UNIVERSITÄT TRIER
Mathematik / Informatik

Forschungsbericht Nr. 95-01

Expressing Uniformity via Oracles

Carsten Damm Markus Holzer Peter Rossmanith

ISSN 0944-0488

Expressing Uniformity via Oracles

Carsten Damm [*†] Markus Holzer [‡†] Peter Rossmanith [‡§]

Abstract

We study under what circumstances different uniformity notions for Boolean circuits of logarithmic depth lead to the same complexity class. Our investigations are based on a characterization of uniformity in terms of oracle access to tally sets that is proved in the present paper: $\mathcal{A}$-uniform circuits of logarithmic depth are of the same computational power as DLOGTIME-uniform circuits of logarithmic depth with oracle access to tally sets in $\mathcal{A}$. This characterization does not only apply to classes $\mathcal{A}$ such as logarithmic space or polynomial time, but to all in some sense "well-behaved" classes and especially to all standard complexity classes. We present many applications for this characterization, among them upward separations, depth–uniformity trade-offs, and inclusion-completeness results for tally languages.

1 Introduction

A circuit family of logarithmic depth (we will also simply speak of *logdepth circuits*) is a sequence $C = (C_n)$ of bounded fan-in circuits with $\mathrm{depth}(C_n) = O(\log n)$. If there is no restriction concerning the constructability of the circuits in the sequence, logdepth circuits can recognize non-recursive languages. In order to make circuit complexity classes comparable to Turing machine based complexity classes, one poses uniformity conditions on the circuit families. Concerning logdepth circuits the most often considered uniformity conditions are *logspace*, *polynomial time* and DLOGTIME-uniformity. We use L-*uniform*-NC^1, P-*uniform*-NC^1, and NC^1 as shorthand notations for the classes of languages recognized by logdepth circuits with these uniformities. Obviously $\mathrm{NC}^1 \subseteq$ L-*uniform*-$\mathrm{NC}^1 \subseteq$ P-*uniform*-NC^1. However, the exact relations are not known. For example, the best known simulation of L-*uniform*-NC^1 by DLOGTIME-uniform circuits requires already depth $\log^2 n$ (see Ruzzo [11]). On the other hand there are problems computable in P-*uniform*-NC^1, but not known to be computable in NC^1 [3].

The question whether or not, e.g., L-*uniform*-NC^1 = P-*uniform*-NC^1 does not seem to be equivalent to the question whether $\mathrm{L} = \mathrm{P}$. This is because polynomial time uniform circuits can not make use of all the computational power of the complexity class P. But what features of P-computations *can* be used? One of the main aims of the present paper is to give one possible precise answer to this question: polynomial time uniform circuits make use of the computational power comprised in *tally* languages in P, i.e., sets of words

*Fachbereich IV — Informatik, Universität Trier, D-54286 Trier, Germany

†Partially supported by DFG-La 618/1-1

‡Fakultät für Informatik, Technische Universität München, Arcisstr. 21, D-80290 München, Germany

§Supported by DFG-SFB 0342 "KLARA"

over a one-letter alphabet, recognizable in polynomial time. A similar statement holds for **L**-uniform logdepth circuits and moreover for $\mathcal{A}$-uniform logdepth circuits where $\mathcal{A}$ is a class fulfilling some weak closure properties.

Allender [1] studied the same question concerning **P**-uniform polylogarithmic depth circuits (polynomial size and depth $\log^{O(1)}$). Our results are similar to Allender's, but the proofs are more subtle. This is because we build on **DLOGTIME**-uniformity [1] and because our results are applicable to more general classes than **P**.

In the following section we introduce the necessary notions. In Section 3 we prove the main theorem — a characterization of uniformity conditions for logdepth circuits in terms of oracles. In Section 4 we apply it to prove some upward separation results concerning NC^1, **L**-*uniform*-NC^1, and **P**-*uniform*-NC^1, which support the believe that these classes are pairwise different. In Section 5 we discuss further applications.

2 Definitions

The reader is assumed to be familiar with the definition of Boolean circuits as developed in [7]. Especially we will consider circuits with oracle gates. Let $B \subseteq \{0,1\}^*$. A circuit with oracle B is a circuit which contains *B-gates*, i.e., gates that compute value 1 iff the string of bits computed by the predecessor gates belongs to B (we assume a prescribed order on the predecessor gates, and hence on the bits they compute). In the obvious way a circuit with oracle computes a certain Boolean function on the input variables. The depth of a circuit is defined as above except that oracle gates with fan-in m count as depth $\log m$ (logarithms are to base 2 throughout the paper). Unless otherwise stated the notion *circuit* will always refer to a circuit with oracle.

A circuit family is a sequence $C = (C_n)$ of circuits such that C_n is a circuit of n input variables. Let $C = (C_n)$ be a circuit family and let $f = (f_n)$ be the corresponding sequence of Boolean functions computed by C. Then the *language recognized by C* is the set $\bigcup_{i\in\mathbf{N}} f_n^{-1}(1)$.

To define uniformity we will adopt the definition of Ruzzo [11] extended to circuits with oracles (see Wilson [14]). Ruzzo's definition relies on bounding the descriptional complexity of certain local properties of the circuits in the sequence. This approach is of interest here because of its applicability to low circuit complexity classes (see Barrington *et al.* [2]).

The extended connection language $L_{ec}(C)$ of a circuit family $C = (C_n)$ is the set of strings of one of the forms $\langle 1^n\#g,\tau\rangle$ or $\langle 1^n\#g,p,g'\rangle$, where, in the former case, gate number g is of type τ and in the latter case, p, with $|p| \le \log(\mathrm{size}(C_n))$, describes a path in C_n from gate number g to gate number g'. p is the string of the binary encoded relative predecessor numbers successively leading from g to g' separated $\#$. We call these strings *Type 1* and *Type 2 strings* of $L_{ec}(C)$.

Let $\mathcal{A}$ denote a certain class of languages (e.g., a complexity class). A circuit family $C = (C_n)$ is said to be *$\mathcal{A}$-uniform*, if the extended connection language $L_{ec}(C)$ of C belongs to $\mathcal{A}$. We denote the class of languages which are recognized by $\mathcal{A}$-uniform circuits $C = (C_n)$ with $\mathrm{depth}(C_n) = O(\log n)$ by $\mathcal{A}$-*uniform*-NC^1. If the circuits are allowed to have oracle gates for a language B, the corresponding complexity class is

[1] this is not necessary when considering polylogarithmic depth circuits since **DLOGTIME**-uniformity and logspace uniformity coincide if depth $\log^2 n$ is allowed.

denoted by $\mathcal{A}$-*uniform*-$\mathrm{NC}^1(B)$. If a language D is computed by an $\mathcal{A}$-uniform logdepth circuit with oracle gates for the language B, it is said to be $\mathcal{A}$-*uniform*-NC^1-*reducible* to B. Finally, if oracle gates for languages out of a class $\mathcal{B}$ of languages are allowed, we denote the corresponding complexity class by $\mathcal{A}$-*uniform*-$\mathrm{NC}^1(\mathcal{B})$.

We will mostly be concerned with **DLOGTIME**-uniform circuits. Here the extended connection language must be recognized by a Turing machine operating in deterministic time $O(\log n)$. These machines are equipped with a binary *index tape* (see [11]). If a certain number i is written on this index tape, the input head *in one step* moves to the ith bit of the input.

We will abbreviate **DLOGTIME**-*uniform*-NC^1 by NC^1.

For a class $\mathcal{A}$ of languages Tally$\mathcal{A}$ denotes the class of all tally languages in $\mathcal{A}$, i.e., languages over a one letter alphabet.

Except of the notations introduced here we make use of common notations for complexity classes, such as **L**, **P**, **DLBA** standing for the classes of languages recognized by deterministic Turing machines within $O(\log n)$ space, $n^{O(1)}$ time, and $O(n)$ space, respectively. Further notations will be introduced as they are needed.

3 Uniformity and Tally Oracles

In this section we develop a general understanding of a uniformity condition that is implied by the presence of a tally oracle, i.e., we show that a tally oracle Tally$\mathcal{A}$ corresponds to $\mathrm{NC}^1(\text{Tally}\mathcal{A})$ uniformity.

Theorem 1 $\mathrm{NC}^1(\text{Tally}\mathcal{A}) = \mathrm{NC}^1(\text{Tally}\mathcal{A})$-*uniform*-$\mathrm{NC}^1$.

The proof of Theorem 1 is contained in two technical lemmas at the end of this section. Before this we show some consequences. At first glance, Theorem 1 presents the connection between oracles and uniformity in a rather easy way. This is due to the generality (there are no preconditions after all). For nice classes like **L**, **P**, **NP**, etc., we can do much better. In general, let us call in this paragraph a language *padded*, if all words are of the form $1^n\#w$, such that $|w| = O(\log n)$. The *tally image* of a padded language is defined via $1^n\#w \mapsto 1^{\langle n,w\rangle}$. Here $\langle\,,\rangle$ denotes any NC^1-computable pairing function $\{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ and further we identify a binary string u with the integer whose binary representation is $1u$. A class of languages is closed under tally image, if it contains the tally image of each of its padded languages. It is easy to see that all classes mentioned above are *closed under tally image*. A simple sufficient condition for the closure under tally image is the closure under NC^1 reductions. **NP** is a candidate that is closed under tally image, but it is unknown whether it is also closed under NC^1 reductions — any language is NC^1-reducible to its complement.

Lemma 2 *If $\mathcal{A}$ is closed under tally image, then each $\mathcal{A}$-uniform-NC^1 circuit is also $\mathrm{NC}^1(\text{Tally}\mathcal{A})$-uniform.*

Proof Let $L_{ec}(C)$ be the extended connection language of some given $\mathcal{A}$-*uniform*-NC^1 circuit $C = (C_n)$.

We describe an $\mathrm{NC}^1(\text{Tally}\mathcal{A})$ circuit family $C' = (C'_n)$ recognizing $L_{ec}(C)$. The extended connection language by definition is a padded language, i.e. each string in $L_{ec}(C)$

is of the form $1^n\#w$ with $|w| \leq c \cdot \log n$ for some sufficiently large c. We call such strings *admissible*. It is easy to filter out non-admissible strings, therefore in the following we consider only admissible inputs to the circuits in the family C'.

The oracle gates used in C' recognize exactly the tally image of $L_{ec}(C)$. So we must only ensure that the circuit sequence is **DLOGTIME**-uniform. On an input of the form $1^n\#w$ the circuit contains an oracle gate for each possible tally image of strings of the form $1^n\#w$ with $|w| \leq c \cdot \log n$. This is only a polynomial number of oracle gates.

Let the input be $\langle 1^n\#w \rangle$, where $w = g_1, p, g_2$. To decide whether this is a Type 2 string of $L_{ec}(C)$ the circuit simply computes whether the output value of the oracle gate that corresponds to the input is one. This can be done in a simple way by comparison sub-circuits for each possible w that share an AND gate with the corresponding oracle gate and taking an OR over all these AND gates. Similarly Type 1 strings of $L_{ec}(C)$ can be recognized. □

If $\mathcal{A}$ is closed under general NC^1 reductions then also the reverse of Lemma 2 holds: $\mathcal{A}$-*uniform*-$\mathrm{NC}^1 = \mathrm{NC}^1(\mathrm{Tally}\mathcal{A})$. Hence

Corollary 3 **P**-*uniform*-$\mathrm{NC}^1 = \mathrm{NC}^1(\mathrm{Tally}\mathbf{P})$, **L**-*uniform*-$\mathrm{NC}^1 = \mathrm{NC}^1(\mathrm{Tally}\mathbf{L})$.

This corollary is the base for many applications in later sections. The remainder of this section is devoted to the proof of Theorem 1. The first lemma we need, establishes a normal form for circuits with a tally oracle.

Lemma 4 *If C is an $\mathrm{NC}^1(\mathrm{Tally}\mathcal{A})$ circuit, then there is an $\mathrm{NC}^1(\mathrm{Tally}\mathcal{A})$ circuit C' that recognizes the same language as C and has the property that all inputs to oracle gates are constant and have the value one.*

Proof An oracle gate with inputs $y_1, \ldots, y_m$ can be replaced by $y_1 \wedge \cdots \wedge y_m \wedge o$, where o is an oracle gate that has m inputs which are connected to the constant one. □

Lemma 5 $\mathrm{NC}^1(\mathrm{Tally}\mathcal{A})$-*uniform*-$\mathrm{NC}^1(\mathrm{Tally}\mathcal{A}) \subseteq \mathrm{NC}^1(\mathrm{Tally}\mathcal{A})$-*uniform*-$\mathrm{NC}^1$.

Proof Let $C = (C_n)$ be an $\mathrm{NC}^1(\mathrm{Tally}\mathcal{A})$-uniform $\mathrm{NC}^1(\mathrm{Tally}\mathcal{A})$ circuit. By definition the extended connection language of C is accepted by some $\mathrm{NC}^1(\mathrm{Tally}\mathcal{A})$ circuit, i.e., $L_{ec}(C) \in \mathrm{NC}^1(\mathrm{Tally}\mathcal{A})$.

What we have to do is to prove the existence of an logdepth circuit family C' that is also $\mathrm{NC}^1(\mathrm{Tally}\mathcal{A})$ uniform (i.e., $L_{ec}(C') \in \mathrm{NC}^1(\mathrm{Tally}\mathcal{A})$), but uses no oracle nodes.

By Lemma 4 we can assume w.l.o.g. that all inputs to oracle gates in C are connected only to a constant gate of value one. We get C' from C by replacing each oracle gate by a constant gate of value 1 or 0 according to the value that is computed by the oracle gate. This is possible, since this value does not depend on the inputs $x_1, \ldots, x_n$. The number of gates in C and C' is the same and the names also remain unchanged. A gate g is connected to a gate g' in C via a path p iff $\langle 1^n\#g, p, g' \rangle \in L_{ec}(C)$, and if there is no oracle gate on the path p, then g and g' are also connected via p. Let $p = p_1\#p_2\# \cdots \#p_m$. If there really is some oracle gate on the path, it is the gate on top of g' due to the normal form of C. This happens iff there is some g'' such that $\langle 1^n\#g, p_1\#p_2\# \cdots \#p_{m-1}, g'' \rangle \in$

$L_{ec}(C)$ and $\langle 1^n\#g'', \text{ORACLE}\rangle \in L_{ec}(C)$. The corresponding circuit that computes whether $\langle 1^n\#g, p, g'\rangle \in L_{ec}(C')$ holds, looks as follows:

$$\begin{aligned}\langle 1^n\#g, p, g'\rangle \in L_{ec}(C') \quad &\equiv \quad \langle 1^n\#g'', \text{ORACLE}\rangle \notin L_{ec}(C) \wedge \\ &\wedge \langle 1^n\#g, p, g'\rangle \in L_{ec}(C). \qquad (1)\end{aligned}$$

The predicates $\langle 1^n\#g'', \text{ORACLE}\rangle \in L_{ec}(C)$ and $\langle 1^n\#g, p_1\# \cdots \#p_{m-1}, g''\rangle \in L_{ec}(C)$ can both be computed by $\mathbf{NC}^1(\text{Tally}\mathcal{A})$ circuits, since it holds $L_{ec}(C) \in \mathbf{NC}^1(\text{Tally}\mathcal{A})$. Clearly, the circuit computing the predicate $\langle 1^n\#g, p, g'\rangle \in L_{ec}(C')$ is then also an $\mathbf{NC}^1(\text{Tally}\mathcal{A})$ circuit.

Next we construct $\mathbf{NC}^1(\text{Tally}\mathcal{A})$ circuits for the predicates $\langle 1^n\#g, t\rangle \in L_{ec}(C')$. For $t \in \{x_1, \dots, x_n, \wedge, \vee, \neg, 0, 1\}$ we have $\langle 1^n\#g, t\rangle \in L_{ec}(C) \Rightarrow \langle 1^n\#g, t\rangle \in L_{ec}(C')$, since we don't want to change the type of non-oracle nodes. However, if $\langle 1^n\#g, \text{ORACLE}\rangle \in L_{ec}(C)$ we have $\langle 1^n\#g, t\rangle \in L_{ec}(C')$, with either $t = 0$ or $t = 1$. Assume the oracle gate g has m inputs. Then $t = 1$ iff $0^m \in \text{Tally}\mathcal{A}$.

There is an $\mathbf{NC}^1$ circuit that computes the predicate $\text{EXACT}_m(y_1, \dots, y_n)$, which determines, whether there are exactly m variables y_i with $y_i = 1$.

If g is an oracle, then

$$\langle 1^n\#g, 1\rangle \in L_{ec}(C')$$

can be computed as follows:

$$\langle 1^n\#g, 1\rangle \in L_{ec}(C') \equiv \bigvee_m \text{EXACT}_m(E_1, \dots, E_l) \wedge 0^m \in \text{Tally}\mathcal{A},$$

where

$$E_i \equiv \bigvee_{g'} \langle 1^n\#g, i, g'\rangle \in L_{ec}(C').$$

E_i means that the ith input of g is connected to some gate (which then is the constant one). $\text{EXACT}_m(E_1, \dots, E_l)$ is true iff g has exactly m inputs. $\langle 1^n\#g, 1\rangle \in L_{ec}(C')$ is true iff the oracle g outputs a one, which is exactly what we wanted. Similarly, an $\mathbf{NC}^1(\text{Tally}\mathcal{A})$ circuit can be constructed for $\langle 1^n\#g, 0\rangle \in L_{ec}(C')$.

We have shown that there exists an $\mathbf{NC}^1(\text{Tally}\mathcal{A})$ circuit for each possible w of the form $\langle 1^n\#g, p, g'\rangle$ or $\langle 1^n\#g, t\rangle$ that computes whether it holds $w \in L_{ec}(C')$. Since there are only polynomially many such words w, it is easy to construct an $\mathbf{NC}^1(\text{Tally}\mathcal{A})$ circuit that accepts exactly $L_{ec}(C')$. That means that C' is $\mathbf{NC}^1(\text{Tally}\mathcal{A})$ uniform. □

Lemma 6 $\mathbf{NC}^1(\text{Tally}\mathcal{A})$-*uniform*-$\mathbf{NC}^1 \subseteq \mathbf{NC}^1(\text{Tally}\mathcal{A})$.

Proof Let C be an arbitrary $\mathbf{NC}^1(\text{Tally}\mathcal{A})$-uniform family of $\mathbf{NC}^1$ circuits. We assume w.l.o.g. that C has the following normal form: C is leveled and strongly alternating, i.e., a level of OR-gates is followed by a level of AND-gates and vice versa. C is therefore in fact a perfect binary tree whose leaves will be denoted by $l_1, \dots, l_m$. Each leaf l_i is actually an input x_k or a negated input $\bar{x}_k$. Let $Path(i)$ denote the path description from the root of C to the ith leaf l_i. Clearly, l_i corresponds to input x_k (resp. $\bar{x}_k$) if and only if $\langle 1^n\#r, Path(i), x_k\rangle \in L_{ec}(C)$ (resp. $\langle 1^n\#r, Path(i), \bar{x}_k\rangle \in L_{ec}(C)$). Here r denotes the root of C.

In order to get a **DLOGTIME**-*uniform*-$\mathrm{NC}^1(\mathrm{Tally}\mathcal{A})$ circuit that simulates C, we replace each leaf l_i by a **DLOGTIME**-uniform subcircuit L_i which computes exactly the same value as l_i.

The L_i-sub-circuits use an oracle T defined as follows:

$$1^{Bin(n,i,x_k)} \in T \iff l_i \text{ corresponds to } x_k \text{ in } C,$$

$$1^{Bin(n,i,\bar{x}_k)} \in T \iff l_i \text{ corresponds to } \bar{x}_k \text{ in } C.$$

T in binary encodes the connections of leaves to inputs. It holds $T \in \mathrm{NC}^1(\mathrm{Tally}\mathcal{A})$, since T can be reduced to $L_{ec}(C)$ via an NC^1 many-one reduction:

$$1^{Bin(n,i,x_k)} \mapsto \langle 1^n\#r, Path(i), x_k\rangle,$$

$$1^{Bin(n,i,\bar{x}_k)} \mapsto \langle 1^n\#r, Path(i), \bar{x}_k\rangle.$$

$\square$

4 Upward Separations

Up- and downward separation results are closely related to padding- or translation techniques. To prove upward separations it is useful to consider tally languages and study the relationships between languages and their unary versions (see Book *et al.* [4] or Savitch [13]).

Let $\mathsf{ATISP}(t(n), s(n))$ denote the class of languages recognized by alternating Turing machines that are $t(n)$-time-bounded and $s(n)$-space-bounded. Further let **ALINTIME** and **ALOGTIME**, respectively, denote the class of languages accepted by alternating Turing machines within time $O(n)$ and $O(\log n)$, respectively. Observe, $\mathrm{NC}^1 = \mathsf{ALOGTIME}$ by Barrington *et al.* [2].

In the sequel $T_A := \{\, 1^w \mid w \in A \,\}$ denotes the unary encoding of a language $A \subseteq \{0,1\}^*$.

Lemma 7 *For $L \subseteq \{0,1\}^*$ the following holds:*

a) $T_A \in \mathrm{NC}^1$ *if and only if* $A \in \mathsf{ALINTIME}$*,*

b) $T_A \in \mathsf{L}$ *if and only if* $A \in \mathsf{DLBA}$*, and*

c) $T_A \in \mathsf{P}$ *if and only if* $A \in \mathsf{DTIME}(2^{O(n)})$*.*

Proof We prove only a). The other equivalences follow in the same manner.

Let $A \in \mathsf{ALINTIME}$. To decide whether $1^w \in T_A$ one first computes w (this can be done by a **DLOGTIME**-machine doing binary search), and then starts the **ALINTIME**-algorithm on w. Since $|w| = O(\log |1^w|)$ the whole algorithm works within alternating logarithmic time and hence, $T_A \in \mathsf{ALOGTIME}$.

Let on the other hand $T_A \in \mathsf{ALOGTIME}$. To decide whether $w \in A$, one simulates the behavior of an **ALOGTIME**-machine $\mathcal{M}$ recognizing T_A on input 1^w. To do this, one keeps a counter of length $|w|$ to store the input head position of $\mathcal{M}$. Since $\mathcal{M}$ works in time $O(\log |1^w|)$, this simulation works in time $O(|w|)$. $\square$

The following proposition is an immediate consequence of Lemma 7.

Proposition 8 *a)* $\text{Tally}\mathbf{NC}^1 = \text{Tally}\mathbf{L}$ *if and only if* $\mathbf{ALINTIME} = \mathbf{DLBA}$

b) $\text{Tally}\mathbf{NC}^1 = \text{Tally}\mathbf{P}$ *if and only if* $\mathbf{ALINTIME} = \mathbf{DTIME}(2^{O(n)})$

c) $\text{Tally}\mathbf{L} = \text{Tally}\mathbf{P}$ *if and only if* $\mathbf{DLBA} = \mathbf{DTIME}(2^{O(n)})$

Now with the help of Lemma 7 and Proposition 8 we prove the following:

Theorem 9 *a)* $\mathbf{NC}^1 = \mathbf{L}$-*uniform*-$\mathbf{NC}^1$ *if and only if* $\mathbf{ALINTIME} = \mathbf{DLBA}$

b) $\mathbf{NC}^1 = \mathbf{P}$-*uniform*-$\mathbf{NC}^1$ *if and only if* $\mathbf{ALINTIME} = \mathbf{DTIME}(2^{O(n)})$

c) $\mathbf{L}$-*uniform*-$\mathbf{NC}^1 = \mathbf{P}$-*uniform*-$\mathbf{NC}^1$ *if and only if* $\mathbf{DLBA} = \mathbf{DTIME}(2^{O(n)})$

Proof We only prove a). By the same methods b) and c) are proved.

($\Longrightarrow$) By Lemma 7, $A \in \mathbf{DLBA}$ iff $T_A \in \text{Tally}\mathbf{L}$, i.e., $T_A \in \mathbf{L}$-*uniform*-$\mathbf{NC}^1 = \mathbf{NC}^1$. Since T_A is a tally language, $T_A \in \text{Tally}\mathbf{NC}^1$. Now by Lemma 7, $T_A \in \text{Tally}\mathbf{NC}^1$ iff $A \in \mathbf{ALINTIME}$ and therefore $\mathbf{ALINTIME} = \mathbf{DLBA}$.

($\Longleftarrow$) By Proposition 8, $\mathbf{ALINTIME} = \mathbf{DLBA}$ iff $\text{Tally}\mathbf{NC}^1 = \text{Tally}\mathbf{L}$. The latter equality holds also for the $\mathbf{NC}^1$-closures and therefore with Corollary 3 follows $\mathbf{NC}^1 = \mathbf{L}$-*uniform*-$\mathbf{NC}^1$. □

Let $\mathbf{NC}$ denote the class of languages accepted by $\mathbf{DLOGTIME}$-uniform circuits of depth $(\log n)^{O(1)}$ and size $O(n^{O(1)})$. With the same translation technique as above we can prove:

Corollary 10 *If* $\mathbf{NC}^1 = \mathbf{P}$-*uniform*-$\mathbf{NC}^1$ *then* $\mathbf{NC} = \mathbf{P}$-*uniform*-$\mathbf{NC}$ *and* $\mathbf{PSPACE} = \mathbf{DTIME}(2^{n^{O(1)}})$.

Proof By Corollary 6.7 of Allender [1] we have $\mathbf{NC} = \mathbf{P}$-*uniform*-$\mathbf{NC}$ if and only if $\mathbf{PSPACE} = \mathbf{DTIME}(2^{n^{O(1)}})$. Therefore it suffices to show that $\mathbf{NC}^1 = \mathbf{P}$-*uniform*-$\mathbf{NC}^1$ implies $\mathbf{NC} = \mathbf{P}$-*uniform*-$\mathbf{NC}$.

Obviously $\mathbf{ALINTIME} \subseteq \mathbf{ATISP}(n^{O(1)}, n) \subseteq \mathbf{ASPACE}(n)$. By Corollary 3.5 of Chandra *et al.* [6] for $s(n) \geq \log n$ holds $\mathbf{ASPACE}(s(n)) = \bigcup_{c>0} \mathbf{DTIME}(2^{O(s(n))})$, hence $\mathbf{ASPACE}(n) = \mathbf{DTIME}(2^{O(n)})$. Now let $\mathbf{NC}^1 = \mathbf{P}$-*uniform*-$\mathbf{NC}^1$. By Theorem 2 holds $\mathbf{ALINTIME} = \mathbf{DTIME}(2^{O(n)})$. Therefore the above inclusions yield $\mathbf{ATISP}(n^{O(1)}, n) = \mathbf{ASPACE}(n)$, which by Corollary 6.6 of Allender [1] is equivalent to $\mathbf{NC} = \mathbf{P}$-*uniform*-$\mathbf{NC}$. □

5 Other Applications

5.1 Logspace-Printability and Logspace-Uniformity

Tally languages have often been studied in connection with other types of sparse sets. For $\mathbf{L}$-*uniform* circuits the results in Section 4 can be generalized to the class **printL** of $\mathbf{L}$-printable sets (see Jenner and Kirsig [9]) and to the class $\mathbf{SET\text{-}KS}[\log n, \log n]$ of languages with small Kolmogorov complexity (see Li and Vitányi [10]).

A language A is $\mathbf{L}$-*printable* if the function $1^n \mapsto A \cap \{0,1\}^n$ is computable in deterministic logspace. The class of $\mathbf{L}$-printable sets is denoted by $\mathbf{printL}$. The class of languages with small information content $\mathbf{SET\text{-}KS}[\log n, \log n]$ consists of all sets A, such that all but finitely many words in A, are in the set formed by strings w for which there is a word y, of length $|y| \leq \log |w|$, such that a fixed universal Turing machine $\mathcal{M}_U$ on input y produces w using at most $\log |w|$ work space.

Proposition 11 $\mathbf{L}$-*uniform*-$\mathrm{NC}^1 = \mathrm{NC}^1(\mathbf{printL}) = \mathrm{NC}^1(\mathbf{SET\text{-}KS}[\log n, \log n])$.

Proof First note that U_{BC}-*uniform*-NC^1 = $\mathbf{L}$-*uniform*-NC^1, where U_{BC}-*uniform*-NC^1 means, that the mapping $1^n \mapsto C_n$ is computable by a deterministic logspace bounded Turing machine (see Ruzzo [11]). Let $A \in \mathbf{printL}$. Since $\mathbf{L}$-printable sets are sparse, it is easy to build a logarithmic depth circuit family which, given a string of length n, decides if this string equals any one of the polynomial many strings in $A \cap \{0,1\}^n$. Since by definition the mapping $1^n \to A \cap \{0,1\}^n$ is computable in deterministic logspace this circuit family is U_{BC}-uniform. Hence $A \in U_{BC}$-*uniform*-NC^1 = $\mathbf{L}$-*uniform*-NC^1. The same holds for languages A with small information content. Further each tally set in $\mathbf{L}$ is obviously $\mathbf{L}$-printable and has small information content.

To prove the first equality we conclude as follows. By the remarks above and Corollary 3 we have

$$\mathbf{printL} \subseteq \mathbf{L}\text{-}\textit{uniform}\text{-}\mathrm{NC}^1 = \mathrm{NC}^1(\mathrm{Tally}\mathbf{L}) \subseteq \mathrm{NC}^1(\mathbf{printL})$$

from which

$$\mathrm{NC}^1(\mathbf{printL}) \subseteq \mathrm{NC}^1(\mathrm{NC}^1(\mathrm{Tally}\mathbf{L})) \subseteq \mathrm{NC}^1(\mathrm{NC}^1(\mathbf{printL}))$$

follows. Since NC^1-reductions are transitive the claim follows.

The proof for the second equality is similar. □

An immediate consequence of the characterization theorems for $\mathbf{L}$-*uniform*-NC^1 is:

Corollary 12 *The following are equivalent:*

a) $\mathrm{NC}^1 = \mathbf{L}$-*uniform*-$\mathrm{NC}^1$

b) $\mathrm{Tally}\mathbf{L} \subseteq \mathrm{NC}^1$

c) $\mathbf{printL} \subseteq \mathrm{NC}^1$

d) $\mathbf{SET\text{-}KS}[\log n, \log n] \subseteq \mathrm{NC}^1$.

It is not known, whether $\mathbf{L}$-*uniform*-NC^1 circuits can be characterized in terms of alternating Turing machines. Note that Ruzzo characterized the classes of the $\mathbf{L}$-*uniform* NC hierarchy (except of $\mathbf{L}$-*uniform*-NC^1), in terms of logspace bounded alternating Turing machines with restricted running time [11]. On the other hand, Allender described all classes of the $\mathbf{P}$-*uniform* NC hierarchy, in terms of logspace bounded alternating Turing machines with restricted input access [1]. Using the idea of restricted input access, we are able to characterize $\mathbf{L}$-*uniform*-NC^1.

Proposition 13 *$L \in$ L-uniform-NC^1 if and only if L is accepted by a logspace-bounded alternating Turing machine with binary index tape that accesses its input only during the first $O(\log n)$ steps and works deterministically afterwards.*

The proof of this theorem is similar to the proofs of Theorems 4.2 in [1] and Theorem 3 and 4 in [11]. Note that this machine model can even be described as an $O(\log n)$ time bounded alternating Turing machine with post-processing. By post-processing we mean, that the Turing machine works after a special state without access to the input, but therefore it works like a deterministic logspace machine. This model is similar to so-called "blind" Turing machines introduced by Hartmanis and Mahaney [8] (see also Jenner and Kirsig [9]).

5.2 Trade-offs between Depth and Uniformity

Uniformity of circuits is a resource. E.g., there are problems known to be solvable in P-*uniform*-NC^1 but not known to be solvable in NC^1[3]. The question arises what increase in the other resources of the circuits (size, depth) have to be paid in order to make circuits more uniform.

Let DLOGTIME-*uniform-Size-Depth*$(n^{O(1)}, d(n))$ denote the class of languages that can be recognized by DLOGTIME-uniform polynomial size circuits of depth $O(d(n))$. Ruzzo proved (in a more general setting) the following result [11]:

Proposition 14 *L-uniform-$\mathrm{NC}^1 \subseteq$ DLOGTIME-uniform-Size-Depth$(n^{O(1)}, a(n) \cdot \log n)$ holds or $a(n) = \log n$.*

This simulation result can be related to the following result which is a special case of Theorem 2 in [12]:

Proposition 15 *For $a(n) = \log n$ holds*

$$\mathrm{DSPACE}(n) \subseteq \mathrm{ATISP}(n \cdot a(2^n), n).$$

It is not known if one of these simulations can be improved to functions $a(n) = o(\log n)$. However with the help of Theorem 1 one can prove the following:

Proposition 16 *Let $a(n)$ be a nondecreasing function with positive integer values. Further let $a(n) \cdot \log n$ be computable in deterministic time $O(\log n)$ (given n in binary notation). Then the following are equivalent:*

a) L-*uniform*-$\mathrm{NC}^1 \subseteq$ DLOGTIME-*uniform-Size-Depth*$(n^{O(1)}, a(n) \cdot \log n)$

b) $\mathrm{DSPACE}(n) \subseteq \mathrm{ATISP}(n \cdot a(2^n), n)$.

Proof Let $s(n)$ and $t(n)$ be functions of at least logarithmic growth. Further let them both be computable in deterministic time $O(s(n))$ (given n in binary). A circuit family of size $2^{O(s(n))}$ is said to be U_E-uniform if it's extended connection language can be recognized in deterministic time $O(s(n))$. So **DLOGTIME**- and U_E-uniformity coincide as long as the circuits are of polynomial size. Let $U_E\text{-}SizeDepth(2^{O(s(n))}, t(n))$ denote the class of languages recognized by U_E-uniform circuits of size $2^{O(s(n))}$ and depth $t(n)$.

By [11] $\mathbf{ATISP}(t(n), s(n)) = U_E\text{-}SizeDepth(2^{O(s(n))}, t(n))$. The proof now follows by Theorem 1 and the usual translation technique as in Section 4. □

By the same methods one proves the following:

Proposition 17 *Let $a(n)$ be a nondecreasing function with positive integer values. Further let $a(n) \cdot \log n$ be computable in deterministic time $O(\log n)$ (given n in binary notation). Then the following are equivalent:*

a) $\mathbf{P}$-*uniform*-$\mathbf{NC}^1 \subseteq \mathbf{DLOGTIME}$-*uniform-Size-Depth*$(n^{O(1)}, a(n) \cdot \log n)$.

b) $\mathbf{DTIME}(2^{k \cdot n}) \subseteq \mathbf{ATISP}(n \cdot a(2^n), n)$ *holds for all k.*

5.3 Inclusion Completeness

Book *et al.* [5] considered consequences of the existence of tally inclusion complete sets for certain pairs of language classes. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be classes of languages and $\mathcal{C}_1 \subseteq \mathcal{C}_2$. A set $A \in \mathcal{C}_2$ is said to be $(\mathcal{C}_1, \mathcal{C}_2)$-*inclusion complete*, if $A \in \mathcal{C}_1$ implies $\mathcal{C}_1 = \mathcal{C}_2$. This notion is of course very weak, since for any pair of nonempty language classes as above inclusion complete languages exist. However, for tally languages the situation is not that obvious. Book *et al.* studied consequences of the existence of tally inclusion complete sets for $(\mathbf{P}, \mathbf{NP})$. With Theorem 1 we obtain similar results for complexity classes considered in this paper.

As an example we mention here only the results for the pair $(\mathbf{NC}^1, \mathbf{L})$.

Proposition 18 *The following are equivalent:*

a) There is a tally inclusion complete language for $(\mathbf{NC}^1, \mathbf{L})$

b) $\mathbf{NC}^1 = \mathbf{L}$ *if and only if* $\mathrm{Tally}\mathbf{NC}^1 = \mathrm{Tally}\mathbf{L}$.

c) $\mathbf{NC}^1 = \mathbf{L}$ *if and only if* $\mathbf{NC}^1 = \mathbf{L}$-*uniform*-$\mathbf{NC}^1$

Proof The equivalence of b) and c) follows from Corollary 3. Therefore it suffices to prove the equivalence of a) and b).

Suppose A is a tally inclusion complete language for $(\mathbf{NC}^1, \mathbf{L})$. Obviously from $\mathbf{NC}^1 = \mathbf{L}$ follows $\mathrm{Tally}\mathbf{NC}^1 = \mathrm{Tally}\mathbf{L}$. On the other hand if $\mathrm{Tally}\mathbf{NC}^1 = \mathrm{Tally}\mathbf{L}$ then $A \in \mathbf{NC}^1$ and by inclusion completeness $\mathbf{NC}^1 = \mathbf{L}$.

For the other direction suppose $\mathbf{NC}^1 = \mathbf{L}$ holds if and only if $\mathrm{Tally}\mathbf{NC}^1 = \mathrm{Tally}\mathbf{L}$. If $\mathrm{Tally}\mathbf{NC}^1 \neq \mathrm{Tally}\mathbf{L}$ then any $A \in \mathrm{Tally}\mathbf{L} - \mathrm{Tally}\mathbf{NC}^1$ is inclusion complete for $(\mathbf{NC}^1, \mathbf{L})$. In the other case if $\mathrm{Tally}\mathbf{NC}^1 = \mathrm{Tally}\mathbf{L}$ then any language in $\mathrm{Tally}\mathbf{L}$ is inclusion complete for $(\mathbf{NC}^1, \mathbf{L})$. □

Proposition 19 *If there is* no *tally inclusion complete language for* $(\mathrm{NC}^1, \mathbf{L})$ *then*

a) $\mathrm{NC}^1 \neq \mathbf{L}$

b) $\mathsf{ALINTIME} = \mathsf{DLBA}$.

Proof Part a) follows because $\mathrm{NC}^1 = \mathbf{L}$ implied that any language in Tally$\mathbf{L}$ where inclusion complete for $(\mathrm{NC}^1, \mathbf{L})$. By Proposition 18 c) from $\mathrm{NC}^1 \neq \mathbf{L}$ and the assumption follows $\mathrm{NC}^1 = \mathbf{L}$-*uniform*-$\mathrm{NC}^1$. Hence by Theorem 9 follows $\mathsf{ALINTIME} = \mathsf{DLBA}$. $\square$

Especially interesting is the following consequence which follows by combining Propositions 18 and 19.

Corollary 20 *One of the following statements is true:*

a) $\mathrm{NC}^1 = \mathbf{L}$ *if and only if* $\mathrm{NC}^1 = \mathbf{L}$-*uniform*-$\mathrm{NC}^1$

b) $\mathsf{ALINTIME} = \mathsf{DLBA}$.

5.4 Parallelizability and Self-Parallelizability

Let $\mathcal{B}$ be a complexity class. We can regard $\mathcal{B}$-*uniform*-NC^1 as the class of problems which are highly parallelizable with the help of $\mathcal{B}$'s computational power. A complexity class $\mathcal{A}$ is said to be *highly* $\mathcal{B}$*-parallelizable* if $\mathcal{A} = \mathcal{B}$-*uniform*-$\mathrm{NC}^1$. It is said to be *highly parallelizable* if it is highly $\mathcal{B}$-parallelizable for some class of languages $\mathcal{B}$. It is said to be *highly self-parallelizable* if $\mathcal{A} = \mathcal{A}$-*uniform*-$\mathrm{NC}^1$. For instance, NC^1 is highly self-parallelizable. We will prove that if a complexity class $\mathcal{A}$ is highly $\mathcal{B}$-parallelizable for a "well-behaved" class $\mathcal{B}$, it is already self-parallelizable:

Proposition 21 *Let* $\mathcal{A}$ *and* $\mathcal{B}$ *be complexity classes such that* $\mathcal{B}$ *is closed under tally image and* $\mathrm{NC}^1(\mathrm{Tally}\mathcal{B}) \subseteq \mathcal{B}$. *If* $\mathcal{A}$ *is highly* $\mathcal{B}$*-parallelizable, then it is highly self-parallelizable.*

Proof By the assumptions we have $\mathcal{A} = \mathcal{B}$-*uniform*-$\mathrm{NC}^1 = \mathrm{NC}^1(\mathrm{Tally}\mathcal{B})$. Therefore $\mathcal{A}$-*uniform*-$\mathrm{NC}^1 = \mathrm{NC}^1(\mathrm{Tally}\mathcal{B})$-*uniform*-$\mathrm{NC}^1 = \mathcal{B}$-*uniform*-$\mathrm{NC}^1 = \mathcal{A}$ by Lemma 6 and Theorem 1. $\square$

As a consequence of this fact we get the following:

Corollary 22 *If* $\mathbf{L} \neq \mathbf{L}$-*uniform*-$\mathrm{NC}^1$ *then there is no class of languages* $\mathcal{B}$ *such that* $\mathbf{L} = \mathrm{NC}^1(\mathrm{Tally}\mathcal{B})$*, or in other words, if* $\mathbf{L}$ *is highly parallelizable, then it is already highly self-parallelizable.*

Similar statements are true for other uniformity conditions.

Acknowledgments We would like to thank Klaus-Jörn Lange for drawing our attention to the subject and for his encouragement. Thanks also to Birgit Jenner for many useful remarks.

References

[1] E. W. Allender. P-uniform circuit complexity. *Journal of the Association for Computing Machinery*, 36(4):912–928, October 1989.

[2] D. A. M. Barrington, N. Immerman, and H. Straubing. On uniformity within $\mathcal{NC}^1$. *Journal of Computer and System Sciences*, 41:274–306, 1990.

[3] P. W. Beame, S. A. Cook, and H. J. Hoover. Log depth circuits for division and related problems. *SIAM Journal on Computing*, 15:994–1003, 1986.

[4] R. V. Book. Comparing complexity classes. *Journal of Computer and System Sciences*, 9:213–229, 1974.

[5] R. V. Book, C. Wrathall, A. L. Selman, and D. Dobkin. Inclusion complete tally languages and the Hartmanis-Berman conjecture. *Mathematical Systems Theory*, 11:1–8, 1977.

[6] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1): 114–133, 1981

[7] S. A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64:2–22, 1985.

[8] J. Hartmanis and S. Mahaney. Languages simultaneously complete for one-way and two-way log-tape automata. *SIAM Journal on Computing*, 10(2):383–390, 1981.

[9] B. Jenner and B. Kirsig. *Alternierung und Logarithmischer Platz*. Dissertation (in German), Universität Hamburg, 1989.

[10] M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and its applications*. Springer-Verlag, New York, Berlin, Heidelberg, 1993.

[11] W. L. Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences*, 22(3):365–383, 1981.

[12] W. L. Ruzzo. Tree-Size Bounded Alternation. *Journal of Computer and System Sciences*, 21(2):218–235, 1980.

[13] W. J. Savitch. A note on multihead automata and context-sensitive languages. *Acta Informatica*, 2:249–252, 1973.

[14] C. B. Wilson. Relativized Circuit Complexity. *Journal of Computer and System Sciences*, 31:169–181, 1985.