Empty Alternation *

Klaus-Jörn Lange

Institut für Informatik, TU München
Arcisstr.21, D-80290 München
e-mail: lange@informatik.tu-muenchen.de

Klaus Reinhardt

Institut für Informatik, Universität Stuttgart
Breitwiesenstr.22, D-70565 Stuttgart
e-mail: reinhard@informatik.uni-stuttgart.de

Abstract. We introduce the notion of *empty alternation* by investigating alternating automata which are restricted to empty their storage except for a logarithmically space-bounded tape before making an alternating transition. In particular, we consider the cases when the depth of alternation is bounded by a constant or a polylogarithmic function. In this way we get new characterizations of the classes AC^k, SAC^k and P using a push-down store and new characterizations of the class Θ_2^P using Turing tapes.

1 Introduction

Alternation as introduced by Chandra, Kozen, and Stockmeyer in [CKS81] proved to be a very fruitful and versatile parallel concept in complexity theory. It has been combined with both automata and grammar models. The language families described by alternating devices typically coincide with sequential time or space classes. Furthermore, several well-known hierarchies defined by iterated relativizations of sequential time and space have been characterized by constant depth-bounded alternation, that is by bounding the number of transitions between existential and universal configurations within each computation path by a constant. Let us review these relations for logarithmically space-bounded Turing machines, polynomially time-bounded Turing machines, and for the intermediate model of polynomially time-bounded auxiliary push-down automata:

Logarithmic Space: There are two types of relativizations of nondeterministic logarithmic space, that of Ladner and Lynch (see [LL76]) and that of Ruzzo, Simon, and Tompa (see [RST84]), which is more restricted by prohibiting nondeterministic oracle queries: while writing on its query tape the oracle machine has to work deterministically. While the Ladner-Lynch relativization leads to the Polynomial Hierarchy of Meyer and Stockmeyer , Ruzzo-Simon-Tompa relativization characterizes the "late" Logarithmic Space Hierarchies (see [Lan86]). Thus, in the case of logarithmically space-bounded computations, alternation is related to the weaker kind of relativization.

Polynomial Time: The two types of relativization — Ladner-Lynch vs. Ruzzo-Simon-Tompa — coincide in the case of nondeterministic polynomially time-bounded computations and lead to the Polynomial Hierarchy of Meyer and

* This research was supported by DFG-SFB 342, Teilprojekt A4 "KLARA".

Stockmeyer. In [JKL89] the Deterministic Polynomial Hierarchy was introduced by iterating restricted relativizations of deterministic polynomial time equipped with an nondeterministic polynomial time oracle. This hierarchy does not seem to coincide with the Polynomial Hierarchy but rather shows some similarities to the Logarithmic Space Hierarchies - in particular, its collapse. If we look at alternating polynomial time, bounding the depth of alternation by a constant leads to the Polynomial Hierarchy. We see that in the case of polynomially time-bounded computations, alternation is related to the stronger kind of relativization and the corresponding hierarchy.

Polynomial Time of Auxiliary Push-Down Automata: Finally, let us consider polynomially time-bounded two-way push-down automata which are augmented by a logarithmically space-bounded working tape (see [Coo71], [Sud78]). Bounding the depth of alternation of this device by a constant again leads to the Polynomial Hierarchy, but (as in the Ladner-Lynch case) shifted by one level ([JK89], [Bun87]). It is possible to define both a Ladner-Lynch-like and a Ruzzo-Simon-Tompa-like relativization of this device. In the latter case not only nondeterminism but also access to the push-down store has to be prohibited during the generation of oracle queries. Here again, the Ladner-Lynch approach leads to the Polynomial Hierarchy, while the corresponding Ruzzo-Simon-Tompa hierarchy behaves like the Logarithmic Space Hierarchies, in particular it collapses on its first level down to $LOG(CFL)$, the class of problems reducible to context-free languages.

Thus, we see that in all three cases we have two types of relativizations and corresponding hierarchies, a stronger relativization which leads to the Polynomial Hierarchy and a weaker relativization which collapses on a low level of the corresponding hierarchies.[2] Unless the considered type of storage is just a logarithmically space-bounded working tape, alternation is related to the stronger hierarchy.

In this paper we introduce a new kind of restricted alternation, which we call *Empty Alternation*. It is characterized by the property that in moments of alternation all tapes or stacks have to be empty except for one working tape of logarithmic size. While "full" alternation usually corresponds to the stronger relativization and the corresponding hierarchy, empty alternation will correspond to the weaker one. Thus, the fact that full alternation and Empty Alternation coincide if there is no additional storage but a logspace tape explains the correspondence of logspace alternation to the Ruzzo-Simon-Tompa relativization.

This paper is organized as follows. In section 2 we will give a short overview on complexity classes obtained by bounding depths of alternations by constants

[2] Generally, the former allows opposite relativizations of certain relations, while the latter often is "positive" in the sense, that the existence of certain relativizations would settle the unrelativized case. For example, there are oracle sets A and B such that $LOG^A = NLOG^A$ and $LOG^B \neq NLOG^B$ if we use the relativization of Ladner and Lynch. Considering the relativization of Ruzzo, Simon, and Tompa, however, the existence of any oracle set C separating LOG^C from $NLOG^{\langle C \rangle}$ would imply $LOG \neq NLOG$!

or polylogarithmic functions. In section 3 the notion of Empty Alternation is introduced and related to full alternation, yielding new characterizations of well-known classes like AC^k, SAC^k, or $\Theta_2^P := L^{NP} = P^{NP[\log]} = P_{\|}^{NP}$.

2 Alternation

We assume the reader to be familiar with the basic notions and results of complexity theory as they are contained in [HoUl79] or [BDG88] and [BDG90]. In particular, *LOG, NLOG, P, NP,* and *PSPACE* denote the sets recognizable in (nondeterministic) logarithmic space, (nondeterministic) polynomial time, and polynomial space. Let $LOG(X)$ denote the class of all languages logspace many-one reducible to a class X. By *NAuxPDA–TIME*(*pol*) and *DAuxPDA–TIME*(*pol*) we denote the classes of languages recognizable by nondeterministic and deterministic auxiliary push-down automata (see [Coo71]) in polynomial time. In addition, we use the STA-notation: $STA(f, g, h)$ denotes the set of all languages recognizable by alternating Turing machines, which are $f(n)$ space-bounded, $g(n)$ time-bounded, and make no more than $h(n) - 1$ alternations. Thus, $h(n) = 1$ covers nondeterministic languages and by convention $h(n) = 0$ denotes the deterministic case. By AC^k (respectively SAC^k) we denote the class of languages recognized by circuit families of polynomial size, $O(\log^k n)$ depth, and of unbounded (respectively semi-unbounded) fan-in (see [Ven87]).
In the following we will consider alternating logspace machines, push-down automata (both with and without polynomial time bound), and polynomially time- or space-bounded Turing machines with two-way input and bounded depth of alternation which are always equipped with a logarithmically space-bounded working tape. This, of course, does not change the computational power in the case of polynomially time- or space-bounded Turing machines. But it will be useful when introducing empty alternation. Thus, for $X \in \{LOG,\ PDA–TIME(pol), PDA,\ P,\ PSPACE\}$ and a function g, where we allow the cases that g is a constant or that g is unbounded which will be indicated by the symbol ω, let $A\Sigma_g^{s(n)}X$ (respectively $A\Pi_g^{s(n)}X$) denote the set of all languages recognized by alternating X-machines augmented with an $s(n)$ space-bounded working tape, which make $g(n) - 1$ alternations starting in an existential (respectively universal) state.
Characterizations of these classes with $s(n) = \log n$ are collected in Table 1.
Here, the result $STA(pol, -, pol) = PSPACE$ in Column 5 is due to the result $STA(f, -, g) \subseteq DSPACE(f(f+g))$ for $f(n) \geq \log(n)$ of Borodin (cited according to [CKS81]). Most of the results may be found in [LSL84], [CKS81], [JK89], [Sud78], [Imm88], and [Sze88]. The remaining cases for push-down automata are settled by the following theorems, which we will state without proof. First, the *Auxiliary PDA Theorem* from [LSL84] can be extended to:

Theorem 1. *For $s(n) \geq \log(n)$ and $a(n) \geq 0$ computable within space $s(n)$ it holds:*

$$\bigcup_c A\Sigma_{a(n)}SPACE(2^{cs(n)}) \subseteq A\Sigma_{1+a(n)}^{s(n)}PDA$$

Step of alter-nation	without push-down store	with pushdown store		with polynomial tape	
		polynomial time bound	without time bound	polynomial time bound	without time bound
determ.	L	$LOG(DCFL)$ [Sud78]	P		*PSPACE*
$A\Pi_1$		[Bo et al 88]	P [Coo71]	*co–NP*	
$A\Sigma_1$	NL	$LOG(CFL)$ [Sud78]		*NP*	
$A\Pi_2$	[Imm88] [Sze88]	*co–NP* [JK89] [LSL84]		Π_2^p	
$A\Sigma_2$		*NP* [JK89]	*PSPACE* [LSL84]	Σ_2^p	
$\vdots$		$\vdots$		$\vdots$	
$A\Pi_k$		Π_{k-1}^p [JK89]		Π_k^p	
$A\Sigma_k$		Σ_{k-1}^p [JK89]		Σ_k^p	[Imm88] [Sze88]
$\vdots$		$\vdots$		$\vdots$	
$A\Sigma_{\log(n)}$	AC^1	$\Sigma_{\log(n)}^p$ ⟨Corollary 4⟩		$\Sigma_{\log(n)}^p$	
$\vdots$	$\vdots$	$\vdots$		$\vdots$	
$A\Sigma_{\log^k(n)}$	AC^k	$\Sigma_{\log^k(n)}^p$ ⟨Corollary 4⟩	⟨Corollary 2⟩	$\Sigma_{\log^k(n)}^p$	[CKS81]
$\vdots$	$\vdots$	$\vdots$		$\vdots$	
$A\Sigma_\omega$	P [CKS81]	*PSPACE* [JK89]	EXPTIME [LSL84]	*PSPACE* [CKS81]	EXPTIME [CKS81]

Table1. Complexity classes of automata with space-bounded tape

As a consequence with $A\Sigma_{a(n)}^{\log} PDA \subseteq \bigcup_c NSPACE(a(n)2^{c\log(n)})$ [LSL84] we get:

Corollary 2. $A\Sigma_{\log^k}^{\log} PDA = PSPACE$

Second, we give the extension of a result in [JK89] to the case of non-constant depth of alternation:

Theorem 3. *For $s(n) \geq \log(n)$ and $a(n) \geq 1$ computable within space $s(n)$ it holds:*

$$\bigcup_c A\Sigma_{a(n)}^{s(n)} TIME(2^{cs(n)}) = \bigcup_c A\Sigma_{1+a(n)}^{s(n)} PDA\text{–}TIME(2^{cs(n)})$$

Corollary 4. $A\Sigma_{\log^k}^{\log} POL = A\Sigma_{\log^k}^{\log} PDA\text{–}TIME(pol)$

3 Empty Alternation

If we add bounded alternation to logarithmically space-bounded Turing machines, the result is comparatively small: we just get $NLOG$. The situation changes completely, if we slightly increase the power of the underlying machine model. For example, consider the rather small class $DAuxPDA\text{–}TIME(pol)$, which is contained in both SC^2 and NC^2, and hence in P and in $POLYLOGSPACE$. For $DAuxPDA\text{–}TIME(pol)$ the addition of depth-bounded alternation yields already the Polynomial Hierarchy ([JK89],[Bun87]). If we take a closer look to this phenomenon, we see that the underlying machine has now the possibility of pushing polynomially many bits in an existential or universal way onto the push-down store. These bits are then popped, i.e. read one-way, and evaluated again with the help of bounded alternation. This phenomenon and its explanation led us to the concept of empty alternation: we augment machines with several storage types and then add alternation under the restriction that in moments of alternation (during transitions between existential and universal configurations) all auxiliary memories are empty and all transferred information is contained in the state and on a logarithmically space-bounded working tape.
In the following, for $X \in \{LOG,\ PDA\text{–}TIME(pol),\ PDA,\ P,\ PSPACE\}$ and a function g, where we again admit the cases that g is a constant or that g is unbounded, let $EA\Sigma_g^{\log}X$ denote the set of all languages recognized by logspace Turing machines augmented with storage of type X, which make $g(n)-1$ empty alternations. The main results of this chapter are collected in Table 2, which is the "empty" analogue of Table 1.

3.1 Empty Alternation and Push-down Automata

In this section we study the concept of empty alternation for machines equipped only with an additional logspace tape (i.e. for 'unaugmented' machines), with one push-down store while maintaining a polynomial time bound, and with one push-down store without any restriction of the running time. Obviously, for unaugmented machines empty alternation coincides with (full) alternation, which yields $EA\Sigma_g^{\log}LOG = A\Sigma_g^{\log}LOG$ for $g \in O(1) \cup \{\log^k n, \omega\}$. Thus, $EA\Sigma_\omega^{\log}LOG = P$. The following result shows that this relation holds even for machines augmented with an unrestricted push-down store.

Theorem 5. $EA\Sigma_\omega^{\log}PDA \subseteq P$

Proof. According to [Coo71] it can be decided in polynomial time for two configurations K_1, K_2 with empty pushdown store, whether $K_1 \overset{*}{\underset{M}{\vdash}} K_2$ without alternation. If M without loss of generality only stops with empty pushdown store, then for input x it can be calculated recursively for all such configurations in the calculation of $M(x)$, whether there is a partial accepting subtree under that configuration. Because this has to be calculated once for each of the polynomially many configurations, this can be done in polynomial time. ∎

Step of alternation	without push-down store	with pushdown store		with polynomial tape	
		polynomial time bound	without time bound	polynomial time bound	without time bound
determ.	L	$LOG(DCFL)$ [Sud78]	P		$PSPACE$
$A\Pi_1$		[Bo et al 88]	P [Coo71]	co–NP	
$A\Sigma_1$	NL	$LOG(CFL)$ [Sud78]		NP	
$EA\Pi_2$	[Imm88]				
$EA\Sigma_2$	[Sze88]				
$\vdots$					
$EA\Pi_k$					
$EA\Sigma_k$		⟨Theorem 8⟩			
$\vdots$					
$EA\Sigma_{\log(n)}$	AC^1 ⟨Theorem 9⟩				
$\vdots$	$\vdots$				
$EA\Sigma_{\log^k(n)}$	AC^k ⟨Theorem 9⟩			Θ_2^p	
$\vdots$	$\vdots$				
$EA\Sigma_\omega$	[CKS81] P ⟨Corollary 7⟩		⟨Corollary 6⟩	⟨Theorem 10⟩	⟨Theorem 11⟩

Table2. Complexity classes of automata with logarithmically space-bounded tape and empty alternation

Contrast this with $A\Sigma_\omega^{\log} PDA = EXPTIME$ in [LSL84]. By Cook's characterization of P by auxiliary push-down automata in [Coo71] Theorem 5 yields

Corollary 6. *$EA\Sigma_g^{\log} PDA = P$, for $g \in \{O(1), \log^k, \omega\}$.*

Another consequence of Theorem 5 follows from
$P = EA\Sigma_\omega^{\log} LOG \subseteq EA\Sigma_\omega^{\log} PDA\text{–}TIME(pol)$:

Corollary 7. *$EA\Sigma_\omega^{\log} PDA\text{–}TIME(pol) = P$*

On the other hand using the result of [Bo et al 88], it is easy to see that we have a collapse to $LOG(CFL)$ for constant bounded empty alternation:

Theorem 8. *$EA\Sigma_k^{\log} PDA\text{–}TIME(pol) = LOG(CFL)$, for each k.*

In an obvious way it is possible to introduce *semi-unbounded* empty alternation (compare with [Ven87]) which yields classes named $SEA\Sigma_g^{\log}X$. Now, the nature of polylogarithmically bounded empty alternation of polynomially time-bounded push-down automata is characterized by

Theorem 9. $EA\Sigma^{\log}_{\log^k n}PDA\text{–}TIME(pol) = AC^k$
and $SEA\Sigma^{\log}_{\log^k n}PDA\text{–}TIME(pol) = SAC^k$, *for* $k \geq 1$.

Proof. The inclusions from right to left are obvious. An empty alternating push-down automaton with $h \cdot \log(n)$ space-bounded tape can be simulated by an AC^k (respectively SAC^k) circuit which calculates with $O(|x|^{2h+2})$ sub-circuits for every pair of surface-configurations $\langle K_1, K_2\rangle$ with empty push-down store whether K_2 is reachable from K_1 ($K_1 \overset{*}{\underset{M}{\vdash}} K_2$) without any alternation. Since this can be done in $LOG(CFL)$ or $LOG(co\text{–}CFL)$ it can also be done by a SAC^1-circuit because of [Bo et al 88]. Then the circuit recursively calculates in each level j of the $\log^k$ levels for every surface-configuration K_i the bit $c_{i,j}$, which is 1, if K_i has an accepting tree of depth j. For an accepting (rejecting) configuration, this is 1 (0), for an existential configuration it is

$$c_{i,j} = \bigvee_l (c_{l,j-1} \wedge \langle K_i, K_l\rangle) \quad \text{and it is} \quad c_{i,j} = \bigwedge_l (c_{l,j-1} \vee \overline{\langle K_i, K_l\rangle})$$

for an universal configuration. In case of a semi-unbounded push-down automaton there are only finitely many l's in a conjunction, so the whole construction results in a SAC^k circuit. ∎

This result indicates the comparatively small computational power of Empty Alternation when dealing with polynomially time-bounded auxiliary push-down automata: the addition of a push-down store does not increase the power of a logspace machine as long as the depth of alternation is at least logarithmically growing.
But the other direction holds too in the sense that empty alternating push-down automata without two-way input and without logspace working tape generate languages complete for $EA\Sigma^{\log}_{a(n)}PDA\text{–}TIME(pol)$ (respectively $SEA\Sigma^{\log}_{a(n)}PDA\text{–}TIME(pol)$), which is shown in [Rei92]. This generalizes the equations

$NAuxPDA\text{–}TIME(pol) = LOG(CFL)$ and
$DAuxPDA\text{–}TIME(pol) = LOG(DCFL)$

of Sudborough [Sud78]; it may be interpreted in the sense that in $EA\Sigma^{\log}_{a(n)}PDA\text{–}TIME(pol)$-automata a one-way push-down part may be separated from a two-way logspace part. This decomposition is also possible for fully alternating push-down automata as shown in [Rei89] and [Rei90].
Similar results can be obtained with alternating grammars. For the alternating context free grammars in [Mor89] we have $LOG(ACFL^{left}_{\lambda-free}) = PSPACE$ according to [CT90], but for the alternating (even λ-free) alternating context free grammars in [Rei89] $LOG(CFL\Sigma_\omega) = EXPTIME$ holds. A surprising result of [Rei92] is that alternating linear grammars generate the complete languages corresponding to empty alternation: $LOG(LIN\Sigma_\omega) = P$, $LOG(LIN\Sigma_{\log^k n}) = AC^k$ and $LOG(LIN\Sigma_k) = NL$ for $k \geq 1$.

3.2 Empty Alternation and Turing Tapes

If we consider machines with two or more auxiliary push-down stores, it is easy to see that from the aspect of complexity these are equivalent to Turing tapes. That is why we will consider empty alternation of polynomial time and of polynomial space in this subsection. In the case of polynomial time we will get a characterization of the class $\Theta_2^p := L^{NP}$. This class was named and characterized by Wagner, who gave several representations of the classes $\Theta_{k+1}^p := L^{\Sigma_k^p}$.
In the following $P^{A[\log]}$ refers to classes defined by polynomial time-bounded oracle machines, which are allowed to ask at most $O(\log(n))$ queries and $\mathrm{P}_{\|}^{A}$ refers to classes defined by polynomial time-bounded oracle machines, which are allowed to ask a polynomial number of queries in parallel. With the help of Wagner's characterizations $L^{NP} = P^{NP[\log]} = P_{\|}^{NP}$ in [Wag90], we show one of our main results:

Theorem 10. $EA\Sigma_2^{\log}P = EA\Sigma_\omega^{\log}P = \Theta_2^p$

Proof. $EA\Sigma_\omega^{\log}P \subseteq P_{\|}^{NP}$:
Let $K(x)$ be the set of those configurations of an $EA\Sigma_\omega^{\log}P$-machine M on input x, where the not logarithmic space-bounded tape is empty. Thus $|K(x)|$ is bounded by a polynomial. We consider the language

$$L_1 := \{(x, K_1, K_2) \mid K_1, K_2 \in K(x) \text{ and } K_1 \overset{*}{\underset{M}{\vdash}} K_2 \text{ without any alternation } \}.$$

Obviously, we have $L_1 \in NP$. With one parallel round of queries to L_1 we can compute the complete reachability relation

$$R(x) := \{(K_1, K_2) \mid K_1, K_2 \in K(x) \text{ and } K_1 \overset{*}{\underset{M}{\vdash}} K_2 \text{ without any alternation } \}.$$

Then the partial accepting subtrees for all the configurations in $K(x)$ can be computed like in the proof of Theorem 5. We assume w.l.o.g. that the tape, which is not logarithmic space-bounded is empty, if the machine accepts or stops. The simulation accepts, if the start configuration belongs to a partial accepting subtree.
$P^{NP[\log]} \subseteq EA\Sigma_2^{\log}P$:
Let $L \in P^{NP[\log]}$ by an oracle machine M with oracle SAT. An $EA\Sigma_2^{\log}P$-machine A simulates M twice:
In the first simulation A starts in an existential state and simulates the deterministic steps of M. If M asks the i-th oracle question '$v_i \in SAT$?', then A guesses the answer and stores it as the i-th bit on the logarithmic space-bounded tape. If the answer 'Yes' is guessed, then A simulates the NP-machine B for SAT on v_i and rejects, if B rejects. If the answer 'No' is guessed, the verification is postponed to the second phase of alternation. Then A continues the simulation of M. If M accepts, then A alternates into an universal state and starts the second phase of the simulation by simulating again the deterministic steps of M. If M asks the i-th oracle question '$v_i \in SAT$?', then A looks up the answer from the

logarithmic space-bounded tape. If the answer is 'No', then A simulates universally the *co–NP*-machine C for $UNSAT$ on v_i and rejects, if C rejects. Then A continues the simulation of M. If M accepts, then A accepts. ■

As we see, a hierarchy defined by bounded empty alternation of polynomially time-bounded machines would collapse on its second level down to Θ_2^P. (Even totally unbounded empty alternation collapses to alternation depth two!) But this then is precisely the Deterministic Polynomial "Hierarchy" of [JKL89]. This again shows the close relationship of empty alternation to weak relativizations and hierarchies compared to the closeness of full alternation to more powerful ones.
Finally, we shortly consider the case of polynomial space. By a result of Borodin (cited in [CKS81]) we have $A\Sigma_{pol}^{PSPACE} = PSPACE$. While $A\Sigma_{\omega}^{PSPACE} = EXPTIME$, empty alternation does not lead beyond $PSPACE$:

Theorem 11. $EA\Sigma_{\omega}^{\log}PSPACE = PSPACE$

Proof. It can be decided with polynomial space for two configurations K_1, K_2, whether $K_1 \overset{*}{\underset{M}{\vdash}} K_2$ without alternation. If M without loss of generality only stops with all auxiliary tapes empty, then for input x it can be calculated recursively for all such configurations in the calculation of $M(x)$, whether there is a partial accepting subtree under that configuration. Because this has to be calculated once for each of the polynomially many configurations, this can be done in polynomial space. ■

Discussion and Open Questions

We introduced the concept of *Empty Alternation* as a restriction of the usual 'full' alternation and exhibited close connections to questions of how to relativize complexity classes and about the collapses of hierarchies. As a result new representations of many well-known complexity classes have been obtained. Since alternation is a very powerful mechanism, it seems reasonable not only to restrict the concept itself, but also the device it is applied to. In this way, relations between formal languages and complexity could be generalized. This leaves open to investigate these relations with respect to other models of formal language theory. First candidates should here be all types of stack automata, since the relations between their deterministic, nondeterministic, (fully) alternating, and auxiliary versions show a very similar pattern to that of push-down automata. Another interesting question would be to determine both an alternation type and an automaton model which together characterize the NC^k classes not as time classes, but directly by the depth of alternation.

Acknowledgment We thank Volker Diekert, Werner Ebinger, Birgit Jenner, Anca Muscholl and Peter Rossmanith for many helpful remarks, Prof. Dr. W. Knödel, who made this joint work possible and an anonymous referee, who helped us to simplify the proof of Theorem 10.

References

[BDG88] J. Balcázar and J. Diáz and J. Gabárro: Structural Complexity Theory I; Springer 1988.

[BDG90] J. Balcázar and J. Diáz and J. Gabárro: Structural Complexity Theory II; Springer 1990.

[Bo et al 88] A.Borodin. S.A. Cook, P.W.Dymond, W.L. Ruzzo, M.Tompa; Two applications of complementation via inductive counting, 3rd Structure in Complexity Theory.

[Bun87] G. Buntrock: On the Robustness of the Polynomial Time Hierarchy, Technische Universität Berlin, Technischer Bericht, Nr.: 87-11,1987.

[CKS81] A.K. Chandra, D.C. Kozen, L.J. Stockmeyer; Alternation, Journ. of the ACM 28,1 (1981), 114-133.

[Coo71] S.A. Cook: Characterizations of push-down machines in terms of time bounded computers, Journ. of the ACM 18,1 (1971), 4-18.

[CT90] Z.-Z. Chen, S. Toda: Grammatical Characterizations of P and PSPACE, transactions of the IEICE, Sep. 1990.

[HoUl79] J.E. Hopcroft, J.D. Ullmam: Introduction to Automata Theory, Languages and Computation, Addison-Wesley, 1979.

[Imm88] N. Immerman: Nondeterministic space is closed under complementation, SIAM Journ. Comput. 15, 5 (1988), 935-938.

[JK89] B. Jenner, B. Kirsig: Characterizing the polynomial hierarchy by alternating auxiliary push-down automata. Theoretical Informatics and Applications, 1989, 87-99.

[JKL89] B. Jenner, B. Kirsig, K.-J. Lange: The Logarithmic Alternation Hierarchy Collapses, Information and Computation 80 (1989), 269-288.

[LL76] R. Ladner and N. Lynch: Relativization of questions about log space computability, Math. Systems Theory 10 (1976),19-32.

[LSL84] R.E. Ladner,L.J.Stockmeyer,R.J. Lipton: Alternation bounded auxiliary pushdown automata, Information and Control 62 (1984), 93-108.

[Lan86] K.-J. Lange: Two Characterizations of the Logarithmic Alternation Hierarchy, Proc. of 12th MFCS 233, LNCS, Springer 1986, 518–526.

[Mor89] E. Moriya: A grammatical characterization of alternating push-down automata, TCS 67 (1989), 75-85.

[Rei89] K. Reinhardt: Hierarchien mit alternierenden Kellerautomaten, alternierenden Grammatiken und finiten Transducern, Diplomarbeit, Universität Stuttgart, 1989.

[Rei90] K. Reinhardt: Hierarchies over the context-free Languages, Proc. of 6th IMYCS, LNCS, 464, Springer 1990, 214-224.

[Rei92] K. Reinhardt. Counting and empty alternating pushdown automata, Proc. of 7th IMYCS, pages 198–207, Smolenice Castle, Tschechoslowakei, 1992.

[RST84] W. Ruzzo and J. Simon and M. Tompa: Space – Bounded hierarchies and probabilistic computations, JCSS 28 (1984), 216-230.

[Sze88] R. Szelepcsenyi: The Method of forced enumeration for nondeterministic automata, Acta Informatica 26 (1988), 96-100.

[Sud78] I.H. Sudborough: On the tape complexity of deterministic context-free languages, Journ. of the ACM 25, 3 (1978), 405-414.

[Wag90] K. Wagner. Bounded query classes. SIAM Journ. Comput. 19(1990), 833–846.

[Ven87] H. Venkateswaran: Properties that characterize LOGCFL. In Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, 141-150, New York,May 1987.

This article was processed using the LaTeX macro package with the LLNCS document class.