Unambiguity and Fewness for Logarithmic Space[*]

(Preliminary Version, January 1991)

Gerhard Buntrock[†] Birgit Jenner[‡] Klaus-Jörn Lange[‡]
Peter Rossmanith[‡]

Abstract

We consider various types of unambiguity for logarithmic space bounded Turing machines and polynomial time bounded log space auxiliary push-down automata. In particular, we introduce the notions of (general), reach, and strong unambiguity. We demonstrate that closure under complement of unambiguous classes implies equivalence of unambiguity and "unambiguous fewness". This, as we will show, applies in the cases of reach and strong unambiguity for logspace. Among the many relations we exhibit, we show that the unambiguous linear contextfree languages, which are not known to be contained in LOGSPACE, nevertheless are contained in strongly unambiguous logspace, and, consequently, in LOGDCFL. In fact, this turns out to be the case for all logspace languages with reach unambiguous fewness. In addition, we show that general unambiguity and fewness of logspace classes can be simulated by reach unambiguity and fewness of auxiliary push-down automata.

1 Introduction

Although the power of nondeterminism for space classes has been demonstrated by the development of skillful techniques like inductive counting, [Imm88], [Sze88], the long open LBA-problem "DSPACE(n) = NSPACE(n)?" accompanied by its logarithmic version "L = NL?" remains unsolved.

For the area of logarithmic space, various interesting approaches have been made to gain insight into properties of complexity classes defined by taking the number of acceptable computations of nondeterministic Turing machines into account. In [AJ90],

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

[†]Institut für Informatik, Universität Würzburg, D-8700 Würzburg, Germany. This research was started while this author was a guest of SFB-342, A4, at the Technische Universität München, Institut für Informatik.

[‡]Technische Universität München, Institut für Informatik, Arcisstr. 21, D-8000 München 2, Germany

[BDHM91] e.g., counting classes, like e.g. #L, MOD$_k$L, have been considered and shown to be related to parallel complexity classes like the NC-hierarchy. On the other hand, by imposing bounds on the number of paths leading to configurations of logspace bounded nondeterministic Turing machines classes of bounded nondeterminism have been defined in [BHS90], where some insights into the possibilities of removing nondeterminism were gained.

Both of these approaches were conducted by similar approaches in the area of polynomial time and the P = NP? question. There, properties of counting classes like $\#P$, $\oplus P$, MOD$_k$P, are ongoing research topics, and various of classes with bounded nondeterminism have been considered. The first such class was UP, defined by Valiant as the class of languages that are unambiguously acceptable by NP machines, that is, with at most one accepting computation [Val76]. This class is of particular importance in relation with one-way functions and cryptography [GS88]. By allowing an NP machine up to polynomially many accepting computations, the class FewP, introduced by Allender [All86], is obtained. This class has been shown to be contained in MODZ$_k$P [CH89].

In this paper we want to investigate properties of various language classes that are characterizable by nondeterministic logspace space Turing machines and simultaneously logspace and polynomial time bounded auxiliary push-down automata (AuxPDA) that obey similar ambiguity restrictions as UP and FewP machines. While many known methods approved for the treatment of general nondeterminism remain suitable for our purpose, some techniques like inductive counting, which respects the existence but not the number of accepting computations, in general turn out to be inadequate.

First we will consider the logspace bounded analog of the polynomial time classes UP and FewP. This leads to the two classes UL and FewL, and, to the auxiliary push-down automata classes UAuxPDA and FewAuxPDA. But we will consider further restrictions of these classes, which arise from ambiguity requirements that are w.l.o.g. fulfilled for the corresponding time classes UP and FewP, and seem to be in particular appropriate in the case of logspace classes. Since nondeterministic polynomial time Turing machines can protocol their computation, we can w.l.o.g. assume that for such machines there is at most one path between any two arbitrary configurations. (This is true for any purely time bounded Turing machine.)Due to lack of freely accessible space, logspace machines and AuxPDA cannot store their computation. Therefore, here the classes FewUL and FewUAuxPDA are of interest. These consist of FewL and FewAuxPDA languages, respectively, acceptable by machines which satisfy for all inputs that there exists at most one path from the initial to an accepting configuration. The class FewUL has already be demonstrated to be the more appropriate logspace equivalent to FewP, since the inclusion FewP $\subseteq$ MODZ$_k$P for all $k \geq 2$ translates to FewUL but does not seem to translate to FewL, as shown in [BDHM91].

We will state various closure properties of the classes UL, FewUL, and UAuxPDA, FewUAuxPDA. In particular, we will show that the two *few* classes are exactly the closure of the corresponding *unambiguous* classes under logspace disjunctive Turing reductions or equivalently, under logspace bounded existential quantification.

We furthermore consider stronger notions of ambiguity, *reach unambiguity* and *strong* *unambiguity*, for both logspace machines and AuxPDA. While the usual unambiguity notions pertain to accepting computations only, that is, computations starting in the initial configuration and ending in accepting configurations, reach unambiguity puts the corresponding restrictions to each computation path starting in the initial configuration and ending in any (reachable) configuration. In addition, strong unambiguity requires the corresponding conditions to hold even for arbitrary computations between any pair of configurations. These concepts may be compared with those used within formal language theory. There, usually, unrestricted unambiguity is studied, but since it is possible to remove both the unreachable and the unproductive nonterminals of a grammar, strong unambiguity and unrestricted unambiguity of formal languages are coincident. This is reflected by the recognizability of unambiguous contextfree languages by strongly unambiguous push-down automata, which could recently be shown (see [LR90]). It is stressed by the inclusion of all unambiguous linear contextfree languages in the class of strongly unambiguous logspace languages, which we show below. We furthermore will show that for both—reach unambiguity and strong unambiguity—unambiguous fewness and unambiguity coincide.

The reader is assumed to be familiar with basic facts and definitions of structural complexity theory as e.g. stated in [HU79] or [BDG88], [BDG90]. In general, more specific concepts and notations are introduced just before used.

2 Logarithmic Space

In this section we will introduce and investigate various ambiguity and fewness notions for nondeterministic logarithmic space NL.

2.1 Unambiguous Computations and Few Computations

A direct logspace analogon of Valiant's class UP [Val76] is given with the following definition.

Definition 1 *We define the class* UL *as the class of all sets accepted by nondeterministic logarithmic space bounded Turing machines for which there exists* at most one accepting computation *for all inputs x.*

Generalizations of UL[1] can be obtained by bounding the potentially exponential number of accepting computations by (non-constant) functions (depending on the length of the input). Of primary interest in the case of polynomial time classes are here polynomial bounds on the number of accepting paths, leading to the class FewP introduced and investigated in [All86], [CH89]. Again, we get a direct logspace analogon of this class FewP, by simply exchanging polynomial time bounded nondeterministic Turing machines by logarithmic space bounded machines.

[1]We understand the letter "U" as an abbreviation for the fact that the underlying machines accept *unambiguously*.

Definition 2 *The class* FewL *is defined as the class of sets accepted by nondeterministic logarithmic space bounded Turing machines M for which there is a polynomial p_M such that for all inputs x there are* fewer than $p_M(|x|)$ accepting computations *of M on x.*

In the case of polynomial time classes, clearly, any of the computations on an arbitrary input x can be made "unambiguous" in the sense that no two computations on x lead to the same accepting configuration. This can be achieved simply by storing the whole computation. In the case of logarithmic space, no complete computation can be recorded. Therefore here, too, the following (weaker) generalization of the class UL is of interest, which nevertheless still has the spirit of "fewness".

Definition 3 *Define the class* FewUL *as the class of all sets accepted by nondeterministic logarithmic space bounded Turing machines which satisfy for all inputs x that there is* at most one computation from the start configuration to any accepting configuration.

Note that a polynomial bound on the number of paths is induced by the polynomial bound on the number of different configurations.

Clearly, it holds L $\subseteq$ UL $\subseteq$ FewUL $\subseteq$ FewL. Furthermore, it is easily verified that UL, FewUL, and FewL satisfy the following closure properties.

A language class $\mathcal{A}$ is closed under join, if for all $A, B \in \mathcal{A}$, the language $1 \cdot A \cup 0 \cdot B$ is contained in $\mathcal{A}$. A language class $\mathcal{A}$ is closed under disjunctive (respectively, conjunctive) logspace Turing reductions, iff for all $B \in \mathcal{A}$, the language accepted by a deterministic logspace oracle machine with oracle B that immediately accepts (respectively, rejects) when a query is answered positively (respectively, negatively) is contained in $\mathcal{A}$ (this notion goes back to [LL76]; see [JKL89] for various properties of such reductions).

Proposition 4 *(1)* UL, FewUL, FewL *are closed under join.*

(2) UL, FewUL, *and* FewL *are closed under logspace many-one reductions.*

(3) UL *and* FewL *are closed under conjunctive logspace Turing reductions* $\mathrm{L}_c(\cdot)$, *and thus under intersection and marked concatenation.*

(4) FewUL *and* FewL *are closed under disjunctive logspace Turing reductions* $\mathrm{L}_d(\cdot)$, *and thus under union.*$\Box$

The class FewUL has been proposed in [BDHM91] (named there LogFewL) as a more adequate logspace equivalent of FewP than FewL, since it satisfies some of the properties known to hold correspondingly for polynomial time classes. The following statement of [BDHM91] parallels the result FewP $\subseteq$ MODZ$_k$P shown in [CH89]. We give a simple proof.

Theorem 5 *[BDHM91]* FewUL $\subseteq$ MODZ$_k$L *for any $k \geq 2$.*

Proof. Let M be a FewUL machine that accepts a language A with $f(x)$ accepting computation paths on input x such that $x \in A \iff f(x) > 0$. Then a logspace machine M' with exactly $k^{f(x)} - 1$ accepting computation paths on input x can be constructed for any $k \geq 2$. Since $k^{f(x)} - 1$ is not divisible by k if $f(x) > 0$ and equals 0 if $f(x) = 0$, M' is a MODZ$_k$L machine for A.

M' works as follows: On input x, M' cycles through all accepting configurations of M in lexicographical order. Each of these configurations is nondeterministically chosen to be selected up to $k-1$ times or not to be selected. Any time a selection of a configuration c occurs, M' verfies by simulation of M that c is reachable from the start configuration c_0. If this is not the case M' rejects. Clearly, then, any accepting path of M' corresponds to a (non-empty) subset of the reachable accepting configurations of M with up to $k-1$ occurences of each configuration. Since there are exactly $k^{f(x)} - 1$ (non-empty) of such subsets, M' will have so many accepting paths if $f(x) > 1$ and none if $f(x) = 0$. $\square$

A similar result for FewL appears difficult to achieve, since there seems to be no way to differentiate between more than a constant number of computation paths. In fact, no nontrivial relationship of FewL to any other complexity class is known.

In the proof of Theorem 5, for the class FewUL it could be made use of the fact that each computation path of a FewUL machine has a short description by the configuration in which the paths ends. Thus, by cycling through all configurations in some ordering a logspace machine has the capacity to differentiate between a polynomial number of paths.

The availability of a short description of paths furthermore enables us to show that FewUL is just the closure of UL under disjunctive logspace Turing reductions $\mathrm{L}_d(\cdot)$, or alternatively expressed, that any language $L_1 \in$ FewUL can be obtained from a language $L_2 \in$ UL by logspace Turing reductions and log space bounded existential quantification is very close. For any class of languages $\mathcal{A}$ closed under log space many-one reductions, it holds: $\mathrm{L}_d(\mathcal{A}) = \exists_{\log}\mathcal{A}$. A dual relationship holds between conjunctive log space Turing reductions and log space bounded universal quantification.

Theorem 6 FewUL $= \mathrm{L}_d(\mathrm{UL}) = \exists_{\log}\mathrm{UL}$.

Proof. For the inclusion FewUL $\subseteq \mathrm{L}_d(\mathrm{UL})$ construct for a given FewUL machine M a deterministic log space Turing machine M' that queries the following oracle A for each accepting configuration in lexicographical order and accepts if and only if one of the queries is answered positively:

$$A := \{(x, c) \mid \begin{array}{l} c \text{ is an accepting configuration of } M, \text{ and} \\ \text{there exists a computation path of } M \text{ on input } x \text{ ending in } c\} \end{array}$$

Thus, M' works disjunctively.

Since M is a FewUL machine, it is easy to see that A is contained in UL. Furthermore, if M accepts an input x, then there exists a smallest accepting configuration which leads to acceptance. This configuration will eventually lead to a positively answered query of M'. Conversely, if M' accepts, then only because one of its queries (x, c_i) is answered positively, i.e.,'when the configuration c_i is reachable by M on input x. (In fact, it is the first reachable configuration in lexicographic order that is reachable by M on input x.) Thus, M and M' accept the same language.

The inclusion $\mathrm{L}_d(\mathrm{UL}) \subseteq \mathrm{FewUL}$ follows from the fact that FewUL is closed under disjunctive log space Turing reductions (Proposition 4*(4)*).

Since UL is closed under log space many-one reductions Proposition (4*(2)*), it holds $\mathrm{L}_d(\mathrm{UL}) = \exists_{\log}\mathrm{UL}$, too. □

It is questionable whether UL or FewUL are closed under complementation. Note that the inductive counting technique developed by Immerman and Szelepcsényi for showing that NL is closed under complementation [Imm88], [Sze88] does neither maintain the unambiguous nor the few accepting requirement, and can thus not be applied to UL or FewUL. For the procedure of counting the configurations on a certain level of the computation tree, reachability questions have to be solved, which we can only assume to be solved unambiguously, or, respectively, with a few number of paths, for configurations lying on accepting paths.

But since FewUL is exactly the closure of UL under disjunctive log space Turing reductions, and UL is closed under conjunctive log space Turing reductions, the closure of UL under complementation already implies the equality of UL and FewUL. This follows from the fact that for any language class $\mathcal{A}$ it holds: $\mathrm{L}_d(\mathcal{A}) = \mathrm{CoL}_c(\mathrm{Co}\mathcal{A})$ ([JK89]).

Corollary 7 *If* $\mathrm{UL} = \mathrm{CoUL}$*, then* $\mathrm{UL} = \mathrm{FewUL}$.

Proof. The assumption together with Theorem 6 and Proposition 4 imply $\mathrm{FewUL} = \mathrm{L}_d(\mathrm{UL}) = \mathrm{CoL}_c(\mathrm{CoUL}) = \mathrm{CoL}_c(\mathrm{UL}) = \mathrm{CoUL} = \mathrm{UL}$. □

Another consequence of Theorem 6 is that we cannot hope for the closure of UL under (unmarked) concatenation unless we believe UL = FewUL, since the concatenation of UL with UL is FewUL-complete:

Corollary 8 $\mathrm{FewUL} = \mathrm{LOG}(\mathrm{UL} \cdot \mathrm{UL})$.

Proof. Inga Niepel showed $\mathrm{L}_d(\mathcal{A}) = \mathrm{LOG}(\mathrm{LOG}(\mathcal{A}) \cdot \mathrm{LOG}(\mathcal{A}))$, if $\mathrm{LOG}(\mathcal{A})$ is closed under intersection (see [Nie91]). Setting $\mathcal{A} = \mathrm{UL}$ the result follows from Proposition 4. □

The preceding results show that conjunctive and disjunctive log space Turing reductions applied in taking turns, build up a hierarchy. The first Σ-level of this hierarchy is just FewUL. This hierarchy collapses to UL iff UL is closed under $\mathrm{L}_d(\cdot)$, or to FewUL iff FewUL is closed under $\mathrm{L}_c(\cdot)$. But it is not clear whether UL is even closed under union or FewUL under intersection.

Due to the lack of knowledge whether UL or FewUL are closed under complementation various questions concerning the relativizations $\mathrm{L}_1(\mathrm{UL})$, $\mathrm{L}(\mathrm{UL})$, $\mathrm{UL}(\mathrm{UL})$, $\mathrm{FewUL}(\mathrm{UL})$, FewL(UL), FewUL(FewUL), FewL(UL) etc. of these classes arise. Are there any non-trivial upper bounds for any of these classes? Here, $L_1(\cdot)$ denotes the closure of UL under Turing reductions computed by log space oracle Turing machines that query at most once. Note that it is even unclear whether this "smallest" of these classes is contained in the "biggest" class FewL.

Clearly, considering conjunctive and disjunctive restrictions of these relativizations we can say something more. Here Theorem 6 can be generalized. We define these reductions by requiring the machines realizing the conjunctive (disjunctive) reduction to immidiately reject (accept) on a computation path as soon as a query on this path has been answered negatively (positively). Furthermore, we require the machines to write on the oracle tape deterministically (this is the Ruzzo-Simon-Tompa restriction of the original Ladner-Lynch log space oracle machine model; see [LL76], [RST84]). The number of admissible accepting paths is guided by the type of reduction, as in the case of the machines without oracle.

Proposition 9 *(1)* $\mathrm{UL} = \mathrm{UL}_c(\mathrm{UL})$.

(2) $\mathrm{FewUL} = \mathrm{FewUL}_d(\mathrm{FewUL}) = \mathrm{UL}_d(\mathrm{UL})$.

(3) $\mathrm{FewL} = \mathrm{FewL}_d(\mathrm{FewUL})$

Proof. *(1)* It suffices to show that $\mathrm{UL}_c(\mathrm{UL}) \subseteq \mathrm{UL}$. For this, construct a UL machine M' that simulates the $\mathrm{UL}_c(\mathrm{UL})$ machine M step by step. Before each query made by M, M' guesses the result 0 or 1, and stops the simulation with a reject if it has guessed 0 and verifies the query if it has guessed 1. Clearly, M' will accept iff M accepts. Furthermore it is easily verified that M' is a UL machine.

The proof of *(2)* and *(3)* is analog to *(1)*. Here, dually, after guessing 0 or 1 M' goes on with the simulation, if it has guesses a 0 and verifies the query if it has guessed 1. □

These results can be compared with corresponding results concerning unrestricted nondeterministic relativizations. Also the crucial distinction between Ruzzo-Simon-Tompa relativization and that of Ladner-Lynch appears in this context. Here we have, indicating with the index "ll" the Ladner-Lynch relativization, where the oracle machine is allowed to write nondeterministically onto a polynomially bounded oracle tape:

Proposition 10 $\mathrm{FewL}_d(\mathrm{L})_{ll} = \mathrm{FewP}$.

Proof. The inclusion $\mathrm{FewL}_d(\mathrm{L})_{ll} \subseteq \mathrm{FewP}$ holds since a FewP machine can completely simulate the $\mathrm{FewL}_d(\mathrm{L})$ machine. Conversely, an FewP machine can be simulated by a $\mathrm{FewL}_d(\mathrm{L})_{ll}$ that guesses a computation path onto its oracle tape and verifies the correctness of the path with the help of its oracle. □

It should be remarked here that even one question suffices.

Note that some of the properties stated here for logspace bounded classes hold for polynomially time bounded classes as well.

Proposition 11 *(1)* $\mathrm{P}_c(\mathrm{UP}) = \mathrm{UP}_c(\mathrm{UP}) = \mathrm{UP}$*;*

(2) $\mathrm{P}_d(\mathrm{FewP}) = \mathrm{UP}_d(\mathrm{FewP}) = \mathrm{FewP}_d(\mathrm{FewP}) = \mathrm{FewP}$*;*

(3) $\mathrm{P}_d(\mathrm{UP}) \subseteq \mathrm{FewP}$. $\square$

One of the properties not likely to hold for polynomial time is the reverse inclusion of Proposition 11 *(3)*.

2.2 Unambiguous Reachability

The fact that the whole computation of polynomial time bounded Turing machines can always be recorded not only ensures that for FewP machines w.l.o.g. it can be excluded that there are two computations leading to the same accepting configuration. It furthermore ensures that in general it can be excluded that there are more than two paths between *any* two configurations, whether reachable or not. For logarithmic space classes such differences give rise to a variety of subclasses of UL and FewL. With the requirement that each reachable configuration is unambiguously reachable we get the two classes ReachUL and ReachFewUL. [2]

Definition 12 *The classes* ReachUL *and* ReachFewUL *are defined as the subclass of languages in* UL*, and, resp.,* FewUL*, for which there exists a nondeterministic log space Turing machine, which satisfies for any input x and* for any configuration c that there is at most one path from the start configuration to c.

It holds $\mathrm{L} \subseteq \mathrm{ReachUL} \subseteq \mathrm{ReachFewUL}$. Note that expressed in the terminology used in [BHS90] the class ReachFewUL is the class NSPACE-AMBIGUITY$[\log n, 1]$.

The further requirement[3] that also the non-reachable configurations reach any other configuration unambiguously leads to the following subclasses of ReachUL and ReachFewUL.

Definition 13 *The classes* StrongUL *and* StrongFewUL *are the subclasses of* UL *and* FewUL *for which there exist nondeterministic log space Turing machines which satisfy the even stronger requirement that there is* at most one path between any two configurations.

Again, it holds $\mathrm{L} \subseteq \mathrm{StrongUL} \subseteq \mathrm{StrongFewUL}$.

Note that both definitions imply the very tight restriction that the computation tree of an arbitrary ReachUL or StrongUL machine M is polynomially sparse, that is of polynomial size. The computation tree can only contain a polynomial number of nodes,

[2]We understand the prefix "Reach" of "UL" and "FewUL" as an abbreviation for the *further* restriction of these classes that any reachable configuration for the underlying machines is *unambiguously* reachable.

[3]We have chosen the prefix "Strong" for this strong requirement to indicate that the unambiguous acceptance requirement here extends to an unambiguous reachability for any two configurations between which there exists a path, whether they are reachable or not.

each node representing a configuration c of M and the path leading to c from the start configuration.

The class ReachUL can be shown to be closed under complementation by using the counting techniques developed by Immerman/Szelepcsényi. Here now by definition the unambiguous reachability of any reachable configuration is guaranteed by definition. Unfortunately, this does not work for StrongUL, since usage of inductive counting produces multiple paths between not reachable configurations. Nevertheless, StrongUL is closed under complementation, too, which can be shown by a different proof technique.

Theorem 14 StrongUL *and* ReachUL *are closed under complementation.*

Proof. For the proof of StrongUL = CoStrongUL, first note that, as shown in Proposition 9 *(1)* for the class UL, an UL oracle accessed conjunctively does not increase the power of a UL machine. The same holds true for a StrongUL machine with a StrongUL oracle. Since the whole computation tree of a StrongUL machine has only polynomial size, we will do a depth-first traversal through it. The problem of erroneously leaving this tree when climbing up, and getting lost in the area of not reachable configurations, can be solved by always checking the reachability of the current node. Clearly, this can be done by questions conjunctively to a StrongUL oracle.

For the class ReachUL the proof is analog. Alternatively, the inductive counting technique described for NL in [Imm88] can be applied. □

For classes ReachUL, ReachFewUL on the one hand, and the classes StrongUL, and StrongFewUL on the other, Proposition 4 and Theorem 6, and Corollary 7 remain valid. Thus, with Theorem 6 we see that ReachUL and ReachFewUL on the one hand, and StrongUL and StrongFewUL on the other, in fact are equal. Furthermore, with Theorem 14 it is then easily shown that ReachUL and StrongUL are closed under (arbitrary) log space Turing reductions.

Corollary 15 *(1)* ReachUL = ReachFewUL*.*

(2) StrongUL = StrongFewUL*.*

(3) ReachUL *and* StrongUL *are closed under log space Turing reductions.* □

The class ReachUL is interesting in a different respect, too. It is a candidate for a class not known to coincide with L but contained in LOGDCFL, the closure of the deterministic context-free languages under log space many-one reductions, see [Sud78]. The latter class is known to be characterizable by CROW-PRAMs ([DR86]). It holds ReachUL $\subseteq$ LOGDCFL. This result will be generalized in the following subsection to those UL languages that have polynomial ambiguity.

The strongly unambiguous classes considered so far—in particular StrongUL—may seem to be very restricted and one might argue that these classes coincide with L. But the following result shows that the unambiguous linear languages, which are not known to lie in L, are contained in StrongUL. In fact, we will see that the usual nondeterministic logspace algorithm to recognize linear languages behaves strongly unambiguous when simulating elements of ULIN.

Theorem 16 ULIN $\subseteq$ StrongUL.

Proof. Let $G = (N, T, P, S)$ an unambiguous linear context-free grammar. Without loss of generality we assume every nonterminal $A \in N$ to be both reachable and productive. Let $w \in T^*$ be an arbitrary, but fixed input of length $n := |w|$, i.e., $w = a_1 a_2 \ldots a_n$ for some $a_i \in T$. For $0 \leq i \leq j \leq n$ we set ${}_iw_j := a_{i+1} \ldots a_j$. Thus we have ${}_iw_i = \lambda$ and ${}_0w_n = w$.

We now describe the behavior of a nondeterministic logspace algorithm M which will accept $L(G)$. The configurations of M are triples $\langle i, j, A\rangle$ with $0 \leq i \leq j \leq n$ and $A \in N$. The initial configuration is $\langle 0, n, S\rangle$. All triples $\langle i, j, A\rangle$ with $A \to {}_iw_j \in P$ are accepting configurations. M now works, as follows, when in a nonaccepting configuration $\langle i, j, A\rangle$ M guesses i', j' with $i \leq i' \leq j' \leq j$, and B with $B \in N$ such that $A \to {}_iw_{i'}\, B_{j'}w_j \in P$. If no i', j', and B exists with this property, there is no following configuration for $\langle i, j, A\rangle$ in M, thus M would stop rejecting. If M finds adequate i', j', and B, it enters the configuration $\langle i', j', B\rangle$.

Obviously, M is a nondeterministic logspace algorithm. Further on, the following equivalence holds for arbitrary $A, B \in N$, and $i \leq i' \leq j' \leq j$:

$$(*) \qquad \begin{array}{c} \langle i, j, A\rangle \underset{M}{\overset{*}{\vdash}} \langle i', j', B\rangle \\ \text{if and only if} \\ A \xrightarrow[G]{*} {}_iw_{i'} B_{j'}w_j. \end{array}$$

Hence, the unambiguity of G assures the unambiguity of M. Assume now M is not to be strongly unambiguous. Then there exist computations $\langle i, j, A\rangle \underset{M}{\overset{1}{\vdash}} \langle i_1, j_1, A_1\rangle \underset{M}{\overset{*}{\vdash}} \langle i', j', B\rangle$ and $\langle i, j, A\rangle \underset{M}{\overset{1}{\vdash}} \langle i_2, j_2, A_2\rangle \underset{M}{\overset{*}{\vdash}} \langle i', j', B\rangle$ such that $\langle i_1, j_1, A_1\rangle \neq \langle i_2, j_2, A_2\rangle$. By $(*)$ this gives rise to two different derivations from A to ${}_iw_{i'}B_{j'}w_j$ in G. Since A is reachable and B productive there exist $v_1, v_2, v_3 \in T^*$ with $S \xrightarrow[G]{*} v_1 A v_3$ and $B \xrightarrow[G]{*} v_2$. But then the word $v_{1\,i}w_{i'}v_{2\,j'}w_j v_3$ possesses more than one derivation in G, contradicting our assumption of the unambiguity of G. □

Theorem 16 leaves open the question for the StrongUL-completeness of the family ULIN.

By Theorem 16 this gives us the inclusion ULIN $\subseteq$ DAuxPDA and thus by Sudboroughs result (see [Sud78]) LOGULIN $\subseteq$ LOGDCFL, which is remarkable since there are unambiguous linear context free languages, like the mirror language, which are not deterministic context-free. LOGDCFL coincides with the class of languages accepted by CROW-PRAMs in logarithmic time (see [DR86]), which exhibits OROW-PRAM languages (see [Ros91]) as another class between L and LOGDCFL. Its relations to StrongUL, ReachUL, and ReachFewL remain open.

2.3 Few Reachability

We considered restrictions ReachUL and ReachFewUL of UL by requiring *all* reachable configurations to be unambiguously reachable instead of only accepting ones. The same restriction applied to FewL leads to the class ReachFewL, which is denoted by NSPACE-AMBIGUITY$(\log n, n^{O(1)})$ in the terminology of [BHS90].[4]

Definition 17 *The class* ReachFewL *is defined as the subclass of languages in* FewL *for which there exists a nondeterministic log space Turing machine, which satisfies for any input x and for any configuration c that there are at most polynomially many paths from the start configuration to c.*

Proposition 18 ReachFewL $\subseteq$ LOGDCFL.

Proof. A deterministic auxiliary push-down automaton can test reachability in a computation tree of an NL machine by a simple depth-first search. In general, this will take more than polynomial time, since this tree can have exponential size, but if each configuration is reachable by only polynomially many paths, the size of this tree will also stay polynomial. $\Box$

This result improves the upper bound of ReachFewL in terms of unambiguous auxiliary push-down automata UAuxPDA, see [BHS90].

3 Push-down Automata with Auxiliary Logarithmic Space

In this section we will deal with auxiliary push-down automata introduced by Cook in [Coo71]. An auxiliary push-down automaton is a space bounded Turing machine with an additional push-down store that does not count for the space bound. Historicly, we consider only auxiliary push-down automata that are simultaneously logarithmically space bounded and polynomially time bounded, which accept with empty push-down store (see [Sud78]). The power of these machines lies between NL and NP, since auxiliary push-down automata can use more space than NL machines. Actually they can use as many space as any NP machine, but since the bulk of space can be accessed only in a very limited way, NP machines might be a lot more powerful. It is well-known that nondeterministic auxiliary push-down automata accept exactly LOGCFL, i.e., the closure of context free languages under log space many-one reductions [Sud78].

Many of the above defined log space classes coincide for polynomial time since a P machine can store its whole computation. Again, this is *not* possible for auxiliary

[4]With prefixing "Reach" to "FewL" we want to indicate that the polynomial bound on the number of acceptance paths extends to a polynomial bound on the number of paths between two arbitrary configurations.

push-down automata, so it makes sense to consider AuxPDA classes that are defined analogously to the log space classes of Section 2.

However, only few results for log space classes hold literally for AuxPDAs, too, but in most cases a similar, but somewhat weaker, result will be obtained.

3.1 Unambiguous Computations and Few Computations

We begin with a definition of unambiguous auxiliary push-down automata, analogous to UP and UL.

Definition 19 UAuxPDA *is the class of sets accepted by auxiliary push-down automata for which there exists at most one accepting computation for all inputs x.*

Next we define the class FewAuxPDA as a direct analogon of FewP and FewL.

Definition 20 *The class* FewAuxPDA *is defined as the class of sets accepted by nondeterministic logarithmic space and polynomially time bounded auxiliary push-down automata M for which there is a polynomial p_M such that for all inputs x there are* fewer than $p_M(|x|)$ accepting computations *of M on x.*

Just as in the case of FewL, the class FewAuxPDA does not reflect all properties of FewP in a way we would expect. Again, this is a reason to consider FewUAuxPDA as an alternative to FewAuxPDA to define a class that reflects "fewness" in a nice way.

Definition 21 *Define the class* FewUAuxPDA *as the class of all sets accepted by nondeterministic logarithmic space and polynomially time bounded auxiliary push-down automata which satisfy for all inputs x that there is* at most one computation from the start configuration to any accepting configuration.

A polynomial number of accepting paths is automatically guaranteed by Definition 21. This follows from the at most polynomial number of accepting configurations due to acceptance with empty push-down store.

We can show the same closure properties for UAuxPDA and FewAuxPDA as were shown in Proposition 4 for UL and FewUL.

Proposition 22 *(1)* UAuxPDA *is closed under conjunctive log space Turing reductions* $\mathrm{L}_c(\cdot)$ *— and thus under intersection and marked concatenation.*

(2) FewAuxPDA *is closed under disjunctive log space Turing reductions* $\mathrm{L}_d(\cdot)$ *— and thus under union.*

(3) Both UAuxPDA *and* FewUAuxPDA *are closed under join.*

The possibility to identify accepting computations by accepting configurations for UAuxPDA and FewUAuxPDA allows us to prove Theorem 6 and thus Corollary 7 for AuxPDA classes, too.

Proposition 23 FewUAuxPDA = L_d(UAuxPDA) = $\exists_{\log}$UAuxPDA.

Proof. Analog to Theorem 6. □

Corollary 24 *If* UAuxPDA = CoUAuxPDA, *then* UAuxPDA = FewUAuxPDA.

Proof. Analog to Corollary 7. □

Just as it is not possible to apply the Immerman/Szelepcsényi inductive counting technique to UL, it is not possible to transfer the proof technique of [BCD$^+$88] from nondeterministic to unambiguous auxiliary push-down automata. (Of course, actually this proof has first to be translated from circuits to auxiliary push-down automata.)

3.2 Unambiguous Reachability

In subsection 2.1 we introduced log space classes with the additional restriction that each language is accepted by some log space bounded Turing machine which reaches each of its configurations with only one computation path. In this subsection we will examine AuxPDAs that are restricted in the same way. Here we use the notion of Cook [Coo71] and denote by an configuration of an auxiliary push-down automaton the instantaneous description of the state, the input tape position, the work tape(s) content, the work tape(s) head(s) position(s), and the topmost symbol of the push-down store. This information can be stored with $O(\log n)$ space.

Definition 25 *The classes* ReachUAuxPDA *and* ReachFewUAuxPDA *are defined as the subclass of languages in* UAuxPDA *and* FewUAuxPDA *resp., for which there exists a nondeterministic auxiliary push-down automaton, which satisfies for any input x and* for any configuration c that there is at most one path from the start configuration to c.

Neither log space bounded Turing machines nor auxiliary push-down automata are able to store their whole computation. However, an auxiliary push-down automaton can simulate a log space machine and simultaneously use its push-down to store the computation of the log space machine. Of course, this works only for rejecting computations, since an auxiliary push-down automaton is required to accept with empty push-down store. This means that an additional push-down enables a log space machine to reach unambiguously each of its configurations which are not part of some accepting computation.

Theorem 26 *(1)* UL ⊆ ReachUAuxPDA,

(2) FewUL ⊆ ReachFewUAuxPDA.

Proof. Simulate a UL (resp. FewUL) machine with some AuxPDA and store its computation on the push-down store. When an accepting configuration is reached, empty the push-down store and accept. □

Definition 27 *The classes* StrongUAuxPDA *and* StrongFewUAuxPDA *are defined as the subclass of languages in* UAuxPDA*, and, resp.,* FewUAuxPDA*, for which there exists a nondeterministic auxiliary push-down automaton, which satisfies for any input x and* for any two configuration c_1 and c_2 that there is at most one path between c_1 and c_2.

The strongly unambiguous logspace class StrongUL which equals StrongFewUL by Corollary 15 play only a minor role in this paper since most results of Section 2 that could be shown for these classes, could be shown also for the more general classes ReachUL = ReachFewUL. For strongly unambiguous AuxPDA classes this is not the case. For example, the technique of inductive counting can only be applied to strongly unambiguous push-down automata. On the other hand, inductive counting does not maintain strong unambiguity. In contrast to ReachUL we cannot prove closure under complementation for these reasons, but a weaker result:

Proposition 28 [NR90] CoStrongUAuxPDA $\subseteq$ ReachUAuxPDA*.*

An implication of closure under complement of ReachUL and StrongUL was that ReachUL and ReachFewUL, resp. StrongUL and StrongFewUL coincide. Since we have complementation results for neither ReachUAuxPDA nor StrongUAuxPDA, we can again only prove a weaker proposition for auxiliary push-down automata.

Theorem 29 CoStrongFewUAuxPDA $\subseteq$ ReachUAuxPDA*.*

Proof. Since CoStrongUAuxPDA $\subseteq$ ReachUAuxPDA, and the latter class is closed under conjunctive log space many one reductions, we have L_c(CoStrongUAuxPDA) $\subseteq$ ReachUAuxPDA. Clearly, L_c(CoStrongUAuxPDA) equals Co(L_d(StrongUAuxPDA)), which coincides with CoStrongFewUAuxPDA by an analog of Proposition 23. □

It should be noted that by a modification of the proof of Proposition 28 also the following result can be obtained:

Corollary 30 StrongFewUAuxPDA $\subseteq$ ReachUAuxPDA*.*

Both ReachUL and StrongUL are contained in LOGDCFL. This means that they can be simulated deterministically with time bound remaining polynomial and only a moderate increase of space usage. In some sense ReachUL and StrongUL have a "deterministic flavor". The next proposition shows that this is unlikely to hold also for ReachUAuxPDA or StrongUAuxPDA.

Theorem 31 [LR90] LOGUCFL $\subseteq$ StrongUAuxPDA*.*

3.3 Few Reachability

As for log space machines in Subsection 2.3 we generalize the classes ReachUAuxPDA and StrongUAuxPDA from one computation path to any configuration (resp. between any two configurations) to a polynomial number of paths.

Definition 32 *The class* ReachFewAuxPDA *is defined as the subclass of languages in* FewAuxPDA *for which there exists an auxiliary push-down automaton, which satisfies for any input x and for any configuration c that there are at most polynomially many paths from the start configuration to c.*

Definition 33 *The class* StrongFewAuxPDA *is defined as the subclass of languages in* FewAuxPDA *for which there exists an auxiliary push-down automaton, which satisfies for any input x and for any two configurations c_1 and c_2 that there are at most polynomially many paths between c_1 and c_2.*

StrongFewAuxPDA is called NAPDA-SPACE,AMBIGUITY($\log n, n^{O(1)}$) in [NR90].

Though a ReachFewL machine can have up to polynomially many computation paths to any configuration, it can be simulated deterministically in polynomial time. The price that must be paid is an additional push-down store. An even more surprising result can be obtained for the analog class StrongFewAuxPDA. Languages of this class can be recognized by AuxPDAs with at most one accepting computation. The price that has to be paid is that polynomial reachability of rejecting configurations is lost.

Theorem 34 *[NR90]* StrongFewAuxPDA $\subseteq$ UAuxPDA.

Finally, by storing its computation on a push-down store, a FewL machine can be simulated by some ReachFewAuxPDA machine.

Theorem 35 FewL $\subseteq$ ReachFewAuxPDA.

An overview of the inclusion relationships of the classes investigated can be found on the following page.

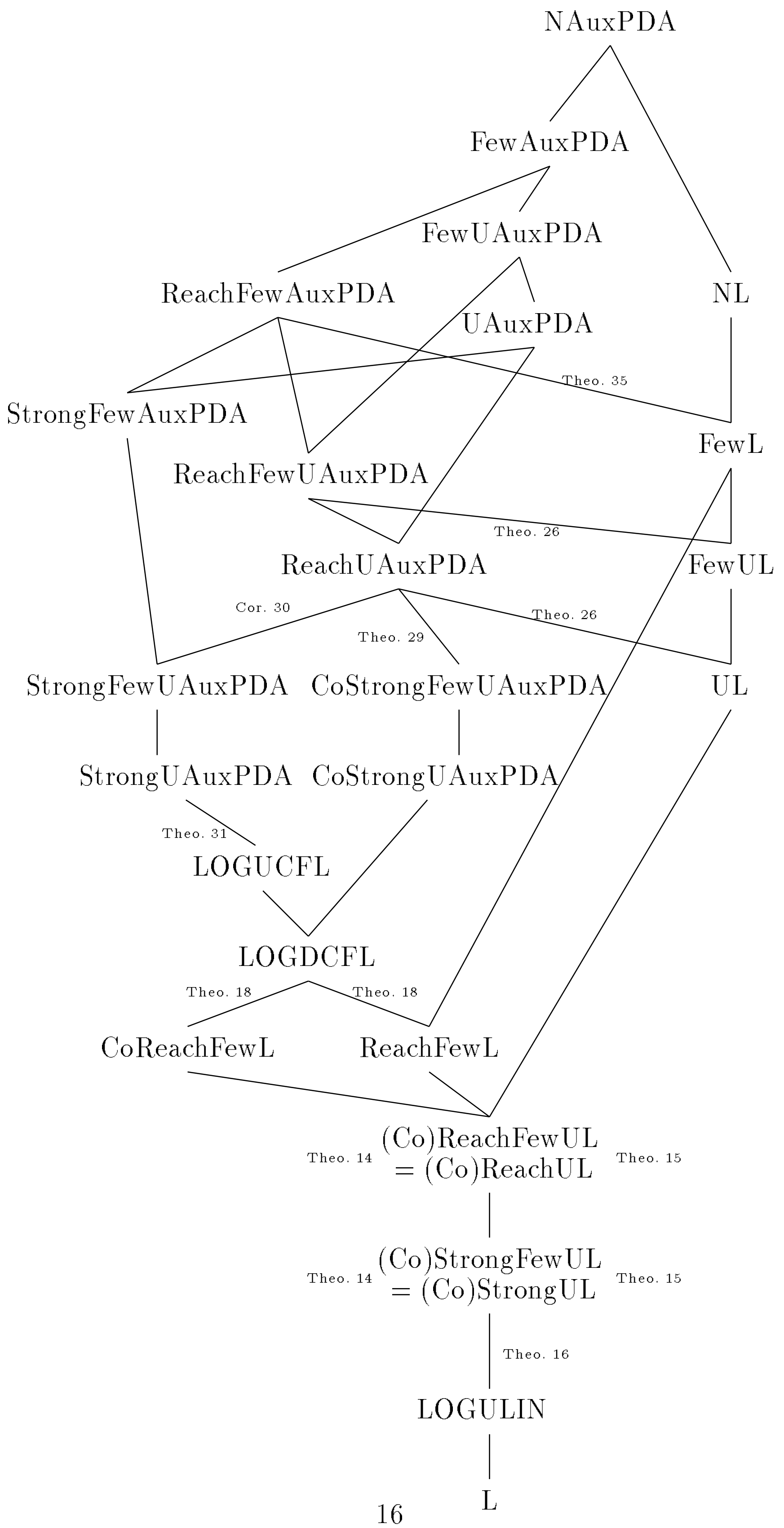
Acknowledgement

We thank Inga Niepel for helpful discussions and suggestions.

References

[AJ90] Carme Àlvarez and Birgit Jenner. A very hard log space counting class. In *Proc. of 5th Conference on Structure in Complexity Theory*, pages 154–168, 1990.

[All86] E. W. Allender. The complexity of sparse sets in P. In *Proc. of 1st Structure in Complexity Conf.*, number 223 in LNCS, pages 1–11. Springer, 1986.



NAuxPDA
FewAuxPDA
FewUAuxPDA
ReachFewAuxPDA
UAuxPDA
NL
Theo. 35
StrongFewAuxPDA
FewL
ReachFewUAuxPDA
Theo. 26
ReachUAuxPDA
FewUL
Cor. 30
Theo. 26
Theo. 29
StrongFewUAuxPDA
CoStrongFewUAuxPDA
UL
StrongUAuxPDA
CoStrongUAuxPDA
Theo. 31
LOGUCFL
LOGDCFL
Theo. 18
Theo. 18
CoReachFewL
ReachFewL
Theo. 14
(Co)ReachFewUL = (Co)ReachUL
Theo. 15
Theo. 14
(Co)StrongFewUL = (Co)StrongUL
Theo. 15
Theo. 16
LOGULIN
L

[BCD$^+$88] A. Borodin, S. A. Cook, P. Dymond, W. L. Ruzzo, and M. Tompa. Two applications of complementation via inductive counting. In *Proc. of 3rd Conference on Structure in Complexity*, 1988.

[BDG88] J. Balcácar, J. Diáz, and J. Gabárro. *Structural Complexity Theory I.* Springer, 1988.

[BDG90] J. Balcácar, J. Diáz, and J. Gabárro. *Structural Complexity Theory II.* Springer, 1990.

[BDHM91] G. Buntrock, C. Damm, U. Hertrampf, and C. Meinel. Structure and importance of logspace-MOD-classes. In *Proc. of 7th STACS*, To appear in LNCS. Springer, 1991.

[BHS90] G. Buntrock, L. A. Hemachandra, and D. Siefkes. Using inductive counting to simulate nondeterministic computation. In *Proc. of 15th MFCS*, number 452 in LNCS, pages 187–194. Springer, 1990. (to appear in *Information and Computation*).

[CH89] J. Cai and L. A. Hemachandra. On the power of parity polynomial time. In *Proc. of 6th STACS*, number 349 in LNCS, pages 229–239. Springer, 1989.

[Coo71] S. A. Cook. Characterizations of pushdown machines in terms of time-bounded computers. *Journal of the ACM*, 18:4–18, 1971.

[DR86] P. Dymond and W. L. Ruzzo. Parallel RAMs with owned global memory and deterministic language recognition. In *Proc. of 13th ICALP*, number 226 in LNCS, pages 95–104. Springer, 1986.

[GS88] J. Grollmann and A. Selman. Complexity measures for public-key cryptosystems. *SIAM Journal on Computing*, 17:309–335, 1988.

[HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley Series in Computer Science. Addison-Wesley, 1979.

[Imm88] N. Immerman. Nondeterministic space is closed under complement. *SIAM Journal on Computing*, 17(5):935–938, 1988.

[JK89] B. Jenner and B. Kirsig. *Alternierung und Logarithmischer Platz.* Dissertation, Universität Hamburg, 1989.

[JKL89] B. Jenner, B. Kirsig, and K.-J. Lange. The logarithmic alternation hierarchy collapses. *Inform. and Comp.*, 80:269–288, 1989.

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

[LR90] K.-J. Lange and P. Rossmanith. Characterizing unambiguous augmented pushdown automata by circuits. In *Proc. of 15th MFCS*, number 452 in LNCS, pages 399–406. Springer, 1990.

[Nie91] I. Niepel. Personal communication, 1991.

[NR90] R. Niedermeier and P. Rossmanith. *Unambiguous Simulations of Auxiliary Pushdown Automata and Circuits.* SFB-Bericht 342/31/90 A, I9054, Institut für Informatik, Technische Universität München, Arcisstr. 21, D-8000 München 2, December 1990.

[Ros91] P. Rossmanith. The owner concept for PRAMs. In *Proc. of 8th STACS*, number 480 in LNCS, pages 172–183. Springer, 1991.

[RST84] W. L. Ruzzo, J. Simon, and M. Tompa. Space-bounded hierarchies and probabilistic computations. *Journal of Computer and System Sciences*, 28:216–230, 1984.

[Sud78] I. H. Sudborough. On the tape complexity of deterministic context-free languages. *Journal of the ACM*, 25:405–414, 1978.

[Sze88] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26:279–284, 1988.

[Val76] L. Valiant. The relative complexity of checking and evaluating. *Inform. Proc. Letters*, 5:20–23, 1976.