

ON EXISTENTIALLY FIRST-ORDER DEFINABLE LANGUAGES AND THEIR RELATION TO NP*

BERND BORCHERT¹, DIETRICH KUSKE² AND FRANK STEPHAN¹

Abstract. Under the assumption that the Polynomial-Time Hierarchy does not collapse we show for a regular language L : the unbalanced polynomial-time leaf language class determined by L equals NP iff L is existentially but not quantifierfree definable in $\text{FO}[\langle, \min, \max, +1, -1\rangle]$. Furthermore, no such class lies properly between NP and co-1-NP or $\text{NP} \oplus \text{co-NP}$. The proofs rely on a result of Pin and Weil characterizing the automata of existentially first-order definable languages.

Résumé. Sous l'hypothèse que la hiérarchie de temps polynomial est stricte, nous montrons que, pour un langage régulier L , la classe des langages de feuille de la hiérarchie polynomiale associée à L est égale à NP si et seulement si L est définissable dans $\text{FO}[\langle, \min, \max, +1, -1\rangle]$ par une formule existentielle comportant au moins un quantificateur. De plus, il n'existe aucune classe de ce type entre NP et co-1-NP ou $\text{NP} \oplus \text{co-NP}$. Les preuves reposent sur un résultat de Pin et Weil qui caractérise les automates des langages définissables par des formules existentielles du premier ordre.

AMS Subject Classification. 03D05, 68Q15, 68Q68.

1. INTRODUCTION

NP is the set of languages A for which there is a nondeterministic polynomial-time Turing machine (NPTM) M such that a word x is in A iff some computation path of the computation tree $M(x)$ 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

Keywords and phrases: Leaf languages, NP, first-order definable languages.

* Supported by the German Research Foundation (DFG).

¹ Universität Heidelberg, Im Neuenheimer Feld 294, 69120 Heidelberg, Germany;
e-mail: bb@math.uni-heidelberg.de fstephan@math.uni-heidelberg.de

² Institut für Algebra, Technische Universität Dresden, 01062 Dresden, Germany;
e-mail: kuske@math.tu-dresden.de

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 [2] (the first paper about leaf languages by Bovet *et al.* [3] used the balanced approach).

We had three examples of languages such that the complexity class characterized by it equals NP. Now an obvious question is of course: which are exactly the languages that 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 an absolute answer. But under the assumption that the Polynomial-Time Hierarchy (PH) does not collapse we are able to give the following answer (see Th. 3.4).

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 the first-order logic $\text{FO}[\langle, \min, \max, +1, -1]$.

The proof heavily relies on the results of Pin and Weil [9] characterizing the ordered syntactic monoids and the finite automata of the existentially definable languages.

Borchert [2] 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 $\text{NP} \oplus \text{co-NP}$ (see Th. 3.5).

2. LANGUAGE-THEORETIC RESULTS

Let in this paper languages be over the alphabet $\Sigma = \{0, 1\}$. We consider the usual model of defining languages by formulas, see for example [8, 10, 11]. Like in [11] we will use the extended first-order logic $\text{FO}[\langle, \min, \max, +1, -1]$ from now on called FOX (with X for “extension”) instead of the usual first-order logic $\text{FO}[\langle]$. The additional functions \min, \max (both 0-ary) and $+1, -1$ (both 1-ary) are definable in $\text{FO}[\langle]$, but on the level of existential definability FOX is strictly more powerful than $\text{FO}[\langle]$. Formulas in FOX are interpreted over words, *i.e.* a variable x indicates a position in a word. In a word of length n there

are the positions $0, \dots, n-1$ and the relation $<$ is the usual “properly-smaller”-relation on the natural numbers. The constant \min stands for position 0, \max stands for position $n-1$. $x+1$ is the usual successor function with the exception $(n-1)+1 := n-1$, likewise $x-1$ is the usual predecessor function. It is possible to define with a FOX-sentence (a sentence is a formula without free variables) a language in $\{0,1\}^+$ in the following usual way. Choose a predicate name Q and build a FOX-sentence f containing it as a unary relation symbol. For any length- n word $w \in \{0,1\}^n$ let Q_w be the predicate $Q_w(i) = \text{true}$ iff w has the letter 1 in position i . For a given word w and a FOX-sentence f containing the predicate symbol Q , the predicate variable Q is interpreted as the predicate Q_w , and now f either evaluates to true or to false. This way, each FOX-sentence f containing a 1-ary relation variable Q defines a language in $\{0,1\}^*$, namely the set of all words w such that f evaluates to true when Q is interpreted as Q_w . A language L is *existentially definable in FOX* if there is an existentially quantified sentence of FOX in prenex normal form defining L . For example, the sentence $\exists x(Q(x))$ is an existentially quantified sentence of FOX and defines the language $\Sigma^*1\Sigma^*$. Another example is the sentence $\exists x\exists y(x < y \wedge Q(x) \wedge \neg Q(y))$ defining the language $\Sigma^*1\Sigma^*0\Sigma^*$. Similarly, the language $\Sigma^*10\Sigma^*$ is defined by the sentence $\exists x(Q(x) \wedge \neg Q(x+1))$. Note that the three languages $\Sigma^*1\Sigma^*$, $\Sigma^*10\Sigma^*$ and $\Sigma^*1\Sigma^*0\Sigma^*$ were already mentioned in the introduction as examples of leaf languages for NP. An example of a universally quantified sentence is $\forall x(\neg Q(x))$, it defines the language 0^* . The set of FOX-definable languages (with any number of quantifier alternations) is a proper subset of the set of regular languages, namely the set of star-free regular languages, see for example [8, 10, 11].

A language L is *generalized definite* iff it is a Boolean combination of finite languages and of languages of the form $v\Sigma^*w$ for some words v and w . A language in Σ^+ has *dot-depth 1/2* if it is the finite union of languages of the form $w_1\Sigma^*w_2\Sigma^*w_3 \dots \Sigma^*w_n$ where w_1, \dots, w_n are nonempty words, *i.e.* it is the finite union of products of generalized definite languages. Finite automata are defined as usually, see for example [8, 10], we consider them to be deterministic.

We 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 where F denotes the set of accepting states of the automaton (the strange name of this pattern is justified by Lem. 3.2). The UP-pattern is defined like the co-UP-pattern by exchanging “ $\in F$ ” and “ $\notin F$ ”.

The part (a) \iff (b) in the following theorem is due to Thomas [11]. The equivalence (b) \iff (c) is a direct reformulation of [9], Theorem 8.15 by Pin and Weil.

Theorem 2.1 (Thomas [11], Pin and Weil [9]). *For a language L it is equivalent:*

- (a) L is existentially (universally, quantifierfree) definable in FOX;
- (b) L has dot-depth 1/2 (the complement of L has dot-depth 1/2, is generalized definite);
- (c) L is accepted by a finite automaton which does not contain the co-UP-pattern (the UP-pattern, neither the co-UP nor the UP-pattern).

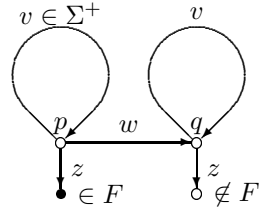


FIGURE 1. co-UP-pattern.

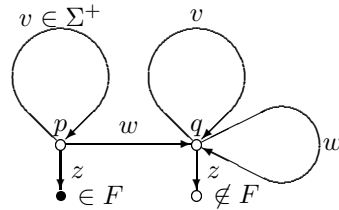


FIGURE 2. co-NP-pattern.

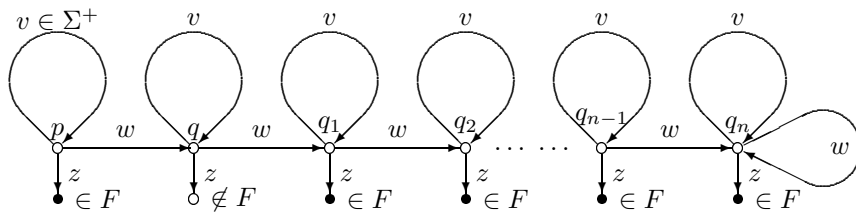


FIGURE 3. co-1-NP-pattern.

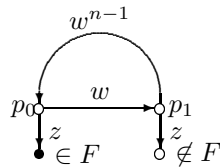


FIGURE 4. n -counting pattern.

We define more patterns: the co-NP-pattern, the co-1-NP-pattern and the counting pattern (Figs. 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 $i = 1, 2, \dots, n - 1$ and $q_n.w = q_n$. Finally, $p.z \in F$, $q.z \notin F$ and $q_i.z \in F$ for $i = 1, 2, \dots, n$. 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 “ $\notin F$ ”.

The following Corollary 2.2 will be the bridge from Theorem 2.1 to the results about complexity classes presented in the next section.

Corollary 2.2. *Let \mathcal{A} be the minimal automaton accepting the regular language L . Then L is not existentially (universally) definable in FOX iff \mathcal{A} contains at least one of the following patterns:*

- (1) the counting pattern;
- (2) the co-NP-pattern (the NP-pattern);
- (3) the co-1-NP-pattern (the 1-NP-pattern).

Proof. We give the proof for the existentially definable languages: suppose the automaton \mathcal{A} contains one of these patterns. Since each of the patterns contains the co-UP-pattern, L is not existentially definable in FOX by Theorem 2.1.

For the other direction let L be a regular language which is not existentially definable in FOX and assume that \mathcal{A} does not contain the counting pattern, *i.e.* that \mathcal{A} is counterfree [8]. 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 $p.v = p, p.wv = p.w, p.z \in F$ and $p.wz \notin F$. With $x := wv^n$ where n is the number of states of the minimal automaton \mathcal{A} one gets $p.x^n = p.x^{n+1}$ since the automaton is counter-free. Then some state $p.x^i$ with $0 \leq i < n$ together with $p.x^n$ and the words v, x^{n-i} and z forms the co-NP- or the co-1-NP-pattern depending on whether $p.x^n z \notin F$ or $p.x^n z \in F$. The characterization of the languages not universally definable works dually. \square

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 [2]). 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 0. This way one gets an ordered tree in which the leaves are labeled by 0 and 1, see Figure 5. The word consisting of the leaf labels

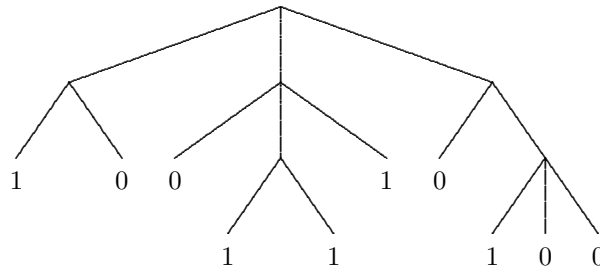


FIGURE 5. Computation tree (whose yield is 1001110100).

read from left to right is called the *yield* of the tree. Let any language $L \subseteq \Sigma^+$ over the alphabet $\Sigma = \{0, 1\}$ be given. Let M be a NPTM. A word x is in the language $M[L]$ iff the yield of the computation tree of M running on input x is in L . Then the (*unbalanced polynomial-time*) leaf language class $L - P$ is the set of all languages $M[L]$ for some NPTM M .

Example. In the introduction we already mentioned that $NP = \Sigma^*1\Sigma^* - P = \Sigma^*10\Sigma^* - P = \Sigma^*1\Sigma^*0\Sigma^* - P$. As another example, $0^* - P = \text{co-NP}$. It is easy to see that $1\Sigma^* - P = P$. The classes $\text{MOD}_n P$ for each $n \geq 2$ are defined as $C_n - P$ where C_n is the set of words w such that the number of 1's in w is not a multiple of n , see [1]. The two *trivial* leaf languages are \emptyset and Σ^+ , it holds $\emptyset - P = \{\emptyset\}$ and $\Sigma^+ - P = \{\Sigma^*\}$. The *join of NP and co-NP*, $NP \oplus \text{co-NP}$, is characterized as $J - P$ where $J = 0\overline{0}^* \cup 10^*$, it is the smallest class among all classes $L - P$ containing both NP and co-NP, see [2], Proof of Proposition 2c. 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 Lem. 3.2); we do not use them for our results. The class $1\text{-NP} = 0^*10^* - P$ and its class of complements $\text{co-1-NP} = \overline{0^*10^*} - P$ will be crucial for our main result. Note that $\text{co-NP} \subseteq 1\text{-NP} \subseteq \text{DP}$ and $\text{NP} \subseteq \text{co-1-NP} \subseteq \text{co-DP}$, where DP and co-DP are the two classes of the second level of the Boolean Hierarchy over NP which is contained in $P(\text{NP})$ and therefore contained in Σ_2^P and Π_2^P , see for example [5]. Remember that both DP and co-DP contain both NP and co-NP. Figure 6 gives an idea about the location of 1-NP and co-1-NP.

The following theorem shows that under the assumption that PH does not collapse also other separations can be proven.

Theorem 3.1. *Assume that PH does not collapse.*

- (1) **(Chang et al. [6])** *1-NP (co-1-NP) is not contained in co-DP (DP), and therefore neither contained in NP, co-NP, $\text{NP} \oplus \text{co-NP}$, nor co-1-NP (1-NP).*

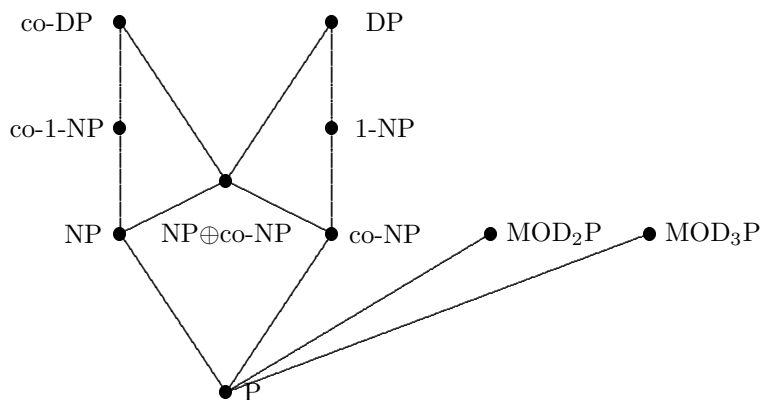


FIGURE 6. 1-NP and co-1-NP.

- (2) (**Toda [12]**) For $n \geq 2$, MOD_nP is not contained in PH and therefore neither contained in NP, co-NP, $\text{NP} \oplus \text{co-NP}$, 1-NP, nor co-1-NP.

The following Lemma 3.2 gives the main link 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.*

- (1) *Let X be in $\{\text{NP}, \text{co-NP}, 1\text{-NP}, \text{co-1-NP}, \text{UP}, \text{co-UP}\}$. If \mathcal{A} contains the X -pattern then X is a subset of $L - \text{P}$.*
- (2) *If \mathcal{A} contains a counting pattern, then MOD_pP is a subset of $L - \text{P}$ for some prime p .*
- (3) *If \mathcal{A} contains both the NP- and the co-NP-pattern, then $\text{NP} \oplus \text{co-NP}$ is a subset of $L - \text{P}$.*

Proof. (1) This proof is very similar to the one of [2], Lemma 5. We give it for the case $X = \text{co-NP}$, only. 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 $L - \text{P}$. It suffices to construct for every NPTM M an NPTM M' such that $M[0^*] = M'[L]$. Given M , let M' be the following machine. On input x , M' initially branches nondeterministically into three subtrees. The left (right) one consists of $|a|$ ($|z|$) computation paths with a (z) written on them, this way the yield of the computation tree $M'(x)$ will have the prefix a (the suffix z , respectively). The middle subtree 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. 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. Thus, the yield is in $a\{v, w\}^*z$. In particular, it reaches one of the states $p.z$ and $q.z$. Looking at Figure 2 one

can easily verify: the yield of the computation tree $M'(x)$ is in L iff the yield of the computation tree $M(x)$ belongs to 0^* , i.e. $M'[L] = M[0^*]$ as desired. For the other patterns the proof (including the construction of M') is the same.

(2) Let \mathcal{B} be the minimal automaton accepting L . With \mathcal{A} , the automaton \mathcal{B} contains an n -counting pattern. Let p be some prime factor of n . Using that \mathcal{B} is minimal, it is easily seen that \mathcal{B} contains a p -counting pattern (where u is replaced by $u^{n/p}$). Now the construction from (a) completes the proof.

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

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 $\text{FO}[\prec]$ and the balanced polynomial-time leaf languages, the following Lemma 3.3 is due to Bertschick and Vollmer [4]. It can be considered to be the first result about the close relation of leaf languages for NP and existentially first-order definable languages. Here, we prove it for our slightly different case of unbalanced computation trees and the logic FOX .

Lemma 3.3 (cf. Bertschick and Vollmer [4]). *If L is existentially first-order definable in FOX then $L - P$ is a subset of NP .*

Proof. Let M be an NPTM and φ a first-order sentence from FOX with m variables (all of which are existentially quantified). We construct an NPTM M' that evaluates φ on the leaf word of the machine M :

this machine M' simulates M m times and memorizes the nondeterministic choices, i.e. the corresponding paths in the computation tree of M . Since these paths represent positions in the yield of M it is then possible to compute the truth value of f in polynomial time.

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

$$f = \exists x_1 \dots \exists x_m \varphi(x_1, \dots, x_m)$$

where φ is quantifierfree and contains no more variables than x_1, \dots, x_m . We have to show that for every NPTM M there is a NPTM M' such that $M[L] = M'[\overline{0^*}]$. 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 tuple $p_1 = (a_1, \dots, a_l)$ 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 tuple p_2 , and simulate M again. Iterate this procedure m times (m was the number of quantifiers in f). This way a computation tree with m 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 m -tuple (p_1, \dots, p_m) of computation paths of $M(x)$, each path p_i is represented as a tuple of numbers. Let this m -tuple (p_1, \dots, p_m) represent in the sentence f above a choice of the positions (x_1, \dots, x_m) . 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 $\varphi(M(x)(p_1), \dots, M(x)(p_m))$ 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 $(1, \dots, 1)$ and (m_1, \dots, m_l) , respectively, where the m_j are the choices for the rightmost path. Given a path $p = (a_1, \dots, a_l)$ 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}, \dots , check which a_j is the first non-maximal nondeterministic choice, and $p + 1$ will be the path $(a_1, \dots, a_j + 1, 1, \dots, 1)$. 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 $\varphi(M(x)(p_1), \dots, M(x)(p_l))$, $M'(x)$ accepts iff $\varphi(M(x)(p_1), \dots, M(x)(p_l))$ is true. Now it is straightforward to check that $M[L] = M'[\overline{0}^*]$ where L is the language given by the sentence f . \square

The language $\overline{0}^*$ is existentially definable by the sentence $\exists x Q(x)$. Since by definition $\text{NP} = \overline{0}^* - \text{P}$, the union of all classes $L - \text{P}$ over all existentially FOX-definable languages L equals NP by the above lemma. Note that this result from Burtschick and Vollmer [4] is weaker than Theorem 3.4 below in a double sense: even under the assumption that PH does not collapse it still allows some existentially definable language to characterize some class properly between P and NP, and still allows not existentially definable regular languages to characterize NP.

Now, finally, we can state our main result (part (2) in the following theorem) about the close relation of existentially definable languages and NP.

Theorem 3.4. *Assume that PH does not collapse and let L be a regular language.*

- (1) $L - \text{P} = \text{P}$ iff L is quantifier-free definable in FOX but not trivial.
- (2) $L - \text{P} = \text{NP}$ iff L is existentially but not quantifier-free definable in FOX.
- (3) $L - \text{P} = \text{co-NP}$ iff L is universally but not quantifier-free definable in FOX.

Proof. (1) By [11] (cf. Th. 2.1), L is quantifier-free definable in FOX iff it is generalized definite. Now (1) is Lemma 11 from [2].

(2) Now let L be existentially but not quantifierfree definable in FOX. Then it is not universally definable by Theorem 2.1, equivalence of (a) and (c). Hence by Corollary 2.2 its minimal automaton \mathcal{A} contains the counting, the NP-, or the 1-NP-pattern. On the other hand, since it is existentially definable, by Corollary 2.2 \mathcal{A} contains neither the counting, the co-NP-, nor the co-1-NP-pattern. But 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 $L - \text{P}$. And since L is existentially definable in FOX, $L - \text{P}$ is contained in NP by Lemma 3.3. Therefore, $L - \text{P} = \text{NP}$.

To show the other implication by contradiction, assume L not to be existentially definable. Then, by Corollary 2.2 and Lemma 3.2, $L - \text{P}$ contains at least one of

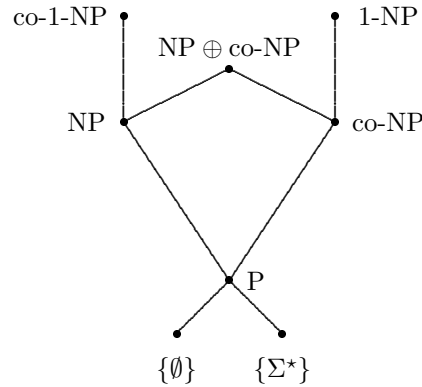


FIGURE 7. Nondensities in the inclusion order $\{L - P \mid L \text{ regular}\}$.

the classes co-1-NP , $\text{MOD}_p P$ for some prime p , or co-NP . Thus, $L - P \neq \text{NP}$ by Theorem 3.1(1), (2) and the assumption that PH does not collapse, respectively.

Part (3) follows immediately from (2). \square

In [2] 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 7.

Theorem 3.5. *Assume that PH does not collapse. Then $\{\emptyset\}$, $\{\Sigma^*\}$, P , NP , co-NP , 1-NP , co-1-NP and $\text{NP} \oplus \text{co-NP}$ are eight different classes. Moreover, each of the intervals depicted as a line between two inclusion-comparable classes in Figure 7 represents a non-density 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.*

Proof. From Theorem 3.1 it follows that we have eight different classes and that the only inclusions which hold are the ones indicated (remember that the class $\text{NP} \oplus \text{co-NP}$ is contained both in DP and co-DP). The emptinesses of the intervals below NP or co-NP were shown in [2]. For the emptiness of the interval between NP and co-1-NP let L be a regular language with $\text{NP} \subset L - P \subseteq \text{co-1-NP}$. Then neither $\text{MOD}_p P$ nor co-NP is contained in $L - P$. Furthermore, by Theorem 3.4, L is not existentially definable. Now $\text{co-1-NP} \subseteq L - P$ follows from Corollary 2.2 and Lemma 3.2 (1) and (2).

Using Lemma 3.2 (3), the emptiness of the interval between NP and $\text{NP} \oplus \text{co-NP}$ follows similarly (one needs the additional observation that $\text{NP} \oplus \text{co-NP}$ is the least leaf language class containing NP and co-NP). \square

4. 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 . So far, no automata criterion like the co-UP-pattern criterion is known for dot-depth $3/2$, $5/2$, etc., see [9]. But such a criterion, as well as a result analogous to Theorem 3.1 (1), 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.

The authors are grateful for comments by Klaus Ambos-Spies, Jean-Eric Pin, Heinz Schmitz, Wolfgang Thomas, and Heribert Vollmer.

REFERENCES

- [1] R. Beigel and J. Gill, Counting classes: Thresholds, parity, mods, and fewness. *Theoret. Comput. Sci.* **103** (1992) 3-23.
- [2] B. Borchert, On the acceptance power of regular languages. *Theoret. Comput. Sci.* **148** (1995) 207-225.
- [3] D.P. Bovet, P. Crescenzi and R. Silvestri, A uniform approach to define complexity classes. *Theoret. Comput. Sci.* **104** (1992) 263-283.
- [4] H.-J. Burtschick and H. Vollmer, Lindström Quantifiers and Leaf Language Definability. *Internat. J. Found. Comput. Sci.* **9** (1998) 277-294.
- [5] J.-Y. Cai, T. Gundermann, J. Hartmanis, L.A. Hemachandra, V. Sewelson, K. Wagner and G. Wechsung, The Boolean Hierarchy I: Structural properties. *SIAM J. Comput.* **17** (1988) 1232-1252.
- [6] R. Chang, J. Kadin and P. Rohatgi, On unique satisfiability and the threshold behaviour of randomized reductions. *J. Comput. System Sci.* **50** (1995) 359-373.
- [7] U. Hertrampf, C. Lautemann, T. Schwentick, H. Vollmer and K. Wagner, On the power of polynomial-time bit-computations, In: *Proc. 8th Structure in Complexity Theory Conference*, IEEE Computer Society Press (1993) 200-207.
- [8] R. McNaughton and S. Papert, *Counter-Free Automata*, MIT Press, Cambridge, MA (1971).
- [9] J.-E. Pin and P. Weil, Polynomial closure and unambiguous product. *Theory Comput. Systems* **30** (1997) 383-422.
- [10] H. Straubing, *Finite Automata, Formal Logic, and Circuit Complexity*, Birkhäuser, Boston (1994).
- [11] W. Thomas, Classifying regular events in symbolic logic. *J. Comput. System Sci.* **25** (1982) 360-376.
- [12] S. Toda, PP is as hard as the Polynomial-Time Hierarchy. *SIAM J. Comput.* **20** (1991) 865-877.

Communicated by J.E. Pin.

Received June, 1998. Accepted January, 1999.