The lower reaches of circuit uniformity

Christoph Behle[1], Andreas Krebs[1], Klaus-Jörn Lange[1], and Pierre McKenzie[2]

[1] Wilhelm-Schickard-Institut, Universität Tübingen
{behlec,krebs,lange}@informatik.uni-tuebingen.de
[2] DIRO, Université de Montréal
mckenzie@iro.umontreal.ca

Abstract. The effect of severely tightening the uniformity of Boolean circuit families is investigated. The impact on NC^1 and its subclasses is shown to depend on the characterization chosen for the class, while classes such as P appear to be more robust. Tightly uniform subclasses of NC^1 whose separation may be within reach of current techniques emerge.

1 Introduction

Motivation. Uniformity is imposed on Boolean circuit families in order for circuit families to define classes of languages that correspond to machine-based classes. For example, logspace-uniform and polytime-uniform Boolean circuit families of polynomial size [Bor77] capture the class P, while non-uniform circuit families of constant size recognize undecidable languages.

Uniformity notions more permissive than the circuit resources under study were considered in the literature (e.g. [All89]). But tighter and tighter notions were needed to capture low complexity classes. Borodin [Bor77] and Cook [Coo79] first showed the usefulness of enforcing uniformity by means of space-bounded Turing machines computing circuit descriptions. This worked well for NC^2 and above. Inspired by Goldschlager [Gol78], Ruzzo [Ruz81] then tied uniformity to circuit connectivity queries. Ruzzo defined an ALOGTIME notion of uniformity under which NC^1 meaningfully equals ALOGTIME. Yet tighter notions were needed to investigate $\mathrm{AC}^0 \subset \mathrm{ACC}^0 \subseteq \mathrm{TC}^0 \subseteq \mathrm{NC}^1$. Barrington, Immerman and Straubing [BIS90] thus developed DLOGTIME-uniformity. They proved that the model-theoretic notion introduced by Immerman [Imm87] and the Turing machine-based notion studied by Buss [Bus87] were equivalent to their own.

Roy and Straubing [RS07] later triggered the need for an even stronger notion of uniformity than DLOGTIME. This requires some explaining, because DLOGTIME is surely the lowest meaningful Turing machine-based complexity class imaginable.

Motivation continued: enters descriptive complexity. Borrowing an example from [MTV10], the language of words $w \in \{a,b\}^\star$ having no b at an even position is described by the intuitive formula $\neg\exists i\big(\mathsf{Even}(i) \wedge Q_b(i)\big)$. In such a formula, the variables range over positions in w, the predicate Q_σ for $\sigma \in \{a,b\}$ holds at i iff $w_i = \sigma$, and the *numerical* predicate Even holds at i iff i is even. This example is a first-order formula, more precisely a $\mathrm{FO}[<, \mathsf{Even}]$ formula,

where the numerical predicates allowed are listed and they include "$i < j$" for the formalism to be able to describe words, i.e., (ordered) sequences of letters.

Descriptive complexity based on ExtFO (our appellation here for FO extended with generalized quantifiers) is tied to circuit complexity in two important ways. First, low circuit complexity classes have crisp ExtFO characterizations. Second, FO provides the desired notions of uniformity finer than DLOGTIME.

Indeed a language is in non-uniform AC^0 iff it can be described by a FO[arb] formula [Imm87], where arb means that "any arbitrary set of numerical predicates" is allowed. A language is in non-uniform TC^0 iff it can be described by a MAJ[arb] formula [BIS90], where MAJ refers to replacing "$\exists$" and "$\forall$" with the "there exist more than half of the possible positions i at which the formula holds" quantifier. And a language is in non-uniform NC^1 iff it can be described in $S_5 + \mathrm{FO}[\mathsf{arb}]$, where $S_5 + \mathrm{FO}$ refers to allowing, in addition to "$\exists$" and "$\forall$", a (Lindström) quantifier testing whether a sequence of five-point permutations associated with each position i composes to the identity permutation [Bar89].

Strikingly, replacing [arb] above with the pair of numerical predicates $[+, \times]$ imposes DLOGTIME-uniformity [BIS90]. For example, $\mathrm{FO}[+, \times]$ precisely equals DLOGTIME-uniform AC^0 and $S_5 + \mathrm{FO}[+, \times]$ equals DLOGTIME-uniform NC^1. The numerical predicates available to the ExtFO description of a circuit-based class of languages thus regulate the uniformity of the circuit families.

Roy and Straubing proved [RS07] that any regular language expressed in $\mathrm{MOD}_q + \mathrm{FO}[<, +]$ can be re-expressed without the non-regular[3] numerical predicate "+", where MOD_q refers to allowing the "there exist m modulo q positions i at which the formula holds" quantifiers. Roy and Straubing asked whether $\mathrm{MOD}_q + \mathrm{FO}[<, +]$ also has a circuit characterization. This was answered by the first and third authors who proved, using a new encoding for circuit connections, that $\mathrm{ExtFO}[<, \mathsf{X}]$ is meaningfully captured by $\mathrm{FO}[<, \mathsf{X}]$-uniform circuits, for any reasonable set X of numerical predicates [BL06].

Motivation concluded. Meaningful uniformity notions even tighter than DLOGTIME uniformity thus abound: examples are $\mathrm{FO}[<, +]$-uniformity and $\mathrm{FO}[<]$-uniformity. In the $\mathrm{FO}[<, +]$-uniform world, Roy and Straubing thus separated classes that are only *conjectured* to differ in the more usual $\mathrm{FO}[+, \times]$-uniform, i.e., DLOGTIME-uniform, world.

Our main motivation is to examine the impact of imposing $\mathrm{FO}[<]$-uniformity. Do the separations conjectured in the DLOGTIME-uniform world hold here? What happens to TC^0 and to the following well-known characterizations of NC^1,

Characterization	Depth	Fan-in	Size	Gates used
NC^1	$O(\log n)$	constant	poly	AND, OR, NOT
$\mathrm{AC}^0(M)$	$O(1)$	poly	poly	AND, OR, M
$\mathrm{AC}^0(\mathbb{D}_+)$	$O(1)$	poly	poly	AND, OR, D_+
$\mathrm{AC}^0(\mathbb{D}_+)_{\mathrm{LIN}}$	$O(1)$	linear	poly	AND, OR, D_+

[3] A numerical predicate is said to be regular if a finite automaton can compute it; much of the internal structure of (uniform and non-uniform) NC^1 hinges on a "*regularity*" conjecture [Str94, Conjecture IX.3.4], stating that ExtFO expressibility of a regular language never inherently requires a non-regular numerical predicate.

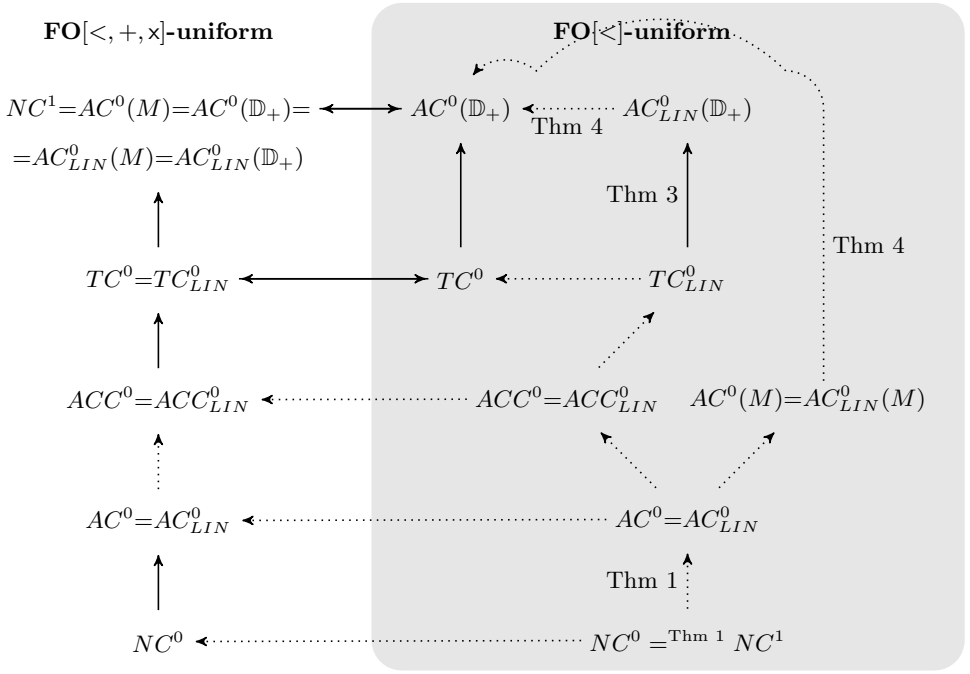

Fig. 1. Except in the case of NC^1 = ALOGTIME, FO[ ]-uniformity here refers to direct connection language expressivity under unary shuffled encoding (see Section 2). A solid arc, a bidirectional arc and a dashed arc from A to B denote $A \subseteq B$, $A = B$ and $A \subset B$ respectively. Arrows with no labels were known prior to this paper.

where $\mathrm{AC}^0(M)$ is the AC^0-closure of a set of word problems over finite monoids from a set M that contains at least one non-solvable monoid, e.g. A_5 [Bar89], where $\mathbb{D}_+$ is a variant of the Boolean formula value problem [Bus87,BCGR92] (see Section 2) and where $\mathrm{AC}^0(\mathbb{D}_+)_{\mathrm{LIN}}$ is $\mathrm{AC}^0(\mathbb{D}_+)$ with linear fanin gates?

Our results. See Figure 1. Together with some further observations, we deduce the following properties of the FO[<]-uniform world:

- the logarithmic depth characterization of NC^1 collapses down to NC^0; more generally (Theorem 1), any class described by sublinear depth circuits, such as AC^1 or NC, collapses down to its constant-depth level;
- by contrast, adding to "$i < j$" the numerical predicate "$i = 2j$", hence a predicate weaker than + or ×, restores ALOGTIME (Theorem 6);
- $\mathrm{AC}^0(A_5) \subset \mathrm{ACC}^0(A_5) \subset$ REGULAR [BIS90,BL06], thus $\mathrm{AC}^0(A_5) \not\supseteq \mathrm{TC}^0$;
- ALOGTIME = $\mathrm{AC}^0(\mathbb{D}_+)$ [Bus87], thus $\mathrm{AC}^0(\mathbb{D}_+) \supseteq \mathrm{TC}^0$;
- $\mathrm{AC}^0(\mathbb{D}_+)_{LIN} \supseteq \mathrm{TC}^0{}_{LIN}$ (Theorem 3);
- $\mathrm{AC}^0(\mathbb{D}_+)_{LIN}$ can neither express the × nor the bit numerical predicate; since the presence of the bit predicate is a major hurdle in lower bound proofs, this suggests attempting to separate $\mathrm{AC}^0(\mathbb{D}_+)_{LIN}$ from $\mathrm{TC}^0{}_{LIN}$ as the next target towards understanding the relationship between TC^0 and NC^1;
- polynomial size Boolean circuits capture P (Theorem 5).

FO[<]-uniformity is defined here as in [BL06] using *unary* shuffled encoding. For purposes of comparison, we also define a FO[+1]-uniform$_{\mathrm{bin}}$ world,

where "$<$" is replaced with the weaker "$+1$" and circuit parameters are expressed in binary notation (following [BCGR92]). In that world, NC^1 does contain ALOGTIME (Theorem 6) and polynomial size circuits still capture P.

2 Preliminaries and definitions

We assume familiarity with circuits and only recall some fundamental definitions. For $i \geq 0$, the class NC^i is the set of languages recognized by circuit families of depth $O((\log n)^i)$ and polynomial size built from bounded fan-in AND, OR and NOT gates. The class AC^i is obtained instead when the AND and OR gates have unbounded fan-in and ACC^i is further obtained when unbounded fan-in MOD_q gates are also permitted. The class TC^i is the set of languages recognized by circuit families of depth $O((\log n)^i)$ and polynomial size built from unbounded fan-in MAJORITY gates. We write $\mathrm{AC}^0(G)$ for the class AC^0 in which the circuits are additionally equipped with unbounded fan-in gates of type G.

Definition 1 (Direct connection language of a circuit family). *Let C_n be a circuit with n inputs and size $\leq n^c$. We label the gates by c-tuples of numbers from 1 to n. Then the direct connection language of C_n is*

$$\begin{aligned} L_{C_n} = \{\langle t, \boldsymbol{a}, \boldsymbol{b}, n\rangle \mid & \textit{the gate } g \textit{ labeled by } \boldsymbol{a} \textit{ has type } t \textit{ and, either,} \\ & t = \mathsf{input} \textit{ and } g \textit{ queries the input bit } \boldsymbol{b}_1 \in \{1,\dots,n\}, \textit{ or} \\ & t \in \{\mathit{true}, \mathit{false}\} \textit{ and } g \textit{ is assigned the value } t, \textit{ or} \\ & t \in \{\neg, \wedge, \vee, G\} \textit{ and the gate labeled } \boldsymbol{b} \textit{ is a predecessor of } g\}. \end{aligned}$$

The direct connection language of a sequence of circuits $C = (C_1, \dots)$ is $L_C = \bigcup_n L_{C_n}$. The predecessors of a gate are fed into the gate in ascending order of their numbers. The output gate is always labeled by $(1, \dots, 1)$.

The language L_{C_n} may describe gates having no path to the output, or unreachable from the inputs, or both. Such gates do contribute to the size of C_n. The depth of C_n is defined as the longest path in the graph of the circuit, regardless of whether this path connects an input gate to the output gate. We will use numbers to encode the type of a gate in t.

We have yet to fix an encoding that will turn the above direct connection language into a set of words over a fixed alphabet. The *unary n-encoding* of a number $1 \leq i \leq n$ is defined as the word $a^i b^{n-i}$ over $\Sigma = \{a, b\}$.

Given a sequence of words $w_1, \dots, w_c$ of a common length n over a common alphabet Σ, the *shuffle* of this sequence is the unique word u of length n over Σ^c, defined by setting the i-th letter of u to $(\sigma_1, \dots, \sigma_c)$ iff σ_j is the i-th letter of w_j for $1 \leq j \leq c$.

Definition 2 (Unary shuffled encoding [BL06]). *The* unary shuffled n-encoding *of a sequence $\alpha_1, \dots, \alpha_k$ of numbers between 1 and n is the shuffle of the sequence of unary n-encodings of $\alpha_1, \dots, \alpha_k$. When n is understood, this shuffle is denoted $\langle \alpha_1, \dots, \alpha_k \rangle$ (and is a word of length n over $\{a, b\}^k$).*

Definition 3 (FO[X]-uniform circuit family). *For* X *a set of numerical predicates, i.e., a set of relations of various arities over* $\mathbb{N}$*, we say that a circuit family* $(C_n)_{n\geq 1}$ *is* FO[X]-uniform *if the following language is expressible in* FO[X]*:*

$$\bigcup_{n\geq 1} \{w \in \{a,b\}^* : \ w = \textit{unary shuffled } n\textit{-encoding of some } \langle t, \boldsymbol{a}, \boldsymbol{b}, n\rangle \in L_{C_n}\}.$$

Remark 1. By [BL06], FO[$<,+,\mathsf{x}$]-uniform $\mathrm{AC}^0(G)$, ACC^0 and TC^0 respectively equal DLOGTIME-uniform $\mathrm{AC}^0(G)$, ACC^0 and TC^0 as defined in [BIS90]. In particular, FO[$<,+,\mathsf{x}$]-uniform $\mathrm{AC}^0(A_5)$ equals what is commonly referred to as DLOGTIME-uniform NC^1 and thus ALOGTIME [BIS90]. To clear possible confusion, recall that to obtain DLOGTIME-uniform NC^1 from a DLOGTIME criterion applied to logarithmic depth circuits, the *extended* connection language is used [Ruz81]. It is known that U_D-$\mathrm{NC}^1 \supseteq$ ALOGTIME, where U_D-NC^1 is defined by DLOGTIME recognition of L_C for logarithmic depth circuit families C, but equality is still open (see [Vol99, P. 162]).

When dealing with the Dyck language $\mathbb{D}_1$ over one pair of parentheses we will use the letters $\{a,b\}$ instead of $\{(,)\}$ to improve readability.

We define now a formal language version of the formula value problem. We encode complete binary trees, i.e.: trees where each inner node has exactly two predecessors, by a traversal from left to right. A left edge is labeled by a and a right edge is labeled by b. This gives a one-to-one correspondence between complete binary trees and the Dyck language $\mathbb{D}_1 \subset \{a,b\}^*$.

We decided in favor of labelling the edges and against the more usual labelling of the vertices, which would lead to the well known representation of complete binary trees by the Lukasiewicz language.

A tree labelled by Boolean functions and constants evaluates either to *true* or to *false*. In this way the Dyck set $\mathbb{D}$ is divided into the two disjoint subsets $\mathbb{D} = \mathbb{D}_+ \cup \mathbb{D}_-$ where $\mathbb{D}_+$ consists in those elements of $\mathbb{D}$ which represent a tree (labelled with the NAND-function and the constant *true*) which evaluates to *true* and $\mathbb{D}_-$ contains those which evaluate to *false*. Thus $\mathbb{D}_+$ is a special formulation of the Boolean formula value problem which makes $\mathbb{D}_+$ NC^1-complete.

Definition 4 ($\mathbb{D}_+$ language). *Any word in the Dyck language* $\mathbb{D}_1$ *corresponds to a tree as defined above. If we let every leaf of the tree be a true node and every inner node be a* NAND*, the tree is a formula evaluating either to true or false. We let* $\mathbb{D}_+ \subset \{a,b\}^*$ *be the set of words that are in the Dyck language and whose corresponding formula evaluates to true.*

We now define two gate types based on languages that are complete for NC^1, even in the case of DLOGTIME-uniformity.

Definition 5 (A_5 Gate). *Let* A_5 *be the alternating group over 5 elements and* $g_0, g_1 \in A_5$ *be two 5 cycles that span the whole group. An* A_5 *gate with* k *Boolean inputs* $x_1, \ldots, x_k$ *evaluates to* 1 *if* $g_{x_1} \ldots g_{x_k} = 1$*, otherwise to* 0*.*

Recall that we write the Dyck language over the alphabet $\{a,b\}$.

Definition 6 ($\mathbb{D}_+$ Gate). *A $\mathbb{D}_+$ gate with k Boolean inputs $x_1, \ldots, x_k$ evaluates to 1 if replacing every 0 by a and every 1 by b in the word $x_1 \cdots x_k$ yields a word in $\mathbb{D}_+$.*

We assume some familiarity with first order descriptive complexity. We follow Straubing and use his semantics based on $\mathcal{V}$-structures [Str94, P. 14]. We write FO to denote first order logic and write the set of allowed numerical predicates in brackets. We will use the binary predicates bit, $<$, $+1$, and $\mathsf{x2}$. Since the value of a numerical predicate depends only on the position of the variables and the word length, we will freely switch between variables and natural numbers denoting their positions. To make this transition clearer we write "$x = i$" for a variable x and a natural number i when x points to the i-th position. The predicate $\mathsf{bit}(i, j)$ is true if the i-th bit of the binary representation of j is a 1. The successor predicate $+1(i, j)$ is true $i = j + 1$. The double predicate $\mathsf{x2}(i, j)$ is true if $i = 2j$. Recall that FO[$<$] only describes regular languages (it in fact captures the aperiodic regular languages [MP71]). The class FO[$+1$] is a proper subclass of FO[$<$] that in fact captures the threshold testable languages [Tho78].

The class FO can also be extended by adding additional quantifiers. We write FO + MAJ, FO + A_5, and FO + M, resp., for the class FO which is also equipped with the majority quantifier, with an A_5-quantifier, and with arbitrary finite monoid quantifiers, respcectively.

3 FO[$<$]-uniform logarithmic depth circuits

Our first theorem shows that FO[$<$]-uniform NC^1 is restrictive as the class collapses down to NC^0. If we add a simple numerical predicate like $\mathsf{x2}$ we obtain full DLOGTIME-uniform NC^1.

To prove our first theorem we show that FO[$<$] cannot express an edge relation such that the resulting graph has paths of length $\Theta(\log n)$. We first explain why we need only to consider the expressiveness of FO[$<$] in terms of numerical predicates.

The expressive power of FO[$<$] is well understood. One important restriction of FO[$<$] is that a fixed formula can only count up to a constant. Beyond this constant it can only check the relative ordering. It is for example known that for quantifier depth d a formula cannot distinguish between the words a^{2^d} and a^{2^d+1}.

We use the following observation from [Str94][p. 79].

Lemma 1. *Let ϕ be a formula of quantifier depth d over the alphabet Σ, let $u, w \in \Sigma^*$, and let $v \in \Sigma$. Then $uv^{2^d-1}w \models \phi$ iff $uv^{2^d}w \models \phi$.*

This allows us to prove the following theorem:

Theorem 1. *Let X be any circuit class that limits the depth $d(n)$ to be sublinear, i.e. $d(n) \in o(n)$, and X' be the same circuit class with the restriction that the depth is constant.* FO[$<$]-uniform X =FO[$<$]-uniform X'.

Proof. Let $(C_n)_n$ be a family of circuits in FO[<]-uniform X of depth $l(n)$, the labels of $(C_n)_n$ are k-tuples of numbers, and the connection language is recognized by a FO[<] formula ϕ of quantifier depth d.

We prove the theorem by showing that one cannot define a circuit of depth sublinear but not constant, i.e. $l(n)$ is sublinear. There is a value N_0 such that for all $n > N_0$ we have $n \geq (l(n) \cdot k + 1) \cdot (2^d + 1)$. By induction we show for all $n \geq N_0$ that $l(n) \leq l(N_0)$. For the basis $l(N_0) \leq l(N_0)$ there is nothing to prove.

Choose any $n > N_0$. Let $(G_1, \ldots, G_{l(n)})$ be a sequence of gates in C_n, such that for $1 < i \leq l(n)$ the gate G_i is an input gate of G_{i-1}. So the gates form a "path" in the circuit C_n. We let L_i be the gate label of G_i, which is a shuffled unary encoding of k numbers. Let w be the shuffled encoding of all L_i for $i = 1, \ldots, l(n)$, and let Σ be the corresponding alphabet. So w is a word which is the unary shuffled encoding of $l(n) \cdot k$ numbers. We have a formula ψ for our family of circuits that checks if a word over Σ^* is the shuffled encoding of a path of length $l(n)$ in any one of the circuits $C_{n'}$, by using a $(l(n) - 1)$-ary conjunction of the connection formula ϕ. The depth of ψ is d.

Let $\alpha_1, \ldots, \alpha_{l(n)\cdot k}$ be the ordered set of the $l(n) \cdot k$ numbers. The shuffled encoding of these numbers will have the same letter in each of these intervals, i.e. for $i < i'$ if there does not exist a j such that $i \leq \alpha_j \leq i'$, then the i-th letter equals i'-th letter, i.e. $w_i = w_{i'}$. Since $n \geq (l(n) \cdot k + 1) \cdot (2^d + 1)$ there exists a factor word in w of the form σ^{2^d} for $\sigma \in \Sigma$, i.e. $w = u\sigma^{2^d}v$ for $u, v \in \Sigma^*$.

So we will apply Lemma 1 to w and ψ, and obtain a word w' of length $n-1$. But this word can be interpreted as the shuffled encoding of the labels of a gates sequence $(G'_1, \ldots, G'_{l(n)})$ in C'_{n-1}. Also by Lemma 1 we know that no formula of depth d can distinguish w and w', hence G'_i is an input gate of G'_{i-1} in C_{n-1}. It follows that $l(n) \leq l(n-1)$ and by the induction hypothesis $l(n) \leq l(N_0)$. □

In the previous theorem if we choose G_1 to be the output gate, we would have that G'_1 is also the output gate, hence not just some irrelavent gates that do not influence the output gate generate the problem, but an actually path form the output gate to an input gate can only have constant length.

Corollary 1. *In the world of* FO[<]*-uniformity,* $\mathrm{NC}^i = \mathrm{NC}^0$*,* $\mathrm{AC}^i = \mathrm{AC}^0$ *and* $\mathrm{TC}^i = \mathrm{TC}^0$ *hold for every* $i \geq 1$*.*

In view of the inability for a FO[<]-uniform class to recognize polylogarithmic depth properly, one might consider adding a unary predicate such as $\log_n(x)$ defined to be true if $\log(n) = x$.

Using the same idea as above one can show that the output gate labeled by $(l_1, \ldots, l_k)$ can only access input positions in a polylogarithmic range around $(l_1, \ldots, l_k)$, since we have a path from the gate labeled $(l_1, \ldots, l_k)$ to the input gate at position p_i of polylogarithmic length. Hence the same proof shows that one cannot obtain FO[<, log]-uniform NC^i circuits that are able to recognize the language 1^*.

While FO[<] cannot describe circuits of logarithmic length, adding a simple binary predicate like x2, which is much weaker than for example +, already allows to describe DLOGTIME uniform circuits (see Theorem 6).

We obtain a dichotomy for our uniformity definitions and circuits of logarithmic depth. The classes result either in subclasses of NC^0 or contain DLOGTIME-uniform NC^i. The fact that even low uniform circuit classes capture DLOGTIME-uniform NC^1 stems from its equivalence to ALOGTIME. Turing machines are uniform and operate only locally. As noted in [BL06] this also allows tightly uniform circuit characterizations for polynomial time. In Section 5 we discuss how this can be extended to a general framework.

4 FO[<]-uniform constant-depth NC^1 characterizations

In this section we will not consider log depth circuits but constant depth circuits equipped with gates that compute NC^1 complete languages. We consider $\mathbb{D}_+$ and the word problem over A_5 as complete problems.

Extending AC^0 by A_5 gates gives a proper subclass of the regular languages.

Theorem 2. *Let M be any subset of monoids, then* FO[<]-uniform $\mathrm{AC}^0(M) \subseteq$ REGULAR*, where equality only occurs if every finite monoid can be simulated by (i.e., is a homomorphic image of a submonoid of) a monoid from M.*

Proof. This follows by translating the circuit into a FO + M[<] formula and applying Theorem 11.6 from [BIS90]. Here FO + M stands for first order logic equipped with monoid quantifiers as defined in [BIS90]. □

The same argument does not only show that FO[<]-uniform $\mathrm{AC}^0(M)$ circuits are contained in the regular languages but that even FO[<]-uniform $\mathrm{ACC}^0(M)$ recognize only a subset of the regular languages. The only way to obtain something containing an acceptably large subclass of TC^0 seems to be the bit predicate, but this immediately yields uniform NC^1.

So instead of choosing gates based on finite non-solvable groups, we choose a gate type that corresponds to the Boolean formula value problem. It is known [BL06] that that FO[<]-uniform TC^0 is separated from FO[<]-uniform TC^0 with linear fan-in. While the former can simulate the bit predicate and is hence equal to DLOGTIME-uniform TC^0 the latter equals FO + MAJ[<] and cannot simulate the bit predicate.

Yet we can show that inclusion under DLOGTIME uniformity remains valid under FO[<] uniformity:

Theorem 3. FO[<]-uniform $\mathrm{AC}^0(\mathbb{D}_+)_{LIN} \supseteq$ FO[<]-uniform TC^0_{LIN}.

We mention that MOD_q gates can be simulated in FO[<]-uniform TC^0_{LIN}. Together with the previous theorem it follows that FO[<]-uniform $\mathrm{AC}^0(\mathbb{D}_+) \supseteq$ FO[<]-uniform $\mathrm{TC}^0 \supseteq$ FO[<]-uniform ACC^0. These inclusions remain valid for the corresponding circuit classes with linear fan-in. The following theorem shows that $\mathrm{AC}^0(\mathbb{D}_+)_{LIN}$ is strictly weaker than NC^1, but it is still not clear whether it is contained in (even non uniform) TC^0.

The following theorem is a consequence of Theorem 4.16 in [LMSV01], where it is shown that only semilinear predicates can be computed by groupoidal qunatifiers using only the order predicate.

Theorem 4. *The predicate* x *(and hence* bit*) cannot be expressed in* FO[<]-uniform $\mathrm{AC}^0(\mathbb{D}_+)_{LIN}$.

Theorem 3 and Theorem 4 highlight the important difference between $\mathrm{AC}^0(\mathbb{D}_+)$ and $\mathrm{AC}^0(\mathbb{D}_+)_{LIN}$; indeed the former is able to simulate bit and thus equals NC^1. Note that superlinear fan-in of gates corresponds to quantifiers over tuples of variables in the logic world.[4]

Summarizing we are able to say: in contrast to A_5 gates, using $\mathbb{D}_+$ gates yields a class that is weaker than DLOGTIME uniform NC^1 but contains uniform subclasses of TC^0_{LIN}. Hence, FO[<]-uniform $\mathrm{AC}^0(\mathbb{D}_+)_{LIN}$ is a candidate subclass of NC^1 worthy of attempts to separate it from TC^0.

5 A guide to minimal Uniformity

In Section 3 we noticed that log depth circuits with very tight uniformity can already simulate circuits which are defined by a more powerful uniformity. Ruzzo already showed that for larger classes in NC different uniformity notions coincide. This phenomenon is much more general and we explore it in this section. We start by showing that polynomial time admits very uniform circuits over the standard Boolean gates. Recall that a polynomial size circuit family is P-uniform if L_{C_n} can be listed by a Turing machine in time polynomial in n.

Theorem 5. P*-uniform and* FO[+1]-uniform *polynomial size circuits each capture the class* P.

An obvious extension to Theorem 5 is to consider more general complexity classes of Turing machines. Consider a complexity class $\mathcal{M}$ defined by time and space bounds. When simulating a TM in $\mathcal{M}$ as a circuit then time translates to depth and space to width. If one equips FO[<] with unary predicates that allow to check the depth and width bounds, it is possible to perform the construction without much overhead. Therefore, a (deterministic) Turing machine can be simulated by very uniform circuits. If now conversely such a circuit can be evaluated by a TM in $\mathcal{M}$ then any $\mathcal{M}$-uniform $\mathcal{C}$ circuit can be converted to a FO[$<, \mathsf{p_B}$]-uniform $\mathcal{C}$ circuit. Here $\mathsf{p_B}$ stands for a set of unary predicates that allow to check the bounds on the circuit. This construction requires some minimal closure properties for the function for the functions giving the time and space bounds on the machine. The idea is to take the machine that evaluates a circuit in $\mathcal{M}$-uniform $\mathcal{C}$ and to construct the FO[$<, \mathsf{p_B}$]-uniform circuit for this machine. An example is polynomial time where P-uniform polynomial size circuits equal FO[<]-uniform polynomial size circuits. (Here, $\mathsf{p_B}$ can be directly expressed within FO[<].)

Similar observations hold for alternating Turing machines as exhibited in [Ruz81]: for $k > 1$ the different notions of uniformity define identical circuit

[4] For the A_5 quantifier we do not have to distinguish between quantifiers over one variable and quantifiers over a tuple of variables.

families including LOGSPACE-uniform families. Then, it is easy to see how to extend the construction of Theorem 6 to circuits of depth $\log^k$.

We believe that the requirement for $\mathcal{C}$ to be contained in some Turing class can be omitted. Let M be the machine that would decide the uniformity language. The idea is the following: Let a be a gate and b be a possible candidate to be a predecessor. Instead of letting the uniformity language to decide whether there is a wire from b into a, we build a circuit, that will evaluate M and then either feed in b or not. (Note this requires to be able to feed in a neutral input.) This is done for all possible gates b for a. We call this construction a "switch gate". So we need to be able to simulate M in the circuit class $\mathcal{C}$. This idea is can be already found in [Ruz81].

This is an explanation why log-depth circuits seem to be always at least DLOGTIME-uniform as exhibited in Section 3.

6 Binary Encoding

In this section only, we replace *unary* by *binary* in our shuffled encodings and consider the effect on the uniformity notions that arise. The *binary n-encoding* of a number $0 \leq i < 2^n$ is defined as $w_1 \cdots w_n \in \{0,1\}^n$ such that $i = \sum_j 2^{j-1} w_j$. Note that the resulting encoding differs from the encoding in [BIS90] not only in the shuffling, but in the fact that here y is also given in binary:

Definition 7 (Binary shuffled encoding). *The* binary shuffled n-encoding *of a sequence $\alpha_1, \ldots, \alpha_k$ of numbers between 1 and n is the shuffle of the sequence of binary n-encodings of $\alpha_1, \ldots, \alpha_k$. When n is understood, this shuffle is denoted $\langle \alpha_1, \ldots, \alpha_k \rangle_b$ (and is a word of length $\lceil \log(n+1) \rceil$).*

We say that a circuit family $(C_n)_{n \geq 1}$ is FO[X]-uniform$_{\text{bin}}$ if the language formed by the union, over all n, of the language of binary shuffled n-encodings of the tuples in L_{C_n} can be described by an FO[X] formula.

Note that Theorem 5 also holds for binary encoding. It is easy to see that we can switch from unary to binary shuffled encoding. The tests for the fixed cases are clearly in FO[+1], for the other tests the formula must compute ± 1 on binary numbers, but this can be done in FO[+1].

For binary encoding we get a similar result for NC^1:

Theorem 6. *The classes* FO[$<$, x2]-uniform NC^1 *and* FO[+1]-uniform$_{\text{bin}}$ NC^1 *each contain* DLOGTIME-*uniform* NC^1.

The proof of Theorem 6 builds the circuit for an ALOGTIME machine from its configuration tree. Both theorems exploit the locality of a computational step of a Turing machine.

7 Discussion

Our focus in this paper was the following reasoning: if circuit classes such as $\mathrm{ACC}^0 \subseteq \mathrm{TC}^0 \subseteq \mathrm{NC}^1$ resist separation attempts, then why not tighten their uniformity and draw intuition from comparing the restricted classes?

Our results slightly extend [BL06]. They mostly concern FO[<]-uniformity, a notion provably tighter than the robust DLOGTIME-uniformity commonly accepted as the natural choice for defining fundamental circuit subclasses of P.

Our first conclusion is that *no* intuition comes out of applying to logarithmic depth circuits the tight uniformities considered here, because these circuits either retain their full power or they cannot exploit their depth beyond a constant.

Another observation is that a language (such as A_5 here) complete for a large class (such as ALOGTIME here) under the reduction relevant to a robust uniformity (such as DLOGTIME-uniformity here) may no longer be complete for the same class defined under a tighter uniformity. Indeed, here we noted that in the FO[<]-uniform world, the AC^0-closure of A_5 no longer contains TC^0.

By contrast with A_5, we observed that the $\mathbb{D}_+$ variant of the NC^1-complete formula value problem behaves differently. We could show that in the FO[<]-uniform world, the AC^0 closure of $\mathbb{D}_+$ equals ALOGTIME and thus contains TC^0. This suggested considering $\mathrm{AC}^0(\mathbb{D}_+)_{LIN} \supseteq \mathrm{TC}^0_{LIN}$ because imposing a linear bound on the fanin of unbounded fanin circuits nicely fits into the first-order characterizations of the language classes captured. In the FO[<]-uniform world (a world free of the "tyranny" of the bit predicate), can $\mathrm{AC}^0(\mathbb{D}_+)_{LIN} \neq \mathrm{TC}^0_{LIN}$ be proved? Such a separation would further amount to distinguishing the power of MAJ from the power of $\mathbb{D}_+$. We have been unable to answer that question though it might be within reach.

In [MTV10], the regularity conjecture (see footnote in Section 1) was generalized and named the "uniform duality property". Simplifying somewhat, the property holds for a class C if any language in C expressed in ExtFO[arb] can be reexpressed in $\mathrm{ExtFO}[<, C^N]$, where C^N is a set of numerical predicates defined from C. A marginal link with the uniform duality property can be found in our Theorem 5. Let ExtFO locally here mean allowing a Lindström quantifier for a P-complete problem under AC^0-reducibility. Then

$$\mathrm{ExtFO}[\mathrm{P}^N] \cap C = \mathrm{ExtFO}[+1] \cap C \subseteq \mathrm{ExtFO}[<, +] \cap C \subseteq \mathrm{ExtFO}[\mathrm{CFL}^N] \cap C$$

where C is the class of context-free languages, the "=" uses Theorem 5 and the rightmost "$\subseteq$" follows by [MTV10]. This is a weaker instance of the duality property in which we replace predicates from P^N (rather than from arb) in order to reexpress any context-free language. Can stronger instances of the duality be proven, by extending the present work or by bringing in the extensions to [RS07] recently announced in [KS12]?

The fact that we can find strictly uniform circuit classes for ALOGTIME is based on the exploitation of uniformity and locality of the steps of a Turing machine. This also allows FO[<]-uniform characterizations of polynomial time as observed in [BL06]. We think that this could be extended to circuit classes that have characterizations in terms of Turing machines, or to circuit classes whose uniformity languages are defined by Turing machines as long as the circuit class is at least as powerful as the uniformity language.

Perhaps one other research avenue would be to consider direct connection language encodings that are intermediate between the unary shuffled encoding

and the binary shuffled encodings studied here. Or could a meaningful encoding-free notion of uniformity be developed? Would there be a use for such a notion?

Acknowledgement. We thank anonymous referees for useful comments on earlier versions of the present paper.

References

[All89] Eric Allender. P-uniform circuit complexity. *J. ACM*, 36(4):912–928, 1989.

[Bar89] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *J. Comput. Syst. Sci.*, 38(1):150–164, 1989.

[BCGR92] Samuel R. Buss, S. Cook, A. Gupta, and V. Ramachandran. An optimal parallel algorithm for formula evaluation. *SIAM J. Comput.*, 21(4):755–780, 1992.

[BIS90] David A. Mix Barrington, Neil Immerman, and Howard Straubing. On Uniformity within NC^1. *J. Comput. Syst. Sci.*, 41(3):274–306, 1990.

[BL06] Christoph Behle and Klaus-Jörn Lange. FO[<]-Uniformity. In *IEEE Conference on Computational Complexity*, pages 183–189, 2006.

[Bor77] Allan Borodin. On relating time and space to size and depth. *SIAM J. Comput.*, 6(4):733–744, 1977.

[Bus87] Samuel R. Buss. The boolean formula value problem is in alogtime. In *STOC*, pages 123–131. ACM, 1987.

[Coo79] Stephen A. Cook. Deterministic CFL's are accepted simultaneously in polynomial time and log squared space. In *STOC*, pages 338–345. ACM, 1979.

[Gol78] Leslie M. Goldschlager. A unified approach to models of synchronous parallel machines. In *STOC*, pages 89–94, 1978.

[Imm87] Neil Immerman. Languages that capture complexity classes. *SIAM J. Comput.*, 16(4):760–778, 1987.

[KS12] Andreas Krebs and A V Sreejith. Non-definability of languages by generalized first-order formulas over $(\mathbb{N},+)$. In *LICS*, page to appear, 2012.

[Lad75] Richard E. Ladner. The circuit value problem is log space complete for p. *SIGACT News*, 7:18–20, January 1975.

[LMSV01] Clemens Lautemann, Pierre McKenzie, Thomas Schwentick, and Heribert Vollmer. The descriptive complexity approach to LOGCFL. *J. Comput. Syst. Sci.*, 62(4):629–652, 2001.

[MP71] Robert McNaughton and Seymour Papert. *Counter-free automata. With an appendix by William Henneman.* Research Monograph No.65. Cambridge, Massachusetts, and London, England: The M. I. T. Press. XIX, 163 p., 1971.

[MTV10] Pierre McKenzie, Michael Thomas, and Heribert Vollmer. Extensional uniformity for boolean circuits. *SIAM J. Comput.*, 39(7):3186–3206, 2010.

[RS07] Amitabha Roy and Howard Straubing. Definability of languages by generalized first-order formulas over N^+. *SIAM J. Comput.*, 37(2):502–521, 2007.

[Ruz81] Walter L. Ruzzo. On uniform circuit complexity. *J. Comput. Syst. Sci.*, 22(3):365–383, 1981.

[Str94] Howard Straubing. *Finite Automata, Formal Logic, and Circuit Complexity.* Birkhäuser, Boston, 1994.

[Tho78] Wolfgang Thomas. The theory of successor with an extra predicate. *Math. Annalen*, 237:121–132, 1978.

[Vol99] Heribert Vollmer. *Introduction to Circuit Complexity.* Springer, 1999.

APPENDIX: proofs of Lemma 1, Theorem 3, Theorem 5 and Theorem 6

Lemma 1. Let ϕ be a formula of quantifier depth d over the alphabet Σ, let $u, w \in \Sigma^*$, and let $v \in \Sigma$. Then $uv^{2^d-1}w \models \phi$ iff $uv^{2^d}w \models \phi$.

Proof. We show that Duplicator has a winning strategy in a d-round Ehrenfeucht-Fraïssé game on $(uv^{2^d-1}w, uv^{2^d}w)$.

The strategy for Duplicator is as follows. If Spoiler puts a pebble on u or w then Duplicator answers at the matching position of the other word. If Spoiler puts a pebble on v, we adapt the strategy used in the proof of Theorem IV.2.1 in [Str94]. There it is shown that Duplicator has a winning strategy A for (a^{2^d-1}, a^{2^d}). So if Spoiler puts a pebble on the i-th copy of v within v^{2^d-1} or v^{2^d}, then Duplicator interprets i as a position within a^{2^d-1} or a^{2^d}. In the strategy A Duplicator would place a pebble at position j on a^{2^d} or a^{2^d-1}, respectively, so Duplicator places his pebble on the j-th copy of v^{2^d-1} or v^{2^d}, respectively. □

Theorem 3. FO[<]-uniform $\mathrm{AC}^0(\mathbb{D}_+)_{LIN} \supseteq$ FO[<]-uniform TC^0_{LIN}.

Proof. We recall the definitions of the following languages: $MAJORITY = \{w \in \{0,1\}^* \mid \#_1(w) > \#_0(w)\}$, $EQUALITY = \{w \in \{a,b\}^* \mid \#_a(w) = \#_b(w)\}$, and $\mathbb{D}_1 = \{w \in EQUALITY \mid \forall u, v \text{ with } uv = w : \#_a(u) \geq \#_b(u)\}$. We need to show that we can simulate a MAJ gate by a $\mathbb{D}_+$ gate in the given uniformity. In order to do so we show how to reduce the majority language to the one sided Dyck language over a single pair of parentheses $\mathbb{D}_1$ and then reduce this to $\mathbb{D}_+$. Let us begin by reducing the language EQUALITY, i.e. $\{w \in \{0,1\}^* \mid \#_1(w) = \#_0(w)\}$, to $\mathbb{D}_1$. We define two morphisms $\pi_a, \pi_b : \{0,1\}^* \to \{a,b\}^*$ by letting $\pi_a(1) = aa, \pi_a(0) = ab$, $\pi_b(1) = ab, \pi_b(0) = bb$. The mapping $w \mapsto \pi_a(w)\pi_b(w)$ is a reduction from EQUALITY to $\mathbb{D}_1$.

To obtain a reduction from MAJORITY to $\mathbb{D}_1$ observe that

$$\#_a(\pi_a(w)\pi_b(w)) - \#_b(\pi_a(w)\pi_b(w)) \geq 0 \Leftrightarrow \#_1(w) \geq \#_0(w)$$

(and $\pi_a(w)\pi_b(w)$ is a prefix of a Dyck word). Hence, if the number of 1's in w is less than half there is no suffix z such that $\pi_a(w)\pi_b(w)z \in \mathbb{D}_1$. We define two more morphisms that will build the Dyck inverse for $\pi_a(w)\pi_b(w)$: Let $\bar{\pi}_a(1) = bb, \bar{\pi}_a(0) = ab$, $\bar{\pi}_b(1) = ab, \bar{\pi}_b(0) = aa$. Now we have a reduction from MAJORITY to $\mathbb{D}_1$ by the mapping $w \mapsto \pi_a(w)\pi_b(w)\bar{\pi}_b(w)\bar{\pi}_a(w)$.

We sum up the idea: we open $\#_1(w)$ many parentheses, then close $\#_0(w)$ many parentheses, then open $\#_0(w)$ many parentheses, and finally close $\#_1(w)$ many parentheses. This expression is valid if no more parentheses in the middle are closed than are opened before, i.e. $\#_1(w) \geq \#_0(w)$. So actually this accepts words with an equal number of 1's and 0's. This can be fixed in the logic by excluding this case through testing on equality as shown above. We have that $w \in L_{MAJ} \Leftrightarrow \pi_a(w)\pi_b(w)\bar{\pi}_b(w)\bar{\pi}_a(w) \in \mathbb{D}_1$.

We reduce $\mathbb{D}_1$ to $\mathbb{D}_+$ by mapping w to $aabbw$. The reduction creates a tree, with a node, where the left subtree defined by $aabb$ evaluates to false and the

right subtree is defined by w. Therefore, the whole tree always evaluates to true iff w is a correct word in $\mathbb{D}_1$.

To simulate the whole reduction in FO[<], we need a reduction that has very limited computational power. We will generate a reduction that reduces a word of length n to a word of length $10n$. The position will be as tuples where (x, y) stands for position $x + ny$ with $x \in \{1, \ldots, n\}$ and $y \in \{1, \ldots, 10\}$. The position $x + ny$ of the reduced word will depend only on x and also either the input at position x, or will be a constant for all $x > 4$. We use the following observations:

First, the reduction from L_{MAJ} to $\mathbb{D}_1$ remains valid if we split the morphism π_a into two morphisms $\pi_a^1(a) = a, \pi_a^1(b) = a$, $\pi_a^2(a) = a, \pi_a^2(b) = b$. We split the other three morphisms in the same fashion and observe that still holds

$$w \in L_{MAJ} \Leftrightarrow \pi_a^1(w)\pi_a^2(w)\pi_b^1(w)\pi_b^2(w)\bar{\pi}_b^1(w)\bar{\pi}_b^2(w)\bar{\pi}_a^1(w)\bar{\pi}_a^2(w) \in \mathbb{D}_1.$$

Second, given a word w of length $n \geq 4$ we can reduce $\mathbb{D}_1$ to $\mathbb{D}_+$ by mapping w to $aabba^{n-4}aabbb^{n-4}w$. The word $a^{n-4}aabbb^{n-4}$ is in $\mathbb{D}_1$ so behaves neutral in the morphism.

Summarizing, we map a word of length n to a word of length $10n$ to obtain a reduction from L_{MAJ} to $\mathbb{D}_+$. Where

$$w \mapsto aabba^{n-4}aabbb^{n-4}\pi_a^1(w)\pi_a^2(w)\pi_b^1(w)\pi_b^2(w)\bar{\pi}_a^1(w)\bar{\pi}_b^2(w)\bar{\pi}_a^1(w)\bar{\pi}_a^2(w)$$

This construction can be carried out in FO[<]-uniformity by using a variable which is bound to the positions $\{1, \ldots, 10\}$. □

Theorem 5. P-uniform and FO[+1]-uniform polynomial size circuits each capture the class P.

Proof. It is known that the first circuit class equals P. The latter class is also easily seen to be in P, it remains to show that P is contained in this class. To see how P can be translated into FO[+1]-uniform circuits recall how a TM M of running time n^k is simulated by circuits ([Lad75]). The basic layout of the circuit is a grid of $2n^k \times n^k$ subcircuits. A subcircuit at position (p, t), will compute the state of the tape at position p at time t. Since such a subcircuit has a constant size, we will enumerate the gates of the circuit by tuples of size $2k + k + c$, where c depends on the size of the subcircuit.

Such a subcircuit encodes the state of a tape cell at position p and time t, i.e. the symbol written on it and eventually the state of the TM, if the head is over the tape cell, by a constant number of, say c, output gates. Its inputs are connected to $3c$ input gates, namely the output gates of the subcircuits at positions $(p-1, t-1)$, $(p, t-1)$, and $(p+1, t)$.

To see why the resulting circuit is FO[+1]-uniform, we observe what has to be checked. To see if two gates, each encoded by a $3k + c$ tuple (p, t, g) are connected, the formula has to check two cases: (Let the gates be (p_1, t_1, g_1) and (p_2, t_2, g_2))

1. $p_1 = p_2, t_1 = t_2$, the wiring is within a subcircuit and hence a finite function ranging over the possible constant values of (g_1, g_2). All such functions are in FO[+1].

2. $t_1 = t_2 - 1$. In this case p_1 is either equal to p_2, $p_2 + 1$, or $p_2 - 1$. This is checkable in FO[+1]. Furthermore, the formula has to check, if g_1 and g_2 have the correct number, but the same argument as above applies.

For the cases of $t = 1$ and $t = n^k$ we have to add special cases, one layer translating the input into tape cells, the other checking with a big OR gate, if one tape cell at the last time step has a head in an accepting state.

Each of these cases, as well as the cases at the border of the grid can be handled similar to above by a FO[+1]-formula. □

Theorem 6. The classes FO[$<, \mathsf{x2}$]-uniform NC^1 and FO[+1]-uniform$_{\mathrm{bin}}$ NC^1 each contain DLOGTIME-uniform NC^1.

Proof. We let M be a ALOGTIME machine that recognizes the language of the DLOGTIME-uniform NC^1 circuit. Our model is similar to the one of Ruzzo [Ruz81], with the exception that we do not use an extra index tape. The index tape of the ALOGTIME machine are the first $\log n$ bits to the right of the head of the working tape and the machine queries the input only in the last step by switching into a state s_σ. Further we choose c such that the ALOGTIME machine uses at most $c \log n$ steps in every run.

We will build a FO[$<, \mathsf{x2}$]-uniform NC^1 circuit that recognizes the same language. We label the gates of the circuits by a tuple: $(t_1, \ldots, t_c, s, l, l_1, \ldots, l_c, r, r_1, \ldots, r_c)$. The idea is the following: $(t_1, \ldots, t_c)$ denotes the time step of the machine. We can count from 1 to $x \log n$ by starting from $(t_1, \ldots, t_e)$ and doubling t_1 each time, until $2 \cdot t_1$ would be greater than n, then we continue by doubling t_2 and so on. In the following, we write $\boldsymbol{x}$ to denote the vector $x_1, \ldots, x_c$. A tuple $s, \boldsymbol{l}, \boldsymbol{r}$ will denote the configuration of the machine, where s is the state of the machine and $\boldsymbol{l}$, $\boldsymbol{r}$ are the parts of the working tape left and right of the head. The auxiliary variables l and r will keep track of how many bits of l_1 and r_1 are used. Hence, the type of the gate depends directly on s, the only problem is to test if two gates are predecessors. Given two gates $(\boldsymbol{t}, s, l, \boldsymbol{l}, r, \boldsymbol{r})$, $(\boldsymbol{t'}, s', l', \boldsymbol{l'}, r', \boldsymbol{r'})$ we have to check the following conditions:

Time The formula has to ensure that $\boldsymbol{t'}$ describes one timestep after $\boldsymbol{t}$. We say that an entry t_i is maximal if there is no position z such that $z = \mathsf{x2}(t_i)$. By our encoding the formula has to check: There is an i such that $t_j = t'_j$ for $j < i$ and the t_j are maximal. For i we have $\mathsf{x2}(t_i) = t'_i$ and $t_k = t'_k$ are at the first position for $k > j$.

Tape We assume that the lowest bit of l is the bit under the working head. We can check if that bit is 0 by checking if l_1 is even, i.e. $\exists x x = \mathsf{x2}(l_1)$. We can write a 0 on that position by first shifting l to the right and then to the left, i.e. the formula first determines the largest x such that $\mathsf{x2}(x) \leq n$ and takes then $\mathsf{x2}(x)$. To write 1 we take $\mathsf{x2}(x) + 1$ of said x. The case of r_1 is handled analogously. Before writing a 1, the formula has to make sure that $\mathsf{x2}(x) + 1 \leq n$. If that is not the case, we copy all $l_i \neq 0$ to l_{i+1} and set $l_1 = 0$. This means, we just shift the vector $(l_1, \ldots, l_c)$ to the right.

State and head movement The transition from s to s' can be checked by reading the lowest bit of l_1. The movement of the head to the right is simulated by reading the lowest bit of l_1, shifting l_1 to the right, and writing that bit on r_1. Note, if l_1 was already zero, we have to shift $(l_1, \ldots, l_c)$ to the left, i.e. set l_i to l_{i+1}. We keep track of the bits stored in l_1 and r_1 by the variables l, r. This way we can distinguish between the vectors 1 and 10 for example. In both cases l_1 would equal 1, but l is 1 in the first case and 2 in the second.

Input The input gates are labeled by (σ, i), where i is given in unary. If the ATM makes a transition into state $READ(\sigma)$, we connect to the input gate (σ, l_j), where j is the maximal used block.

Note that the so constructed circuit also connects a lot of configurations that are not reachable by the ALOGTIME machine. Still any path in the circuit has a length of at most $c \log n$ because of t in our labeling. Hence, the circuit fulfills the required depth restrictions.

The same construction works for FO[+1]-uniform$_{\text{bin}}$ NC^1 circuits. We do not need a double predicate here since we only need to check if the binary numbers are shifted by one. □