Open access • Book Chapter • DOI:10.1007/BFB0035838

Characterizing the Polynomial Hierarchy by Alternating Auxiliary Pushdown Automata — Source link

Birgit Jenner, Bernd Kirsig

Institutions: University of Hamburg

Published on: 11 Feb 1988 - Symposium on Theoretical Aspects of Computer Science

Topics: Deterministic pushdown automaton, Embedded pushdown automaton, Pushdown automaton, Nested word and Polynomial hierarchy

INFORMATIQUE THÉORIQUE ET APPLICATIONS

BIRGIT JENNER
BERND KIRSIG

Characterizing the polynomial hierarchy by alternating auxiliary pushdown automata

Informatique théorique et applications, tome 23, n° 1 (1989), p. 87-99
<http://www.numdam.org/item?id=ITA_1989__23_1_87_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 23, n° 1, 1989, p. 87 à 99)

CHARACTERIZING THE POLYNOMIAL HIERARCHY BY ALTERNATING AUXILIARY PUSHDOWN AUTOMATA

by Birgit JENNER [1] and Bernd KIRSIG [1]

Abstract. – *An alternating auxiliary pushdown hierarchy is defined by extending the machine model of the Logarithmic Alternation Hierarchy by a pushdown store while keeping a polynomial time bound. Although recently it was proven by Borodin* et al. *that the class of languages accepted by nondeterministic logarithmic space bounded auxiliary pushdown automata with a polynomial time bound is closed under complement* [1], *it is shown that, surprisingly, the further levels of this alternating auxiliary pushdown hierarchy coincide level by level with the Polynomial Hierarchy. Furthermore, PSPACE can be characterized by simultaneously logarithmic space and polynomial time bounded auxiliary pushdown automata with unbounded alternation. Finally, it is shown that both results generalize to arbitrary space bounds.*

Résumé. – *Nous définissons une hiérarchie pour les machines alternantes à pile en étendant le modèle de la hiérarchie logarithmique alternante par une pile auxiliaire, tout en gardant une borne en temps polynomial. Bien qu'il a été démontré récemment par Borodin* et al. *que la classe des langages acceptée par un automate à pile non déterministe en espace logarithmique et en temps polynomial est fermé par complémentation, nous montrons que les niveaux supérieurs de cette hiérarchie des machines alternantes à pile auxiliaire coïncident niveau à niveau avec la hiérarchie polynomiale – un résultat assez surprenant. De plus nous montrons que PSPACE est caractérisé par des automates bornés en espace logarithmique et en temps polynomial, en permettant une alternation non bornée. Finalement nous montrons que les deux résultats généralisent aux espaces non bornés.*

1. INTRODUCTION

Recently some very interesting and especially unexpected results in complexity theory answered questions which have been unsolved for a long time such as the collapse of the Logarithmic Alternation Hierarchy [8, 7], the collapse of the Logarithmic Oracle Hierarchy [11], and most of all the closure of nondeterministic space classes under complementation [6, 14]. In this area,

[1] Universität Hamburg, Fachbereich Informatik, Rothenbaumchaussee 67/69, D-2000 Hamburg 13, R.F.A.

Informatique théorique et Applications/Theoretical Informatics and Applications
0296-1598 89/01 87 13/$3.30/

too, falls the surprising result by Borodin *et al.* that even $LOG(CFL)$, the closure of the context-free languages under logarithmic space bounded many-one reductions, is closed under complementation [1]. This class has been introduced and investigated by Sudborough [13] and can be interpreted as an extension of NSPACE(log n). Obviously, Borodin *et al.*'s result means:

$$A\Sigma_1^{\mathscr{L}} PDA_{pt} = A\Pi_1^{\mathscr{L}} PDA_{pt} = LOG(CFL),$$

where $A\Sigma_k^{\mathscr{L}}$ PDA (resp., $A\Pi_k^{\mathscr{L}} PDA$) denotes the class of languages acceptable by $A\Sigma_k^{\mathscr{L}}$-(resp., $A\Pi_k^{\mathscr{L}}$)-machines with additional pushdown store and *pt* means restriction to polynomial time. An $A\Sigma_k^{\mathscr{L}}$-(resp., $A\Pi_k^{\mathscr{L}}$)-machine is a logarithmic space bounded alternating Turing machine which starts in an existential (resp., universal) configuration and alternates at most $k-1$ times during each computation (*cf.* [3]). Note that a language L is in the k-th level of the Logarithmic Alternation Hierarchy $A\Sigma_k^{\mathscr{L}}$ if and only if there is an $A\Sigma_k^{\mathscr{L}}$-machine accepting L.

Now, knowing that the Logarithmic Alternation Hierarchy $\{A\Sigma_k^{\mathscr{L}}, A\Pi_k^{\mathscr{L}} \mid k \geq 0\}$ collapses at its first level [6] one wonders if with the result of [1] an analogously defined alternating auxiliary pushdown hierarchy $\{A\Sigma_k^{\mathscr{L}} PDA_{pt}, A\Pi_k^{\mathscr{L}} PDA_{pt} \mid k \geq 0\}$ (in the following called the "$A\Sigma^{\mathscr{L}} PDA_{pt}$-Hierarchy") collapses at its first level, too. This does not seem to be the case, since we can show that this would imply $\mathrm{NP} = P = LOG(CFL)$, because logarithmic space bounded alternating auxiliary pushdown automata which are restricted to polynomial time and a fixed number of alternations during computations exactly characterize the Polynomial Hierarchy $\{\Sigma_k^{\mathscr{P}}, \Pi_k^{\mathscr{P}} \mid k \geq 0\}$ (*cf.* [12]). In particular, we show:

$$\forall k \geq 1: \quad \Sigma_k^{\mathscr{P}} = A\Sigma_{k+1}^{\mathscr{L}} PDA_{pt},$$

which means that the k-th level of the Polynomial Hierarchy is just the $k+1$st level of the $A\Sigma^{\mathscr{L}} PDA_{pt}$-Hierarchy.

This result not only yields another characterization of the Polynomial Hierarchy besides the three well-known characterizations by bounded iteration of nondeterministic polynomial time Turing reductions, bounded quantification of P-predicates, and alternation bounded polynomial time machines, but also shows that with one additional alternation the working tape of the latter machines can be restricted to logarithmic space plus a pushdown store.

Furthermore, it is proven that $\mathrm{PSPACE} = A\Sigma_\omega^{\mathscr{L}} PDA_{pt}$ (the subscript ω denoting unbounded alternation). Since $\mathrm{PSPACE} = \mathrm{AP}$ [3] this shows that alternating polynomial time machines accept the same languages if their

working tape is restricted to logarithmic space plus a pushdown store. Note that this question for machines without alternation $P=?\ DAPDA(\log n)_{pt}$ $(=LOG(DCFL)$ [13]) is still open. Since EXPTIME equals $A\Sigma_{\omega}^{\mathscr{L}} PDA$ [9], this result further sheds some new light on the relationship between PSPACE and EXPTIME which is known to be APSPACE, the class of languages recognized by alternating polynomial space bounded machines [3]. The difference between these two classes can now be stated as the difference between logarithmic space bounded alternating auxiliary pushdown automata with a polynomial time bound and those without such a time bound.

Finally, we show that both results generalize to arbitrary space-constructible bounds.

2. THE $A\Sigma^{\mathscr{L}} PDA_{pt}$-HIERARCHY

We assume the reader to be familiar with the standard notation and results of complexity theory in [5]. In addition, we denote the complement of a language L by $Co-L$ and for a class of languages $\mathscr{A}$ we define $CO-\mathscr{A}=\{CO-L \mid L\in\mathscr{A}\}$.

In what follows we first define the $A\Sigma^{\mathscr{L}} PDA_{pt}$-Hierarchy. As outlined in the introduction this hierarchy is defined in terms of simultaneously polynomial time and logarithmic space bounded alternating auxiliary pushdown automata. To our knowledge there has not yet been an investigation of auxiliary pushdown automata which are *both* time bounded *and* alternating. The concept of an (nondeterministic, resp., deterministic) auxiliary pushdown automaton (with arbitrary space bound and without any time bound) was introduced and investigated in [2]. In [13] investigations of logarithmic space bounded auxiliary pushdown automata which are restricted to a polynomial time bound followed. There it was shown that the class of languages which are recognized by nondeterministic (resp., deterministic) such automata coincides with the closure of the context-free languages (resp., deterministic context-free languages) under logarithmic space reductions:

$$NAPDA(\log n)_{pt}=LOG(CFL) \quad [13],$$

$$DAPDA(\log n)_{pt}=LOG(DCFL) \quad [13].$$

On the other hand, alternating pushdown automata were introduced and investigated in [3] and their auxiliary versions (without a time bound) in [9].

For the definition of alternating $S(n)$-space bounded auxiliary pushdown automata and the family of languages which are accepted by them ($ALT\text{-}AUX\text{-}PDA(S(n))$) we refer to [9; Sect. 3] or [10].

For given monotone $S(n) \geqq \log n$, $T(n) \geqq n$ let $A\Sigma_\omega^{S(n)} pda_{T(n)}$ denote an $S(n)$-space bounded $T(n)$-time bounded alternating auxiliary pushdown automaton and $A\Sigma_\omega^{\mathscr{L}} pda_{pt}$ any such machine which satisfies $S(n) = O(\log n)$ and $T(n) = O(n^k)$, $k \geqq 1$. Let $A\Sigma_\omega^{\mathscr{L}} PDA_{pt} : = \{ L(M) \mid M \text{ is an } A\Sigma_\omega^{\mathscr{L}} pda_{pt} \}$.

For a given $A\Sigma_\omega^{\mathscr{L}} pda_{pt}$ M define for all $k \geqq 1$: M is an $A\Sigma_k^{\mathscr{L}} pda_{pt}$ (resp., $A\Pi_k^{\mathscr{L}} pda_{pt}$) if M starts in an existential (resp., universal) state and makes at most $k-1$ alternations between existential and universal states during each computation.

With this we define the $A\Sigma^{\mathscr{L}} PDA_{pt}$-Hierarchy as follows (for the definition of $DAPDA(\log n)_{pt}$, *cf.* [13]):

DEFINITION 2.1: *For $k \geqq 1$ let*

$$A\Sigma_k^{\mathscr{L}} PDA_{pt} : = \{ L(M) \mid M \text{ is an } A\Sigma_k^{\mathscr{L}} pda_{pt} \}$$

and

$$A\Pi_k^{\mathscr{L}} PDA_{pt} : = \{ L(M) \mid M \text{ is an } A\Pi_k^{\mathscr{L}} pda_{pt} \},$$

and for sake of completeness let

$$A\Sigma_0^{\mathscr{L}} PDA_{pt} : = A\Pi_0^{\mathscr{L}} PDA_{pt} : = DAPDA(\log n)_{pt}.$$

Then the $A\Sigma^{\mathscr{L}} PDA_{pt}$-Hierarchy is the set $\{ A\Sigma_k^{\mathscr{L}} PDA_{pt}, A\Pi_k^{\mathscr{L}} PDA_{pt} \mid k \geqq 0 \}$.

Note that as in the case of the Logarithmic Alternation Hierarchy the base class of the $A\Sigma^{\mathscr{L}} PDA_{pt}$-Hierarchy is characterized by deterministic machines.

Furthermore, note that $A\Sigma_\omega^{\mathscr{L}} PDA_{pt}$ and $A\Sigma_k^{\mathscr{L}} PDA_{pt}$, $k \geqq 0$, are closed under logarithmic space reductions, as can be shown by standard techniques.

Obviously, the $A\Sigma^{\mathscr{L}} PDA_{pt}$-Hierarchy shares the structure of the Logarithmic Alternation Hierarchy. It holds for all $k \geqq 0$:

$$A\Pi_k^{\mathscr{L}} PDA_{pt} = Co\text{-}A\Sigma_k^{\mathscr{L}} PDA_{pt}$$

and

$$A\Sigma_k^{\mathscr{L}} PDA_{pt} \cup A\Pi_k^{\mathscr{L}} PDA_{pt} \subseteq A\Sigma_{k+1}^{\mathscr{L}} PDA_{pt} \cap A\Pi_{k+1}^{\mathscr{L}} PDA_{pt}.$$

Note that obviously $NSPACE(\log n) \subseteq A\Sigma_k^{\mathscr{L}} PDA_{pt}$ for all $k \geqq 1$, and note that the relationship between the base level of our hierarchy $DAPDA(\log n)_{pt}$ and NSPACE $(\log n)$ is still unsolved. None of the two classes is known to contain the other and no language to separate them has been found yet.

Since an $A\Sigma_1^{\mathscr{L}} pda_{pt}$ is just a nondeterministic polynomial time bounded log n-space bounded auxiliary pushdown automaton, it holds for the first

level of our hierarchy (*cf.* [13]):

$$A\Sigma_1^{\mathscr{L}} PDA_{pt} = NAPDA(\log n)_{pt} = LOG(CFL)$$

and

$$A\Pi_1^{\mathscr{L}} PDA_{pt} = Co\text{-}NAPDA(\log n)_{pt} = Co\text{-}LOG(CFL).$$

Now, these classes have recently been shown by Borodin *et al.* to coincide:

PROPOSITION 2.2: [1] $A\Sigma_1^{\mathscr{L}} PDA_{pt} = A\Pi_1^{\mathscr{L}} PDA_{pt}$.

This is surprising, since in the following section we are going to show that the further levels of the $A\Sigma^{\mathscr{L}} PDA_{pt}$-Hierarchy coincide level by level with the Polynomial Hierarchy [12] $\{\Sigma_k^{\mathscr{P}}, \Pi_k^{\mathscr{P}} \mid k \geqq 0\}$ jumping over the base level $\Sigma_0^{\mathscr{P}} = P$. Thus it seems that to the $A\Sigma^{\mathscr{L}} PDA_{pt}$-Hierarchy the usual collapse arguments cannot be applied all the way down to the first level. Hence we have a much stronger result than just the inclusion of the k-th level of the Logarithmic Alternation Hierarchy in the k-th level of the $A\Sigma^{\mathscr{L}} PDA_{pt}$-Hierarchy which is obvious since an $A\Sigma_k^{\mathscr{L}} pda_{pt}$ is nothing but an $A\Sigma_k^{\mathscr{L}}$-machine with additional pushdown store.

3. THE MAIN RESULT

We are going to prove first that the $A\Sigma^{\mathscr{L}} PDA_{pt}$-Hierarchy essentially coincides with the Polynomial Hierarchy. This is done by showing each inclusion separately. We use two lemmata.

LEMMA 3.1: $A\Sigma_{k+1}^{\mathscr{L}} PDA_{pt} \subseteq \Sigma_k^{\mathscr{P}}$ *for all* $k \geqq 0$.

Proof: The proof is by induction on k. The basis $k=0$ is immediate, since $A\Sigma_1^{\mathscr{L}} PDA_{pt}$ equals $LOG(CFL)$ [13], which is well known to be a subset of P. Assume for induction that $A\Sigma_{k+1}^{\mathscr{L}} PDA_{pt} \subseteq \Sigma_k^{\mathscr{P}}$, and hence $A\Pi_{k+1}^{\mathscr{L}} PDA_{pt} \subseteq \Pi_k^{\mathscr{P}}$. Let $L \in A\Sigma_{k+2}^{\mathscr{L}} PDA_{pt}$, $L \subseteq X^*$, $\$ \notin X$, and let M be the $A\Sigma_{k+2}^{\mathscr{L}} pda_{pt}$ accepting L. W.l.o.g. we assume M to alternate at least once.

Now, define

$$B_M := \{ w\$c \mid w \in X^*,\ c \text{ is a universal configuration of M, and } M \text{ started in configuration } c \text{ accepts } w \}.$$

As can easily be verified, B_M can be recognized by an $A\Pi_{k+1}^{\mathscr{L}} pda_{pt}$ that simulates M starting in configuration c. Thus $B_M \in A\Pi_{k+1}^{\mathscr{L}} PDA_{pt}$ which implies $B_M \in \Pi_k^{\mathscr{P}}$ by the induction assumption. But now, obviously, a word $w \in X^*$ is an element of L if and only if there is a universal configuration c which is reachable from the initial configuration of M on a computation

path with existential configurations only and $c \in B_M$. Thus, an NP oracle machine N with oracle set B_M can accept L as follows: On input $w \in X^* N$ simulates M until M alternates to a universal configuration c. Then N copies c onto the oracle tape, queries the oracle, and accepts if and only if the oracle answer is "YES", i. e., iff $c \in B_M$. Thus, $L \in NP(B_M) \subseteq NP(\Pi_k^{\mathscr{P}}) = \Sigma_{k+1}^{\mathscr{P}}$. As L was arbitrary chosen from $A\Sigma_{k+2}^{\mathscr{L}} PDA_{pt}$, we conclude $A\Sigma_{k+2}^{\mathscr{L}} PDA_{pt} \subseteq \Sigma_{k+1}^{\mathscr{P}}$, thus proving the lemma. ▲

For the second lemma we first need some preparation.

For any $k \geqq 1$ let us define B_k as the set of all boolean formulas $F(X_1, \ldots, X_k)$ over $\{0, 1, \neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ such that $(\exists X_1)(\forall X_2) \ldots (Q_k X_k)[F(X_1, \ldots, X_k) = 1]$ where Q_k stands for $\forall$ if k even and else for $\exists$, X_i stands for the sequence of variable symbols $x_{i1}, x_{i2}, \ldots$ and $\exists X_i$ stands for $\exists x_{i1} \exists x_{i2} \ldots$ (and $\forall X_i$ for $\forall x_{i1} \forall x_{i2} \ldots$), and $F(X_1, \ldots, X_k)$ denotes a boolean formula containing no variable symbol x_{ij} with $i > k$. Let $3\,CNF$ ($3DNF$) denote the set of boolean formulas in conjunctive (disjunctive) normal form such that F is $C_1 \wedge C_2 \wedge \ldots \wedge C_m$ ($C_1 \vee C_2 \vee \ldots \vee C_m$) where, for $1 \leqq i \leqq m$, C_i is a disjunction (conjunction) of at most three literals. Define

$$B_\omega : = \bigcup_{k=1}^{\infty} B_k.$$

Then as shown by Stockmeyer [12; Theorem 4.1 and Theorem 5.1] it holds:

PROPOSITION 3.2: [12] *Let* $k \geqq 1$.

(i) *If* k *is odd (even) then* $B_k \cap 3\,CNF$ ($B_k \cap DNF$) *is log-complete in* $\Sigma_k^{\mathscr{P}}$.

(ii) $B_\omega \cap 3\,CNF$ *is log-complete in PSPACE.*

Now we are prepared to complete our main result.

LEMMA 3.3: $\Sigma_k^{\mathscr{P}} \subseteq A\Sigma_{k+1}^{\mathscr{L}} PDA_{pt}$ *for all* $k \geqq 1$.

Proof: By Proposition 3.2 (i) we know that for odd (even) k, $k \geqq 1$, $B_k \cap 3\,CNF$ ($B_k \cap 3\,DNF$) is $\Sigma_k^{\mathscr{P}}$-complete. Since $A\Sigma_{k+1}^{\mathscr{L}} PDA_{pt}$ is closed under logarithmic space many-one reductions, we only have to show $B_k \cap 3\,CNF \in A\Sigma_{k+1}^{\mathscr{L}} PDA_{pt}$ (resp., $B_k \cap 3\,DNF \in A\Sigma_{k+1}^{\mathscr{L}} PDA_{pt}$) for odd (resp., even) k.

Let k be odd. We will come up with an $A\Sigma_{k+1}^{\mathscr{L}} pda_{pt}$ M that recognizes $B_k \cap 3\,CNF$.

On input w M first checks if w is an encoding of a boolean formula in $3\,CNF$ with variable sequences X_1 to X_k. Obviously, this can be done deterministically in log-space and hence in polynomial time. If w is the encoding of a formula $F(X_1, \ldots, X_k)$ M has to accept iff $\exists X_1 \forall X_2 \ldots \exists X_k$:

$F(X_1, \ldots, X_k)=1$. Now M does the following: For $i=1$ to k M guesses an assignment for X_i (universally, if i even, and existentially, if i odd) and stores it (and the variable) in the pushdown, thereby separating one entry from the other by using a separation symbol \$. (The pushdown will now contain for any X_i a sequence like the following: $x_{i1}\ \$1\ \$x_{i2}\ \$0\$\ldots\$x_{il_i-1}\$0x_{il_i}\$1\$$.)

As k is odd X_k is stored existentially. And since any variable symbol x_{ij} occurs exactly once on the pushdown for this linear time suffices. Now M alternates and universally guesses a clause in F and checks for all variables in the pushdown store (one after the other) – thereby emptying the pushdown store – if it or its negation is contained in that clause and if its assignment satisfies that clause. If one of the variables satisfies the clause M accepts otherwise M rejects. As M operates universally all clauses are checked and M accepts iff $F(X_1, \ldots, X_k)$ is satisfied. Checking one clause to be satisfied requires at most linear time as in any clause there are only 3 variables and M only has to empty the pushdown. Thus the total amount of time used by M is polynomial. As the assignment to $X_1, \ldots, X_k$ was generated alternating existentially and universally M accepts iff $\exists X_1 \forall X_2 \ldots \exists X_k$: $F(X_1, \ldots, X_k)=1$ thereby using exactly k alternations.

The proof for even k and $B_k \cap 3\,DNF$ is similar. In this case the variables and their assignments are stored by M as described above, but since k is even, now X_k is stored universally and with one additional alternation M can check (existentially) if there is one clause that can be satisfied which suffices to check whether the entire $3\,DNF$-formula is satisfied. ▲

Remark: By considering the case $\mathrm{NP}=\Sigma_1^{\mathscr{P}}=A\Sigma_2^{\mathscr{L}}PDA_{pt}$ as an example of the idea of the proof, it becomes evident in what way the space used by a polymial time bounded machine is equivalent to a (polynomial space bounded) pushdown store plus one alternation. An NP-machine which recognizes $SAT \cap 3\,CNF$, the set of all satisfiable formulas in conjunctive normal form with at most three literals per clause, guesses an assignment to the variables onto its tape and then checks for all clauses – one after the other – whether they are all satisfied, thereby using the information stored on its tape *more often than just once.*

In contrast, an $A\Sigma_2^{\mathscr{L}}pda_{pt}$ guesses an assignment to the variables onto its pushdown store and then – since reading erases the information stored on the pushdown and the information therefore can be used only once – the automaton checks by *alternating*, i. e., by using its universal guessing mechanism, whether they are all satisfied.

With Lemma 3.1 and 3.3 our main theorem is now immediate.

THEOREM 3.4: $\Sigma_k^{\mathscr{P}} = A\Sigma_{k+1}^{\mathscr{L}} PDA_{pt}$ *for all* $k \geqq 1$. ▲

Theorem 3.4. provides us with another machine model that characterizes the Polynomial Hierarchy on and above its first level. Note that the first level of the hierarchy defined by these alternating machines is not NP but $LOG(CFL)$.

The ω-jump of the Polynomial Hierarchy is PSPACE, since $B_\omega \cap 3CNF = \bigcup_k^\infty B_k \cap 3CNF$ is log-space complete for this class [12]. The following theorem shows that even this set can be recognized by simultaneously polynomial time and log-space bounded alternating auxiliary pushdown automata, granted unlimited alternation.

As is well known, alternating polynomial time equals polynomial space:

PROPOSITION 3.5: [3] $AP = PSPACE$.

With this the inclusion $A\Sigma_\omega^{\mathscr{L}} PDA_{pt} \subseteq$ PSPACE is obvious, since $A\Sigma_\omega^{\mathscr{L}} pda_{pt}$s are restricted alternating polynomial time machines. But using the same idea as in the proof of Lemma 3.3 we can show, too that PSPACE is contained in $A\Sigma_\omega^{\mathscr{L}} PDA_{pt}$:

THEOREM 3.6: $PSPACE = A\Sigma_\omega^{\mathscr{L}} PDA_{pt}$.

Proof: It remains to show the left-to-right inclusion. Since $A\Sigma_\omega^{\mathscr{L}} PDA_{pt}$ is closed under logarithmic space many-one reductions with Proposition 3.2 (ii) it suffices to show that $B_\omega \cap 3CNF \in A\Sigma_\omega^{\mathscr{L}} PDA_{pt}$. But $w \in B_\omega \cap 3CNF \Leftrightarrow \exists k : w \in B_k \cap 3CNF$, w.l.o.g. k odd, w encoding of $F(X_1, \ldots, X_k)$. This implies that the assignment for the variables $X_1, \ldots, X_k$ can be pushed onto the pushdown store using $k-1$ alternations, and with one more alternation it can be verified that all clauses are satisfied by the technique described in Lemma 3.3. Hence $B_\omega \cap 3CNF$ can be accepted by an $A\Sigma_\omega^{\mathscr{L}} pda_{pt}$. ▲

Note that with results of Ladner, Lipton, and Stockmeyer [10] we know that for logarithmic space bounded alternating auxiliary pushdown automata *without* a time bound there is no hierarchy above the second level:

$$A\Sigma_0^{\mathscr{L}} PDA = A\Sigma_1^{\mathscr{L}} PDA = P \quad [2],$$

$$A\Pi_2^{\mathscr{L}} PDA = \text{Co}-\text{NP} \quad [10],$$

$$A\Sigma_2^{\mathscr{L}} PDA = A\Sigma_k^{\mathscr{L}} PDA$$

$$= A\Pi_3^{\mathscr{L}} PDA = A\Pi_{k+1}^{\mathscr{L}} PDA = \text{PSPACE} \quad \text{for all } k \geqq 2 \quad [10].$$

Thus Theorem 3.6 shows that auxiliary pushdown automata with unbounded alternation but polynomial time constraint are equivalent to those with one (starting with an existential configuration) or more alternations but without any time bound. Hence there is a trade-off between the number of alternations and the time those machines are allowed to use. Note that without any restrictions on both the number of alternations and the time bound Ladner, Lipton, and Stockmeyer obtained $A\Sigma_\omega^{\mathscr{L}} PDA = \text{EXPTIME}$ [9, 10].

Further investigations show that the results obtained here for logarithmic space and polynomial time bounds (Theorem 3.4 and 3.6) generalize to arbitrary space-constructible bounds.

For given monotone $S(n) \geqq \log n$ and all $k \geqq 1$ let $A\Sigma_k TIME(c^{S(n)})$ ($\text{ATIME}(c^{S(n)})$) denote the class of all languages that can be accepted by alternating $c^{S(n)}$-time bounded Turing machines with $k-1$ alternations only (with unbounded alternation).

Then it holds for space-constructible $S(n) \geqq \log n$:

THEOREM 3.7:

(i) $\bigcup_{c \geqq 0} A\Sigma_{k+1}^{S(n)} PDA_{c^{S(n)}} = \bigcup_{c \geqq 0} A\Sigma_k TIME(c^{S(n)})$ *for all* $k \geqq 1$,

(ii) $A\Sigma_\omega^{S(n)} PDA_{c^{S(n)}} = \text{ATIME}(c^{S(n)})$.

Proof: (i) "$\subseteq$": Let L be accepted by an $A\Sigma_{k+1}^{S(n)} pda_{c^{S(n)}} M$. L can be reduced in nondeterministic time $c^{S(n)}$ to a set B_M that is defined as follows:

$$B_M := \{ w\$K\#^{c^{S(|w|)}-|K|-|w|} \mid K \text{ is a universal configuration of } M,$$

and M started in K accepts w using $S(|w|)$ space and $c^{S(|w|)}$ time only$\}$.

On input w the machine computing a reduction from L to B_M simulates the nondeterministic computation of M on w until M alternates to some universal configuration K. Then it outputs $w\$K\#^{c^{S(|w|)}-|K|-|w|}$. The simulation and output consumes at most $c^{S(|w|)}$ time since M has that time bound.

But obviously $B_M \in A\Pi_k^{\mathscr{L}} PDA_{pt}$, since on input $w\$K\#^{c^{S(|w|)}-|K|-|w|}$ the simulation of M by an $A\Pi_k^{\mathscr{L}} pda_{pt}$ consumes

$$S(|w|) = \log(|w\$K\#^{c^{S(|w|)}-|K|-|w|}|) \text{ space}$$

and

$$c^{S(|w|)} = |w\$K\#^{c^{S(|w|)}-|K|-|w|}| \text{ time.}$$

But with Lemma 3.1 we now obtain $B_M \in \Pi_{k-1}^{\mathcal{P}}$, and consequently

$$\begin{aligned} B_M \in \Pi_{k-1}^{\mathcal{P}} &\Rightarrow \exists j : B_M \in A\,\Pi_{k-1}\,TIME(n^j) \\ &\Rightarrow \exists j : L \in NTIME_{c^{S(n)}}(A\,\Pi_{k-1}\,TIME(n^j)) \\ &\Rightarrow \exists j : L \in A\,\Sigma_k\,TIME((c^{S(n)})^j) \\ &\Rightarrow L \in \bigcup_{d \geqq 0} A\,\Sigma_k\,TIME(d^{S(n)}). \end{aligned}$$

($NTIME_{c^{S(n)}}(\cdot)$ denotes the nondeterministic reduction mentioned above).

This completes the proof of the left-to-right inclusion of (i).

"$\supseteq$": Let M an $O(c^{S(n)})$-time bounded and k-alternation bounded Turing machine. W.l.o.g. we can assume that M alternates exactly $k-1$ times during each computation, and that M performs exactly $c^{S(n)}$ steps before each alternation. Now, by computing the following program an $A\,\Sigma_{k+1}^{S(n)}\,pda_{c^{S(n)}}\,M'$ can simulate M:

for $i := 1$ **to** $k+1$ **do**
begin
if i **odd then**
  begin
    • **existentially** guess a number x between 0 and $c^{S(n)}$;
    • pop $x-1$ configurations from the pushdown;
    • **accept** if M can not reach the top configuration on the pushdown in one step from the next to top configuration;
  or if $i \leqq k$ then
    • **existentially** guess $c^{S(n)}$ configurations of M onto the pushdown such that the last of these configurations is an accepting one if $i = k$ and universal if $i < k$;
    • **alternate**;
  end;
if i **even then**
  begin
    • **universally** guess all numbers x between 0 and $c^{S(n)}$;
    • pop $x-1$ configurations from the pushdown;
    • **accept** if M can reach the top configuration on the pushdown in one step from the next to top configuration;
  and if $i \leqq k$ then
    • **universally** guess $c^{S(n)}$ configurations of M onto the pushdown such that the last of these configurations is a rejecting one if $i = k$ and existential if $i < k$;

```
        • alternate;
    end;
end
```

M' essentially guesses and checks all the configuration sequences of length $c^{S(n)}$ which could have been performed by M between two alternations. These sequences contain either only existential configurations or universal ones.

The existential sequences are existentially guessed onto the pushdown store by M' and afterwards in universal mode checked to be a proper sequence while the other branch of the universal part of the program continues by guessing the following universal sequence of M which will be verified afterwards in existential mode either by guessing an error or by continuing the guessing of the following existential computation sequence. Since the machine M is $k \cdot c^{S(n)}$ time bounded all of its configurations are of length at most $k \cdot c^{S(n)}$. M' is $S(n)$ space bounded and thus it can control that the length of any configuration it pushes onto the pushdown does not exceed $k \cdot c^{S(n)}$. With the same space bound M' can count from 1 to $c^{s(n)}$ thus making sure that exactly $c^{S(n)}$ configurations are pushed onto the pushdown.

In order to verify that a sequence of $c^{S(n)}$ configurations—each of them of length at most $k \cdot c^{S(n)}$—is a legal computation of M the machine M' guesses in universal mode a number x between 0 and $c^{S(n)}$ to determine a configuration, and a number y between 0 and $k \cdot c^{S(n)}$ to determine a tape cell on M's worktape. Then M' first pops $x-1$ configurations from the pushdown and then it pops the contents of the first $y-1$ tape cells of configuration x, reads the symbol in cell y, and compares it with the content of cell y in configuration $x+1$. If these two symbols are identical M' accepts. If M's read-write head is positioned on cell y M' verifies that the difference between the two configurations corresponds to a legal move of M. Since M' is in universal mode when checking the sequence of configurations M' guesses all combinations of x and y and can accept only if the entire sequence is a legal computation of M.

For sequences of configurations that are guessed in universal mode M' may accept if the sequence does not correspond to a legal computation since the universal guessing mechanism guarantees that (on some other path of M's computation) there are also legal computations of M stored in the pushdown. Now M' guesses existentially whether the sequence on top of the push-down is a legal computation of M or not. For illegal computations M' guesses a configuration x on the pushdown and accepts if M can *not* reach this configuration in one step from configuration $x+1$. For legal sequences M' continues the simulation of M.

Having checked $k-1$ sequences the program constructs the k-th sequence such that the final configuration is accepting if k is odd and rejecting if k is even. For k odd the correctness of the sequence (i. e., the reachability of an accepting configuration) is verified as above. For k even the final sequence (ending in a rejecting state) is constructed universally. Now M' accepts if it can verify (by existentially guessing two configurations that do not correspond to a legal move of M) that all those sequences ending in a rejecting state are illegal computations of M. Consequently all legal sequences of M must have ended in an accepting configuration.

Since M' has checked all sequences to be legal computations of M it accepts if and only if these sequences correspond to an accepting computation of M. Hence M' accepts if and only if M accepts.

This completes the proof of (i).

(ii) $A\Sigma_{\omega}^{S(n)} PDA_{c^{S(n)}} = \mathrm{ATIME}(c^{S(n)})$ can be shown analogously. ▲

Note that ATIME $(c^{S(n)})$ equals DSPACE $(c^{S(n)})$ [3].

4. DISCUSSION

As shown by Borodin *et al.* [1] there are many ways to define hierarchies based on $LOG(CFL)$ which collapse to this class. We have shown a way to define a hierarchy based on $LOG(CFL)$ (or rather $LOG(DCFL)$) that collapses to $LOG(CFL)$ if and only if NP=P=$LOG(CFL)$, which is widely conjectured to be false. This hierarchy, moreover, essentially coincides with the Polynomial Hierarchy for which thus another characterization by simultaneously logarithmic space and polynomial time bounded alternating auxiliary pushdown automata has been obtained.

By taking a closer look at the proofs of Theorems 3.4 and 3.6 it can be seen that even a further characterization of the Polynomial Hierarchy and PSPACE can be obtained, namely by simultaneously logarithmic space and polynomial time bounded alternating auxiliary *checking stack automata* (for non-alternating versions *cf.* [4]) which alternate at most $k-1$ times during each computation ($A\Sigma_k^{\mathcal{L}} CSA_{pt}$). It holds $\Sigma_k^{\mathcal{P}} = A\Sigma_k^{\mathcal{L}} CSA_{pt}$, for all $k \geqq 1$. The left to right inclusion holds since for odd k $B_k \cap 3CNF$ (resp., $B_k \cap 3DNF$ for even k) can be recognized by a $k-1$ alternation bounded checking stack automaton which checks the satisfiability of all the clauses contained in the given $3CNF$ formula by reading the information stored on its stack again for each clause instead of using a further alternation. The other inclusion is obvious since the Polynomial Hierarchy can be characterized by alternating

polynomial time bounded Turing machines [3]. Note that alternating checking stack automata with the above space and time bounds and without a bound on the alternation depth again characterize PSPACE, i. e., $PSPACE = A\Sigma_{\omega}^{\mathcal{L}} CSA_{pt}$.

REFERENCES

1. A. Borodin, S. A. Cook. P. W. Dymond, W. L. Ruzzo, and M. Tompa, *Two Applications of Complementation via Inductive Counting*, manuscript, Sept. 1987.
2. S. A. Cook, *Characterizations of Pushdown Machines in Terms of Timebounded Computers*, Journ. of the ACM 18, Vol. 1, 1971, pp. 4-18.
3. A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer, *Alternation, Journ. of the ACM* 28, Vol. 1, 1981, pp. 114-133.
4. S. Greibach, *Checking Automata and One-way Stack Languages*, Journ. of Computer and System Sciences, Vol. 3, 1969, pp. 196-217.
5. J. E. Hopcroft, and J. D. Ullman, *Introduction to Automata Theory*, Languages, and Computation, Addison-Wesley, Reading, Mass., 1979.
6. N. Immerman, *Nondeterministic Space is Closed Under Complement*, Techn. Report, Yale University, YALEU/DCS/TR 552, July 1987.
7. B. Jenner, B. Kirsig, and K.-J. Lange, *The Logarithmic Alternation Hierarchy Collapses:* $A\Sigma_2^{\mathcal{L}} = A\Pi_2^{\mathcal{L}}$ *(extended version)*, to be published in *Information and Computation.*
8. K.-J. Lange, B. Jenner, and B. Kirsig, *The Logarithmic Alternation Hierarchy Collapses:* $A\Sigma_2^{\mathcal{L}} = A\Pi_2^{\mathcal{L}}$, Proc. of the 14th ICALP, Karlsruhe, July 1987, Lect. Notes in Comp. Sci., Vol. 267, pp. 531-541.
9. R. E. Ladner, R. J. Lipton, and L.J. Stockmeyer, *Alternating Pushdown Automata*, Proc. of the 19th IEEE Symp. on Foundations of Comp. Sci., Ann Arbor, Mich., 1978, pp. 92-106.
10. R. E. Ladner, L. J. Stockmeyer, and R. J. Lipton, *Alternation Bounded Auxiliary Push-down Automata*, Information and Control, Vol. 62, 1984, pp. 93-108.
11. U. Schöning, and K. W. Wagner, *Collapsing Oracle Hierarchies, Census Functions and Logarithmically Many Queries*, Report No. 140, Universität Augsburg, May 1987.
12. L. J. Stockmeyer, *The Polynomial-time Hierarchy*, Theoret. Comp. Sci., Vol. 3, 1976, pp. 1-22.
13. I. H. Sudborough, *On the Tape Complexity of Deterministic Context-Free Languages*, Journ. of the ACM 25, Vol. 3, 1978, pp. 405-414.
14. R. Szelepcsényi, *The Method of Forcing for Nondeterministic Automata*, Bull. European Assoc. Theoret. Comp. Sci. No. 33, Oct. 1987, pp. 96-100.