and 3.11, L is not existentially definable. Hence $L-$ P contains co-NP, co-1-NP, or MOD$_p$P for some prime p by Lemma 3.7. But MOD$_p$P is not possible, see above, and also co-NP is not possible because then co-NP would be a subset of co-1-NP. Therefore, $L-$ P $=$ co-1-NP. The emptiness of (f) is proven dually. For the emptiness of (g) let L be a regular language with NP $\subset L-$ P $\subseteq$ NP $\oplus$ co-NP. L cannot be existentially definable by Lemma 3.3 and neither it can be universally definable because then $L-$ P $\subseteq$ co-NP and NP would be a subset of co-NP. According to Corollaries 2.4 and 2.5 the minimal automaton $\mathcal{A}$ for L contains (1) the counting pattern, (2) the NP-pattern and the co-NP-pattern, (3) the 1-NP-pattern, or (4) the co-1-NP-pattern (remember that the co-NP-pattern (NP-pattern) is a subpattern of the 1-NP-pattern (co-1-NP-pattern)). Case (1) is not possible, see above. And also cases (3) and (4) are not possible because then by Lemma 3.2(a) 1-NP or co-1-NP, respectively, would be a subset of both DP and co-DP, this contradicts Theorem 3.1. It remains the case (2) and by Lemma 3.2(c) we have that $\mathrm{NP} \oplus \text{co-NP} \subset L-\mathrm{P}$, i.e. $\mathrm{NP} \oplus \text{co-NP} = L-\mathrm{P}$. The proof of (h) works dually. □

4 A Union-Style Result for the Classes of the Boolean Hierarchy over NP

In this paper, we addressed the question what classes are characterized by *single* leaf languages, see for example the main result Theorem 3.9. But there is a different approach which asks about the union of the classes characterized by a *set of* leaf languages, Corollary 3.4 is a typical example. There exist the following examples of results in this union-style (the results were proven for balanced leaf languages but they hold likewise for the unbalanced case). Here $\mathrm{BC}(\Sigma_k^p)$ denotes the Boolean closure of Σ_k^p.

Theorem 4.1 (cf. Hertrampf et al. [HL*93])

$$\begin{array}{lrcl} \text{(a)} & \mathrm{PSPACE} & = & \bigcup\limits_{L \text{ regular}} L-\mathrm{P}, \\ \text{(b)} & \mathrm{BC}(\Sigma_k^p) & = & \bigcup\limits_{L \text{ with dot-depth } k} L-\mathrm{P}. \end{array}$$

Because the languages in the dot-depth hierarchy are exactly the first-order definable languages the following corollary should be mentioned.

Corollary 4.2 (cf. Hertrampf et al. [HL*93])

$$\mathrm{PH} = \bigcup_{L \text{ definable in FOX}} L-\mathrm{P}.$$

Burtschick & Vollmer [BV96] refined these results, the case $k=1$ for (a) is Corollary 3.4.

Theorem 4.3 (cf. Burtschick & Vollmer [BV96])

$$\begin{array}{lrcl} \text{(a)} & \Sigma_k^p & = & \bigcup\limits_{L \text{ is } \Sigma_k\text{-definable in FOX}} L-\mathrm{P}, \\ \text{(b)} & \Pi_k^p & = & \bigcup\limits_{L \text{ is } \Pi_k\text{-definable in FOX}} L-\mathrm{P}. \end{array}$$

In this paper we will add a union-style result for the classes of the Boolean Hierarchy over NP. Let NP(n) denote the level n of the Boolean Hierarchy over NP, i. e.

$$\begin{aligned} \mathrm{NP}(1) &= \mathrm{NP}, \\ \mathrm{NP}(2n) &= \{A \setminus B \mid A \in \mathrm{NP}, B \in \mathrm{NP}(2n-1)\}, \\ \mathrm{NP}(2n+1) &= \{\overline{A \setminus B} \mid A \in \mathrm{NP}, B \in \mathrm{NP}(2n)\}. \end{aligned}$$

Let co-NP(n) be the set of complements of languages in NP(n). Note that DP = NP(2) and co-DP = co-NP(2).

Let $\Gamma(1) = \Gamma$ denote the set of languages of dot-depth 1/2, and define the Boolean Hierarchy over this class by

$$\begin{aligned} \Gamma(2n) &= \{L_1 \setminus L_2 \mid L_1 \in \Gamma, L_2 \in \Gamma(2n-1)\}, \\ \Gamma(2n+1) &= \{\overline{L_1 \setminus L_2} \mid L_1 \in \Gamma, L_2 \in \Gamma(2n)\}. \end{aligned}$$

Let co-$\Gamma(n)$ be the set of complements of languages in $\Gamma(n)$.

We prove the following union-style result, note that for $n = 1$ it equals Corollary 3.4 and Theorem 4.3(a) for $k = 1$. On the other hand, it implies the case $k = 1$ of Theorem 4.1(b) above because the union of all classes NP(n) equals BC(NP) and the union of all classes $\Gamma(n)$ is the set of languages with dot-depth 1.

Theorem 4.4 *For every $n \geq 1$ it holds the following.*

$$\begin{aligned} &\text{(a)} & \mathrm{NP}(n) &= \bigcup_{L \in \Gamma(n)} L\text{-}\mathrm{P}, \\ &\text{(b)} & \text{co-NP}(n) &= \bigcup_{L \in \text{co-}\Gamma(n)} L\text{-}\mathrm{P}. \end{aligned}$$

Proof. First we show by induction on n that L- P $\subseteq$ NP(n) for a language $L \in \Gamma(n)$. By Lemma 3.3 we have L- P $\subseteq$ NP $=$ NP(1) for $L \in \Gamma(1)$, i.e. we have proven the case $n = 1$. Assume, the claim holds for $2n - 1$ and let $L \in \Gamma(2n)$. Then there exist languages $L_1 \in \Gamma$ and $L_2 \in \Gamma(2n-1)$ such that $L = L_1 \setminus L_2$. Let a NPTM M accept a language A via the leaf language L. Let $A_i := \{x \mid \mathrm{yield}(M, x) \in L_i\}$ for $i = 1, 2$. Then $A_1 \in$ NP by Lemma 3.3 and $A_2 \in$ NP($2n - 1$) by the induction assumption. Now it holds that $x \in A$ iff yield(M, x) $\in L$ iff (yield(M, x) $\in L_1$ and yield(M, x) $\notin L_2$) iff ($x \in A_1$ and $x \notin A_2$). Thus, $x \in A$ iff $x \in A_1 \setminus A_2$ proving $A = A_1 \setminus A_2 \in$ NP($2n$). The induction for the odd levels works dually.

For the other direction we show that each class NP(n) is contained in L_n - P for a language $L_n \in \Gamma(n)$. We define L_n explicitly by the following induction. Let K_n be the language $\Sigma^* 0 1^n 0 \Sigma^*$. Note that for each n the language K_n has dot-depth 1/2, i.e. belongs to Γ. Let $L_1 := K_1$, $L_{2n} := K_{2n} \setminus L_{2n-1}$, and let

L_{2n+1} be the complement of the language $K_{2n+1} \setminus L_{2n}$. By definition L_n is an element of $\Gamma(n)$.

We show by induction on n that $\mathrm{NP}(n) \subseteq L_n - \mathrm{P}$ such that for each language $A \in \mathrm{NP}(n)$ this inclusion is witnessed by a machine M whose yield is an element of $\{0, 010, \ldots, 01^n0\}^*$ for every input (we need the second part only as an induction invariant).

For $L_1 = K_1$ such a machine M is given the following way: given a language A which belongs to $\mathrm{NP} = \overline{0^*} - \mathrm{P}$ via a machine M', let M simulate M': if M' rejects let M also reject, and if M' accepts then let M produce 3 computation paths for 010. For $n = 1$ we have proven the induction claim.

Consider the language $L_{2n} = K_{2n} \setminus L_{2n-1}$. Let A be in $\mathrm{NP}(2n)$, i.e. $A = A' \setminus A''$, for a language $A' \in \mathrm{NP}$ and $A'' \in \mathrm{NP}(2n-1)$. Let A' be in NP via the machine M', then A' is in $\Sigma^*01^{2n}0\Sigma^* - \mathrm{P}$ via the machine M which simulates M' and replaces every accepting path by $2n + 2$ paths for $01^{2n}0$ and rejects on a rejecting path. Let M'' be the machine for A'' (via the leaf language L_{2n-1}) whose yield is an element of $\{0, 010, \ldots, 01^{2n-1}0\}^*$ for every input. Finally, let M''' be the machine which first branches nondeterministically and simulates M on the left branch and M'' on the right branch. Note that the yield of M''' is an element of $\{0, 010, \ldots, 01^{2n}0\}^*$ for every input. It remains to show that A is accepted by M''' via the leaf language L_{2n}. It holds $x \in A$ iff ($x \in A'$ and $x \notin A''$) iff ($\mathrm{yield}(M, x) \in K_{2n}$ and $\mathrm{yield}(M'', x) \notin L_{2n-1}$) iff $\mathrm{yield}(M''', x) \in L_{2n}$.

The proof for the odd levels works dually. Case (b) follows immediately. □

5 Open Problems

Under the assumption that PH does not collapse the authors could characterize the regular leaf languages which characterize P, NP, and co-NP, respectively. They would have liked to extend their result to other classes, for example to higher classes of the Polynomial-Time Hierarchy like Σ_k^p. The (unproven) claim for these classes would be the following: If PH does not collapse then a regular language L characterizes Σ_k^p as an unbalanced polynomial-time leaf language if and only if L is Σ_k-definable but not Π_k-definable in FOX. Note that this claim is motivated and supported by Theorem 4.3 due to Burtschick & Vollmer. But Pin & Weil [PW95] remark that there is no algorithm known for the problem whether the language given by an automaton has for example dot-depth 3/2. This means that no automata criterion like the co-UP-pattern criterion is known for dot-depth 3/2, 5/2, etc. But such an criterion seems to be necessary. For the classes of the Boolean Hierarchy like DP the situation does not seem to be as hopeless but the authors could not yet give a characterization.

Acknowledgements

The authors are grateful for comments by Klaus Ambos-Spies, Wolfgang Thomas and Heribert Vollmer.

References

[BG92] R. Beigel, J. Gill: *Counting classes: thresholds, parity, mods, and fewness*, Theoretical Computer Science **103**, 1992, pp. 3–23.

[BG82] A. Blass, Y. Gurevich: *On the unique satisfiability problem*, Information and Control **55**, 1982, pp. 80-88.

[Bo95] B. Borchert: *On the acceptance power of regular languages*, Theoretical Computer Science **148**, 1995, pp. 207–225.

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

[BV96] H.-J. Burtschick, H. Vollmer: *Lindström Quantifiers and Leaf Language Definability*, ECCC Report TR96-005, 1996.

[CG*] J.-Y. Cai, T. Gundermann, J. Hartmanis, L. A. Hemachandra, V. Sewelson, K. Wagner, G. Wechsung: *The Boolean Hierarchy I: structural properties*, SIAM Journal on Computing **17**, 1988, pp. 1232-1252

[CKR95] R. Chang, J. Kadin, P. Rohatgi: *On unique satisfiability and the threshold behaviour of randomized reductions*, Journal of Computer and System Sciences **50**, 1995, pp. 359-373.

[Ka88] J. Kadin: *The Polynomial Time Hierarchy collapses if the Boolean Hierarchy collapses*, SIAM Journal on Computing **17**, 1988, pp. 1263-1282.

[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

[MP71] R. McNaughton, S. Papert: *Counter-Free Automata*, MIT Press, Cambridge, MA, 1971.

[PW95] J.-E. Pin, P. Weil: *Polynomial closure and unambiguous product*, In: 22th ICALP, Lecture Notes in Computer Science 944, Springer Verlag, Berlin, 1995, pp. 348–359. A more detailed version named *Rapport LITP 94-60* from July 1996 can be found on the homepage of the first author.

[St94] H. STRAUBING: *Finite Automata, Formal Logic, and Circuit Complexity*, Birkhäuser, Boston, 1994.

[Th82] W. THOMAS: *Classifying regular events in symbolic logic*, Journal of Computer and System Sciences **25** (1982), pp. 360-376.

[To91] S. TODA: PP *is as hard as the Polynomial-Time Hierarchy*, SIAM Journal on Computing **20**, 1991, pp. 865-877.