Visibly Counter Languages and the Structure of NC^1

Michael Hahn, Andreas Krebs, Klaus-Jörn Lange, and Michael Ludwig

WSI - University of Tübingen
Sand 13, 72076 Tübingen, Germany
{hahnm,krebs,lange,ludwigm}@informatik.uni-tuebingen.de

Abstract. We extend the familiar program of understanding circuit complexity in terms of regular languages to visibly counter languages. Like the regular languages, the visibly counter languages are NC^1 - complete. We investigate what visibly counter languages are in certain constant depth circuit complexity classes, which we have initiated in a previous work for AC^0. We present characterizations and decidability results for various circuit classes. In particular, our approach yields a way to understand TC^0, where the regular approach fails.

1 Introduction

The family of the regular languages is well among the best studied objects in theoretical computer science. They arise in many contexts, have many natural characterizations and good closure and decidability properties, and have been used as a tool to understand other objects. The nice behavior of regular language comes at a price, which is a limited expressiveness. The class of context-free languages contains many important non-regular languages, but at the same time is much harder to analyze. In our present line of work, we attempt to generalize results which are known for the regular case to the non-regular case. A very promising way of generalizing regular languages is given by the *visibly pushdown languages*, which were introduced by Mehlhorn [13] in 1980 and popularized by Alur and Madhusudan [2] in 2004. They have been an active field of research ever since. Visibly pushdown automata are pushdown automata with the restriction that the input letter determines the stack operation: Push, pop or no access to the stack. These automata have good closure and decidability properties that are comparable to those of regular languages, and still cover many important context-free languages. The word problem is as hard as for the regular languages, namely NC^1-complete. We will consider a restriction of visibly pushdown languages. *Visibly counter languages* (VCLs) are languages recognized by *visibly counter automata*, which are visibly pushdown automata that only have one stack symbol, i.e. which only have a counter. In some sense, this class appears to be one of the smallest useful non-regular language classes, and hence our starting point in the program of generalizing results beyond the realm of regular languages.

Already in the 1980s, based in part on the result that PARTITY $\notin \mathrm{AC}^0$ [8,9], and on Barrington's theorem [4], deep connections between regular languages and circuit complexity were uncovered [6]. These results showed that circuit classes have characterizations in terms of algebraic properties of finite monoids, which has led to the characterization of class separations in terms of properties of regular sets. For instance, the classes ACC^0 and NC^1 differ if and only if no syntactic monoid of a regular set in ACC^0 contains a nonsolvable group. Indeed, the circuit classes AC^0, CC^0, $\mathrm{ACC}^0[n]$, ACC^0, and NC^1 can all be characterized using finite monoids and regular languages [15].

What stands out at this point is the absence of TC^0 in these results. Indeed, TC^0 does not relate to any family of regular languages: Assuming $\mathrm{NC}^1 \neq \mathrm{ACC}^0$, then either a regular language is NC^1-hard or it is contained in ACC^0 – depending on whether its syntactic monoid is solvable or not. This is where our contribution comes into play. We lift the correspondence between circuit classes and algebraic properties of regular languages to visibly counter languages. In previous work [10], we characterized the visibly counter languages in AC^0.

Our Results We examine the intersection of visibly counter languages with the circuit classes AC^0, $\mathrm{CC}^0[q]$, CC^0, $\mathrm{ACC}^0[q]$, ACC^0, TC^0, and NC^1. As our main theorem, we unconditionally prove criteria for visibly counter languages to be complete for these classes (Theorem 17). The results are summarized in Figure 1, which compares the known families of regular languages complete for various circuit classes with corresponding complete classes of visibly counter languages. Furthermore, given a visibly counter automaton, we can effectively compute the smallest of these complexity classes that contains the language recognized by this automaton. This also provides an effective method to decide whether a language, given by a visibly counter automaton, is in one of these complexity classes (Corollary 20). Decidability results for non-regular languages classes corresponding to circuit complexity classes are rather sparse. We expect that our more general method will help to lift various decidabilty results about regular languages to visibly counter languages.

Circuit Class	Regular Languages	Visibly Counter Languages	
	Syntactic Monoids	Stack	Regular Part
CC^0	Solvable Groups	No Stack	Solvable Groups
AC^0	**Quasi Aperiodic**	**Well Behaved**	**Quasi Aperiodic**
ACC^0	Solvable Monoids	Well Behaved	Solvable Monoids
TC^0	?	Any	Solvable Monoids
NC^1	**Finite Monoids**	**Any**	**Finite Monoids**

Fig. 1. Circuit Classes with complete classes of regular languages [15], and complete visibly counter classes. The classes in bold are maximal for the respective complexity class, while in the other cells, there may be a larger complete class depending on the separation of circuit classes.

As most separation questions between those languages classes are still open, we prove equivalences to these separations. For example we show that $\mathrm{CC}^0 \neq \mathrm{ACC}^0$ iff CC^0 containtains a non-regular visibly counter language. Assuming that $\mathrm{ACC}^0 \neq \mathrm{TC}^0 \neq \mathrm{NC}^1$, we can show that the visibly counter languages intersected with these languages are contained in logic classes using only semilinear predicates. This observation has been explored by [12], who introduced the notion of *extensional uniformity*, which refers to the study of the intersection of a complexity class with a family of formal languages. For $\mathrm{CC}^0[p]$, $\mathrm{ACC}^0[p]$ we provide similar but unconditional results.

Structure of the paper The paper first introduces the necessary notation in Section 2. Then we introduce our proof method, which splits the computation of visibly counter automata into two parts: *height computations* and a *regular part*. After developing various lemmas in Section 4, we present the main results in Section 5. We end with a discussion about further research.

2 Preliminaries

An *alphabet* is a finite set Σ. A subset of Σ^* is called a *language* For a word w, $|w|$ is the length of the word, $|w|_\Sigma$ the number of occurrences of symbols from Σ in w, and w_i is the letter in position i in w. For $L \subseteq \Sigma^*$, the set $F(L) \subseteq \Sigma^*$ is the set of all *factors* of words in L, i.e.: $F(L) = \{y \in \Sigma^* \mid \exists x, z \in \Sigma^* : \ xyz \in L\}$.

Every language $L \subseteq \Sigma^*$ induces a congruence on Σ^*, the *syntactic congruence* of L: For $x, y \in \Sigma^*$, $x \sim_L y$ iff for all $u, v \in \Sigma^*$ we have $uxv \in L \Leftrightarrow uyv \in L$. The *syntactic monoid* is the quotient monoid $\Sigma^*/\sim_L$. The *syntactic morphism* of L is the canonical projection $\eta_L \colon \Sigma^* \to \Sigma^*/\sim_L$.

The complexity classes we will consider are defined using circuit families. Precisely, we will study the following classes:

- AC^0: polynomial-size circuit families of constant depth with Boolean gates of arbitrary fan-in.
- $\mathrm{ACC}^0[k_1, ..., k_n]$: AC^0 circuit families plus modulo-k_i-gates for each i. ACC^0 is the union of $\mathrm{ACC}^0[k]$ for all k.
- TC^0: polynomial-size circuit families of constant depth with threshold gates of arbitrary fan-in.
- NC^0: polynomial-size circuit families of constant depth with bounded fan-in.
- NC^1: polynomial-size circuit families of logarithmic depth with bounded fan-in.

If for some $n \in \mathbb{N}$ the circuit with input length n is computable in some complexity bound, we speak of uniformity. One prominent example is so called DLOGTIME-uniformity. For further references on circuit complexity, cf. e.g., [16].

If L, L' are languages, then a *quantifier-free reduction* (often called quantifier-free projection) from L to L' is a constant-depth circuit family for L whose only gates are NC^0 gates and an oracle gate for L'.

Let M, N be two monoids. We say M *divides* N iff there is a subset $N' \subseteq N$ and a surjective monoid morphism $N' \to M$. A nontrivial monoid is *simple* if it has no nontrivial monoid congruences. The simple monoids are the aperiodic unit $U_1 = (\{0,1\}^*, \cdot)$, the Abelian simple groups $\mathbb{Z}_p$ (p prime), and the non-Abelian simple groups, which are the perfect groups, whose smallest one is the permutation group A_5. Every finite monoid is divided by a simple monoid.

Related to circuit classes is the framework of logic over finite words. A nonempty word $w \in \Sigma^+$ defines a structure whose domain is $\{1, ..., |w|\}$. For every $a \in \Sigma$, there is a predicate Q_a such that $w \models Q_a(i)$ iff $w_i = a$. Furthermore, any predicate defined on $\mathbb{N}_1$ can be interpreted canonically in such a structure; such predicates are called *numerical predicates*. Important examples are the order predicate $<$ with arity two and an obvious semantics, and the arithmetic predicate $+$ of arity three: $(i, j, k) \in +$ iff $i + j = k$. We allow first-order existential and universal quantification. Furthermore, we will use quantifiers over finite monoids. Let M be a monoid, $M_+ \subseteq M$, $N \in \mathbb{N}$, let $\psi_1(x), ..., \psi_N(x)$ be formulas, let $\delta : \{0,1\}^N \to M$ be a map, let $w \in \Sigma^*$ be a word, and let v be an assignment of variables to positions in w. Then $Q^{M,\delta,M_+} x\ \psi_1(x)...\psi_N(x)$ is a formula, which is true in w under the assignment v iff

$$\Pi_{i=1}^{|w|} \delta(w \models \psi_1(i), ..., w \models \psi_N(i)) \in M_+$$

where the product symbol denotes the multiplication of the monoid M. Informally, a quantifier over a monoid labels each position with a monoid element depending on which formulas $\psi_i(x)$ are true at the position, and then checks whether the resulting word over the monoid evaluates to an element of M_+. First order quantification is equivalent to quantification using the monoid $U_1 = (\{0,1\}, \cdot)$, and modulo quantifiers are equivalent to quantifiers over cyclic groups. Furthermore, there is the majority quantifier Maj, where $Maj\ x\ \psi(x)$ is true if $\psi(i)$ is true for more than half of the positions i in the word. We write Reg for the set of regular predicates, and SL for the set of semilinear predicates, i.e., those definable using regular predicates and $+$.

If Q is a set of finite monoids and quantifiers, and P is a set of predicates, we write $Q[P]$ for the set of formulas built from P and Q, and also for the set of languages recognized by these formulas. We also write FO for U_1 quantification, and MOD for quantification over cyclic groups. We write arb for the set of all numerical predicates. Then we have $\mathrm{AC}^0 = FO[arb]$, $\mathrm{CC}^0[k] = \mathbb{Z}_k[arb]$, $\mathrm{ACC}^0[k] = FO + \mathbb{Z}_k[arb]$, and $\mathrm{TC}^0 = \mathrm{Maj}[arb]$; see e.g. [16].

Mehlhorn [13] and later independently Alur and Madhusudan [2] introduced *input-driven* or *visibly pushdown automata*, which are pushdown automata in which the input symbol determines whether a symbol is pushed or popped. This leads to a partitioning of Σ into call, return and internal letters: $\Sigma = \Sigma_{\mathrm{call}} \cup \Sigma_{\mathrm{ret}} \cup \Sigma_{\mathrm{int}}$. We will assume that Σ has some fixed partitioning. The *height* of $w \in \Sigma^*$ is given by $\Delta : \Sigma^* \to \mathbb{Z} : w \mapsto |w|_{\Sigma_{\mathrm{call}}} - |w|_{\Sigma_{\mathrm{ret}}}$. A word w is *well-matched* if $\Delta(w) = 0$ and $\Delta(w_1...w_i) \geq 0$ for all $1 \leq i \leq |w|$.

The concept of *visibly counter automata* (VCA) was introduced by Bárány et al. [3]. Since every VPA can be determinized and this is also true for visibly counter automata, we restrict attention to deterministic automata:

Definition 1 ($m-$VCA)**.** *An* $m-$VCA $\mathcal{A}$ *over* $\hat{\Sigma} = (\Sigma_{\text{call}}, \Sigma_{\text{ret}}, \Sigma_{\text{int}})$ *is a tuple* $\mathcal{A} = (Q, q_0, F, \hat{\Sigma}, \delta_0, \ldots, \delta_m)$*, where* $m \geq 0$ *is the* threshold*,* Q *is the set of* states*,* q_0 *the* initial state*,* F *the set of* final states*, and* $\delta_i \colon Q \times \Sigma \to Q$ *are the* transition functions*.*

A *configuration* of an $m-$VCA is an element of $Q \times \mathbb{N}$. When reading a letter $\sigma \in \Sigma$, an $m-$VCA $\mathcal{A}$ performs the transition $(q, k) \xrightarrow{\sigma} (\delta_{\min(m,k)}(q, \sigma), k+\Delta(\sigma))$. Then $w \in L(\mathcal{A})$ iff $(q_0, 0) \xrightarrow{w} (f, h)$ for $f \in F$ and $h \geq 0$. Visibly counter automata, and more generally visibly pushdown automata, can only recognize words where the height of all prefixes is non-negative. All other words are rejected.

Definition 2 (VCL)**.** *A language is a* visibly counter language *(*VCL*) iff it is recognized by an* $m-$VCA *for some* m*.*

3 Decomposing Visibly Counter Automata

A first step towards classifying visibly counter languages with concern to their complexity was made in [10], where we gave a decidable characterization of the visibly counter languages in AC^0. The proof is based on considering separately the computation of the height profile and the simulation of the finite-state control. In this section we survey the main points of [10].

3.1 Height Behaviour

For the height computations, the previous work considered the problem of defining unary predicates $H_n(\cdot)$ that are true if a position in a given word iff the symbol at that position has height n.

We defined a notion of *simple height behavior* which guarantees that these height predicates can, in some sense, be approximated in AC^0. Informally, a visibly counter language has simple height behavior if whenever a recognizing automaton loops through a state q reading a subword w, the height profile $\Delta(w)$ is determined by q and the length $|w|$. More formally, the following notions were considered:

Definition 3 (fixed slope [10]). *We say that a state* q *has a* fixed slope *if there are numbers* $\alpha \in \mathbb{Q}$ *and* $\gamma \in \mathbb{N}$ *so that if for all words* $w \in \Sigma^*$ *with* $(q, h_1) \xrightarrow{w} (q, h_2)$ *and* $h_1 + \Delta(w') \geq m$ *for all prefixes* w' *of* w *it holds that:*

- $h_2 = \alpha|w| + h_1$
- $\alpha|w'| - \gamma \leq \Delta(w') \leq \alpha|w'| + \gamma$ *for all prefixes* w' *of* w

Definition 4 (active [10]). *A state q is* active *if there is a word $w \in L(\mathcal{A})$ with positions i and j, $i < j$, such that after reading $w_1 \cdots w_i$, $\mathcal{A}$ is in q, $\Delta(w_1 \cdots w_i) > m + |Q|$ and $\Delta(w_1 \cdots w_i) - \Delta(w_1 \cdots w_j) > |Q|$.*

The first result is that languages recognized by an $m-$VCA which has an active state without fixed slope are TC^0-hard (Theorem 6). This permitted the following definition:

Definition 5 (simple height behavior [10]). *If in a* VCA $\mathcal{A}$ *all active states have a fixed slope, we say that the recognized language $L(\mathcal{A})$ has simple height behavior.*

Theorem 6. *If a visibly counter language does not have simple height behavior, it is hard for* TC^0 *by quantifier-free reductions.*

In the case of simple height behavior we know that the height is computable in FO[+]:

Theorem 7. *Given some $m-$*VCA $\mathcal{A}$ *such that $L(\mathcal{A})$ has simple height behavior, we can define for every $k < m$ a monadic predicate $H_k(x)$ in FO[+] such that:*

- $w_{x=i} \models H_k(x)$ *then* $\Delta(w_1 \dots w_{i-1}) = k$ *for arbitrary* $w \in \Sigma^*$.
- $w_{x=i} \models H_k(x)$ *iff* $\Delta(w_1 \dots w_{i-1}) = k$ *for all* $w \in L$.

We let $H_{\geq m} = \neg \bigvee_{k=0}^{m-1} H_k$ be the negation of these predicates. Hence for $w \notin L$ the predicate might have false-positives, i.e., the predicate might suggest a stack-height greater or equal to m while in fact it is less than m.

So in conclusion we get a dichotomy: Either the height is computable in AC^0, or it is TC^0- hard.

3.2 Regular Part

For simulating the behavior of the finite state control, we considered a regular language, which simulates the actions of the automaton when the height information is already coded into the input.

Definition 8 (Height transduction [10]). *For $m \in \mathbb{N}$ we let $\Sigma_m = \Sigma \times \{0, \dots, m\}$ and set $\Delta_m(w) = \min(\Delta(w), m)$. Then define $\tau_m : \{w : \Delta(w) \leq 0\} \to (\Sigma_m)^*$ by $\tau_m(w_1 w_2 \cdots w_n) =$*
$(w_1, \Delta_m(\epsilon))(w_2, \Delta_m(w_1)) \cdots (w_i, \Delta_m(w_1 \cdots w_{i-1})) \cdots (w_n, \Delta_m(w_1 \cdots w_{n-1}))$

A word $w \in (\Sigma_m)^*$ is valid *if $w \in F(\tau_m(\Sigma^*))$. We call i the* label *of the letter $(a, i) \in \Sigma_m$.*

Informally, τ_m labels symbols with their height up to the threshold m. For instance, if a and b are push and pop letters respectively, then $\tau_2(aaaab) = (a,0)(a,1)(a,2)(a,2)(b,2)$.

The action of the finite state control is modeled by a regular language, the *regular part*:

Definition 9 ($R_{\mathcal{A}}$ [10]). *Let $\mathcal{A} = (Q, q_0, F, \hat{\Sigma}, \delta_0, \ldots, \delta_m)$ be an $m-$VCA. Let M be the finite automaton $M = (Q, q_0, F, \Sigma_m, \delta)$, where $\delta(q, (a, i)) = \delta_i(q, a)$. Then set $R_{\mathcal{A}} := L(M)$.*

$R_{\mathcal{A}}$ is a regular language over the alphabet Σ_m. For $\Delta(w) > 0$, we have $w \in L(\mathcal{A})$ iff $\tau_m(w) \in R_{\mathcal{A}}$. In general, $R_{\mathcal{A}}$ also contains words that are not in the image of τ_m, and the choice of $\mathcal{A}$ determines which these words are.

4 Simple Monoids and the Regular Part

We now show how the concepts of simple height behavior and the regular part can be used to more generally obtain characterizations for the visibly counter languages in low complexity classes. It will become clear that, as in the case of regular languages, the simple monoids contained in the language play a major role in determining its complexity.

Let M be a finite monoid. By the *word problem of M*, we denote the set

$$\{w \in M^* : \pi(w) = 1_M\}$$

of words over M evaluating to the identity element of M, where $\pi : M^* \to M$ is the canonical morphism. Observe, this language has a neutral letter.

The intersection of a constant-depth circuit class C with the regular languages is known to be determined already by the class of simple monoids whose word problem is in C [15]. Broadly speaking, a regular language L is in a circuit class if and only if, for all simple monoids 'contained in' L, their word problems are also in C. In the regular case, the appropriate notion of a monoid M 'being contained' in a language L with syntactic morphism η_L is that there a $t > 0$ such that $\eta_L(\Sigma^t)$ contains a monoid divided by M [5]. In the case of visibly counter languages, the precise meaning of 'being contained' in the language will be somewhat more intricate. We begin with the aperiodic unit U_1, for which the statement is the following:

Lemma 10 (Reducing U_1 to L). *Let L be a visibly-counter language. If L is not regular, then the word problem of U_1 can be reduced to L by a quantifier-free reduction.*

Proof. See appendix.

The other simple monoids are the simple groups. For reducing the word problem of a group to a visibly counter language, we need a technical observation from [10], which states that every $m-$VCA can be transformed into a certain normal form, referred to as *loop-normal* (see Definition 24 in the appendix). This normal form has the property that, if the automaton goes through a loop in reading a prefix that can be completed to an accepted word, then it can still be completed when more loops through the same state are appended. Using the following lemma we can always assume that a given $m-$VCA $\mathcal{A}$ is loop-normal.

Lemma 11 ([10]). *For every visibly counter language L, there is a loop-normal $m-$*VCA *$\mathcal{A}$ recognizing L.*

Now we can make precise in which way a simple monoid has to be 'contained in' a visibly counter language to play a role in its complexity:

Definition 12. *Let L be a visibly counter language. We choose a set Q_L of simple monoids as follows. Let $\mathcal{A}$ be some loop-normal $m-$*VCA *$\mathcal{A}$ such that $L = L(\mathcal{A})$. Then set Q_L to be the set of simple monoids $\mathcal{N}$ such that there is a number $t > 0$ and a set $N \subseteq (\Sigma_m)^t$ with $N^* \subseteq F(\tau_m(\Sigma^*))$ so that $\eta_{R_\mathcal{A}}(N)$ is a monoid divided by $\mathcal{N}$.*

Informally, we take those simple monoids which can be simulated by words over a set N such that all words over N are valid. Our results will not depend on the choice of the $\mathcal{A}$ used for constructing Q_L, as long as $\mathcal{A}$ is loop-normal. Then the adequate notion of '$R_\mathcal{A}$ contains $\mathcal{G}$' is that $\mathcal{G}$ is an element of Q_L, as shown by the following lemma:

Lemma 13 (Reducing Groups to L). *Let L be a visibly-counter language and $\mathcal{G} \in Q_L$ a simple group. Then the word problem of $\mathcal{G}$ is reducible to L by quantifier-free reductions.*

Proof. See appendix.

Thus, a visibly counter language is at least as hard as the word problems of the groups contained in Q_L. Putting these hardness results together, we have:

Theorem 14 (Hardness). *Let L be a* VCL*.*

- *If L is non-regular, it is AC^0-hard.*
- *If L does not have simple height behaviour, it is TC^0-hard.*
- *If Q_L contains cyclic groups $Z_{p_1}, \ldots, Z_{p_k}$, then L is $\mathrm{CC}^0[p_1 \cdots p_k]$-hard*
- *If Q_L contains a non-abelian simple group, then L is NC^1-hard.*

Our notion of completeness is via quantifier free reductions.

Proof. The second statement is Theorem 6. The word problem of U_1 is AC^0-hard, the word problem of $\mathbb{Z}_p$ is $\mathrm{CC}^0[p]$-hard, and the word problem of non-abelian simple groups is NC^1-hard. Thus, the first statement is Lemma 10, and the other statements follow from Lemma 13.

These lower bounds on complexity are complemented by Corollary 16, which shows that the bounds essentially are tight. We first show in Lemma 15 that, when the height profile is already known, the language can be defined using only quantifiers from Q_L:

Lemma 15. *Let $\mathcal{A}$ be an $m-$*VCA *recognizing L. There is a $Q_L[Reg]$ formula ϕ with*

$$L(\phi) \cap \tau_m(\Sigma^*) = R_\mathcal{A} \cap \tau_m(\Sigma^*).$$

Proof. See appendix.

Together with our results on height behaviour and the definability of height predicates, we find that indeed the bounds from Theorem 14 are tight:

Corollary 16 (Definability). *Let L be a* VCL*. Then $L \in Q_L + \mathrm{Maj}[<]$. Furthermore, if L has simple height behavior, we have $L \in Q_L + FO[SL]$, and if L is regular, then $L \in Q_L[SL]$.*

Proof. See appendix.

5 Visibly Counter Languages and Circuit Classes

Putting the results from the previous section together, we have the following unconditional completeness results for VCLs:

Theorem 17 (Main Theorem). *Let L be a* VCL*.*

- *If L is regular and Q_L contains exactly the simple cyclic groups $\mathbb{Z}_{p_1}, \ldots, \mathbb{Z}_{p_k}$ then L is $\mathrm{CC}^0[p_1 \cdots p_k]$-complete.*
- *If L is non-regular and does has simple height behaviour and Q_L does not contain a simple group then L is AC^0-complete.*
- *If L is non-regular and does has simple height behaviour and Q_L contains at most U_1 and simple cyclic groups $Z_{p_1}, \ldots, Z_{p_k}$ then L is $\mathrm{ACC}^0[p_1 \cdots p_k]$-complete.*
- *If L does not have simple height behaviour and Q_L does not contain a non-abelian simple group then L is TC^0-complete.*
- *If Q_L contains a non-abelian simple group then it is NC^1-complete.*

Our notion of completeness is via quantifier free reductions.

Proof. For all these completeness statements, the hardness part was shown in Theorem 14. Completeness follows from Corollary 16 as follows. If L is regular and $Q_L = \{\mathbb{Z}_{p_1}, ..., \mathbb{Z}_{p_k}\}$, we have $L \in (\mathbb{Z}_{p_1}, ..., \mathbb{Z}_{p_k})[Reg] \subset (\mathbb{Z}_{p_1}, ..., \mathbb{Z}_{p_k})[arb] = CC^0[p_1, ..., p_k]$. If Q_L contains a non-abelian simple group, L is NC^1-hard by Barrington's theorem, and since all VCLs are in NC^1, L is NC^1-complete. Now assume L is nonregular but all monoids in Q_L are commutative. If L has simple height behaviour, it is in $FO + Q_L[SL]$, which is in $\mathrm{ACC}^0[p_1, ..., p_k]$ whenever $Q_L \subseteq \{U_1, p_1, ..., p_k\}$. If Q_L contains no group, $L \in FO[SL] \subset \mathrm{AC}^0$. Finally, if L does not have simple height behaviour, we have $L \in \mathrm{Maj}[<] \subset \mathrm{TC}^0$ by Corollary 16, since all abelian groups are computable in $\mathrm{Maj}[<]$.

This result shows that separation conjectures for these circuit classes are equivalent to conjectures about the membership of visibly counter languages in these classes, for instance:

Corollary 18. $\mathrm{ACC}^0 = \mathrm{TC}^0$ *iff* $\mathrm{ACC}^0 \cap \mathrm{VCL} = \mathrm{TC}^0 \cap \mathrm{VCL}$
$\mathrm{CC}^0 = \mathrm{ACC}^0$ *iff* $\mathrm{CC}^0 \cap \mathrm{VCL} = \mathrm{CC}^0 \cap Reg$

Hence, while the separation between ACC^0 and TC^0 cannot be reduced to a question about regular languages, it can indeed be viewed as a question about visibly counter languages.

We also obtain decidability results, assuming a visibly counter language is encoded by any $m-$VCA recognizing it. We can show this rather abstractly for arbitrary circuit classes. Precisely, by a (constant-depth) circuit class $\mathcal{C}$ over a set of gates, we mean the class of languages recognized by constant-depth, polynomial size circuit families over this set of gates. We assume that any class is closed under quantifier-free reductions, i.e., has bounded fan-in boolean gates. By results of [15], the regular languages in any such class are already determined by the simple monoids whose word problems are in the class. For all such classes, we have:

Corollary 19. *Let $\mathcal{C}$ be a circuit class. There is a computable reduction from the membership problem of $VCL \cap \mathcal{C}$ to the membership problem of $Reg \cap \mathcal{C}$.*

Proof. First consider the case when $\mathrm{AC}^0 \subseteq \mathcal{C}$. It was shown in [10] that it is decidable whether an $m-$VCA $\mathcal{A}$ has simple height behavior. If $L(\mathcal{A})$ does not have simple height behaviour and $\mathrm{TC}^0 \not\subseteq \mathcal{C}$, then certainly $L(\mathcal{A}) \notin \mathcal{C}$ by Theorem 6. Otherwise, membership in $\mathcal{C}$ only depends on the groups in Q_L by Lemma 13 and Corollary 16. The construction of loop-normal automata in [10] is constructive, so the set Q_L is computable, and we obtain a reduction from the membership problem of $\mathrm{VCL} \cap \mathcal{C}$ to the membership problem of $Reg \cap \mathcal{C}$. Now if $\mathrm{AC}^0 \subseteq \mathcal{C}$, then since $\mathcal{C}$ is closed under quantifier-free reductions, $L(\mathcal{A})$ must be regular if $L(\mathcal{A}) \in \mathcal{C}$ by Lemma 10. We can decide as follows whether $L(\mathcal{A})$ is regular. $L(\mathcal{A})$ certainly is regular if there is a constant C such that whenever a word in $L(\mathcal{A})$ has a symbol at height N, then no later symbol can have height $< N - C$. To decide this property, it suffices to search for a word w with $\Delta(w) < -2(m + |Q|)$, where Q is the state set of $\mathcal{A}$, that occurs in a word of $L(\mathcal{A})$, i.e., $w \in F(L(\mathcal{A}))$. In any other case, $L(\mathcal{A})$ cannot be regular by a pumping argument. □

Since membership in $\mathrm{TC}^0 \cap Reg$ and $\mathrm{ACC}^0 \cap Reg$ at most depends on the presence of nonsolvable groups and thus is decidable, we get:

Corollary 20. *Given a* VCL *L, it is decidable whether L is in* AC^0*,* CC^0*,* ACC^0*, or* TC^0*, respectively.*

However, it has to be noted that for most classes, the concrete decision algorithm depends on open questions about the separation of classes, as applying it to suitable languages would immediately settle these questions. The algorithm is unconditionally known in the case of AC^0 due to its separation from ACC^0, and also for $\mathrm{ACC}^0[p]$ for p a prime number by a result of [14]:

Corollary 21. *A visibly counter language L is in* $\mathrm{ACC}^0[p]$ *for p a prime number if and only if L has simple height behavior, and $Q_L \subseteq \{U_1, \mathbb{Z}_p\}$.*

Under the separation conjectures, we also obtain uniformity results:

Corollary 22. *If* $\mathrm{TC}^0 \neq \mathrm{NC}^1$*, then* $\mathrm{TC}^0 \cap \mathrm{VCL} \subseteq \mathrm{Maj}[<] \subseteq$ *DLOGTIME-uniform* TC^0*.*

If $\mathrm{ACC}^0 \neq \mathrm{TC}^0$*, then* $\mathrm{ACC}^0 \cap \mathrm{VCL} \subseteq FO + MOD[+] \subseteq$ *DLOGTIME-uniform* ACC^0*.*

Proof. Regular languages with solvable syntactic monoids are computable in $Maj<$ and in FO+MOD[+]. The claim then follows from Corollary 16. □

We now explore a somewhat different and more abstract interpretation of our results. Let S be the set consisting of the quantifiers over simple finite monoids. For each subset $Q \subseteq S$, we have a logic class $Q[SL]$, so we obtain a lattice $\mathcal{L} = \{Q[SL] \mid Q \subseteq S\}$ of logic classes. When A is a class of finite simple monoids and $M \notin A$ is also simple, then the word problem of M is not in $A[SL]$ [11]. We can show unconditionally:

Corollary 23. *Given a visibly counter language L with simple height behaviour, one can effectively compute the smallest logic class of $\mathcal{L}$ that can define L.*

Proof. See appendix.

Assuming a similar statement holds for the majority quantifier, i.e., that whenever A is a (possibly empty) set of perfect groups and $G \notin A$ is also perfect, then the word problem of G is not in $A + \mathrm{Maj}[<]$, then this holds for all visibly counter languages, when Maj is added to S.

6 Discussion

This work considered relations between complexity classes and families of formal languages. This line of research has hitherto mainly been pursued for regular languages, since their syntactic monoids are finite and thus are open to algebraic methods. We made a step to extend these results towards nonregular languages. The most natural candidate to choose were the visibly pushdown languages, where we focused on visbly counter languages. We derived decidable characterizations for ACC^0, CC^0, and TC^0, and indeed for a very large class of circuit classes, and related our results to the open questions of the relationships of these classes.

As a next step we naturally want to extend our approach. For instance, we will try to generalize our results to all visibly pushdown languages, but we probably will need new methods.

A promising approach appear to be forest algebras [7], which complement the *horizontal* Alur monoid [1] by a *vertical* counterpart. They could provide us with purely algebraic proofs of our results without the need for combinatorial arduousness, and possibly extend to general visibly pushdown languages.

References

1. Rajeev Alur, Viraj Kumar, P. Madhusudan, and Mahesh Viswanathan. Congruences for Visibly Pushdown Languages. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 1102–1114. Springer, 2005.
2. Rajeev Alur and P. Madhusudan. Visibly pushdown languages. In László Babai, editor, *STOC*, pages 202–211. ACM, 2004.
3. Vince Bárány, Christof Löding, and Olivier Serre. Regularity Problems for Visibly Pushdown Languages. In Bruno Durand and Wolfgang Thomas, editors, *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23-25, 2006, Proceedings*, volume 3884 of *Lecture Notes in Computer Science*, pages 420–431. Springer, 2006.
4. 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.
5. David A. Mix Barrington, Kevin J. Compton, Howard Straubing, and Denis Thérien. Regular languages in nc^1. *J. Comput. Syst. Sci.*, 44(3):478–499, 1992.
6. David A. Mix Barrington and Denis Thérien. Finite monoids and the fine structure of NC^1. *J. ACM*, 35(4):941–952, 1988.
7. Mikolaj Bojanczyk and Igor Walukiewicz. Forest algebras. In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas].*, volume 2 of *Texts in Logic and Games*, pages 107–132. Amsterdam University Press, 2008.
8. Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. In *FOCS*, pages 260–270, 1981.
9. Johan Håstad. Almost optimal lower bounds for small depth circuits. In *STOC*, pages 6–20. ACM, 1986.
10. Andreas Krebs, Klaus-Jörn Lange, and Michael Ludwig. Visibly Counter Languages and Constant Depth Circuits. In Ernst W. Mayr and Nicolas Ollinger, editors, *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, volume 30 of *LIPIcs*, pages 594–607. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.
11. Andreas Krebs and A. V. Sreejith. Non-definability of Languages by Generalized First-order Formulas over (N, +). In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*, pages 451–460. IEEE Computer Society, 2012.
12. Pierre McKenzie, Michael Thomas, and Heribert Vollmer. Extensional uniformity for boolean circuits. *SIAM J. Comput.*, 39(7):3186–3206, 2010.
13. Kurt Mehlhorn. Pebbling mountain ranges and its application to DCFL-recognition. In Jaco de Bakker and Jan van Leeuwen, editors, *Automata, Languages and Programming*, volume 85 of *Lecture Notes in Computer Science*, pages 422–435. Springer Berlin Heidelberg, 1980.
14. Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *STOC*, pages 77–82, 1987.
15. Howard Straubing. *Finite Automata, Formal Logic, and Circuit Complexity.* Birkhäuser, Boston, 1994.
16. Heribert Vollmer. *Introduction to circuit complexity - a uniform approach.* Texts in theoretical computer science. Springer, 1999.

A Appendix

Proof of lemma 10 The word problem of U_1 is the language AND $= 1^* \subseteq \{0,1\}^*$. Let q be the number of states of $\mathcal{A}$. If the stack height of $\mathcal{A}$ during the run of all words in L is bounded by a constant, then L is already regular as the counter can be simulated by finitely many states. Similar if there is a constant c such that for all w in L the stack height never decreases more than c during the accepting run of w, we could count the stack height up to the threshold $c+m$ by finitely many states and L is regular.

Otherwise for every constant c we will find a word w where the stack height decreases by more than c during the computation of $\mathcal{A}$ on w. We will show that we can reduce AND to L in this case. Let $w \in L$ be such a word where the stack height decreases by at least $m+q+2$. This allows us to split the word $w = stuv$ such that the stack height after processing s is m and while processing t and u is always at least m and after processing st is $h > m+q+1$ and after processing stu is $m < h' < h-q$. As the stack height increases while processing t by more than q we can find a loop in the states that increases of d_t the stack height, similar we can find a decreasing of d_u loop while processing u. This allows us to split w further to $w = st_1t_2t_3u_1u_2u_3v$ where t_2 and u_2 induce a loop in the states. Hence we have $w_n = st_1t_2^{nd_y}t_3u_1u_2^{nd_x}u_3v \in L$ for all $n \in \mathbb{N}$.

We will now reduce AND to L. Let k be the stack height in the accepting run of w after processing w. Note that this is the same height as in the accepting run of w_n after processing w_n.

Let k be an arbitrary number and let $n = km$. Then w_n has at least n push letters. Let $B_{i,n} \subseteq \mathbb{N}$ be the positions of $ik+1$-th up to the $ik+k$-th push letter of w. Also let b be any pop letter in Σ. We define a map f from $x_1 \dots x_m \in \{0,1\}^*$ to $y_1 \dots y_n \in \Sigma^*$. Informally speaking, this mapping will map 1^n to w_n and hence to a word in L, and all other words to a word not in L by decreasing the stack height below 0. The latter is easily accomplished by flipping k push letters of w to pop letters for every 0 encountered in the input.

Hence we define $f(y)_j = b$ if $j \in B_{i,n}$ and $x_i = 0$, otherwise $f(y)_j = (w_n)_j$.

This defines a quantifier free projection from AND to L. □

Definition 24 (Loop-Normal). *An $m-$*VCA *$\mathcal{A}$ is called* loop-normal *if for all $x, y \in \Sigma^*$ with $\Delta(xy_1 \cdots y_k) \geq m$ for $0 \leq k \leq |y|$ and $(q_0, 0) \xrightarrow{x} (q, h_1) \xrightarrow{y} (q, h_2)$, $q \in Q$ then either q is a dead state, or one of the following is true, depending on $\Delta(y)$:*

- *If $\Delta(y) > 0$, then for each $z \in \Sigma^*$ such that $\Delta(xyz) \geq 0$, there is a partition of z into $z = z_1z_3$ and a word $z_2 \in \Sigma^*$ so that for all $i \geq 0$ we have that $xyy^iz_1z_2^iz_3 \in L(\mathcal{A})$ iff $xyz \in L(\mathcal{A})$.*
- *If $\Delta(y) < 0$, then there is a partition of x into $x = x_1x_3$ and a word $x_2 \in \Sigma^*$ so that for all $i \geq 0$ and for all for each $z \in \Sigma^*$ such that $\Delta(xyz) \geq 0$, we have that $x_1x_2^ix_3yy^iz \in L(\mathcal{A})$ iff $xyz \in L(\mathcal{A})$.*
- *If $\Delta(y) = 0$, then for all $i \geq 0$ and for each $z \in \Sigma^*$ such that $\Delta(xyz) \geq 0$, $xy^iz \in L$ iff $xyz \in L(\mathcal{A})$, and*

if $\delta_i = \delta_m$ *for* $m - |Q| < i < m$.

Proof of lemma 13. In the following, we denote by $\overline{a}$ the projection of $a \in (\Sigma_m)^*$ to the first component.

CASE A*:* $\mathcal{G}$ is abelian. Thus $\mathcal{G} = \mathbb{Z}_p$ for some prime p.

If L does not have simple height behaviour, then by Theorem 6, it is hard for TC^0 by quantifier-free reductions. Since the word problems of abelian groups have TC^0 circuits, this yields a quantifier-free reduction from $\mathcal{G}$ to L.

Now assume L has simple height behaviour. In the following we identify the elemnts of the cyclic group $\mathcal{G}$ with $\{0, 1, 2 \cdots, p-1\}$. We now want to reduce the word problem of $\mathcal{G}$ to L.

The word problem $WP_{\mathcal{G}}$ of $\mathcal{G}$ consists of all those words $w \in \{0, 1, 2 \cdots, p-1\}^*$ for which $\sum_{j=1}^{|w|} w_j = 0 \; mod \; p$. We now consider the set $L_p := WP_{\mathcal{G}} \cap \{0,1\}^*$. $WP_{\mathcal{G}}$ is reduced to L_p by the AC^0-computable homomorphism $i \to 0^{p-i-1}1^i$. Hence it suffices to reduce L_p to $L(\mathcal{A})$.

Since , for the set G from the definition of Q_L, there is a subset G' such that $\eta_{R_A}(G') \cong \mathcal{G}$ by Cauchy's theorem. So we may assume $\eta_{R_A}(G) \cong \mathcal{G}$. Let t be as in the definition of Q_A.

Let $\psi : \mathcal{G} \to G$ be an assignment of group elements to words of length t such that $\eta_{R_A}(\psi(g)) = g$ for each $g \in G$, where we have identified $\eta_{R_{\mathcal{A}}}(G)$ with $\mathcal{G}$. Because $\mathcal{A}$ has simple height behavior, we know that $\Delta(\psi(g))$ is independent of g, since if $\Delta(\psi(g)) \neq \Delta(\psi(g'))$,$\psi(g)^p$ and $\psi(g')^p$ would be two words demonstrating $\mathcal{A}$ not to have simple height behavior. We continue ψ to a morphism $\mathcal{G}^* \to \Sigma^*$, assigning words over Σ to words over $\mathcal{G}$.

Take $g := 1 \in \mathbb{Z}_p = \mathcal{G}$.

- *Case I:* $\Delta(\psi(g)) = 0$. Since all words over $\psi(G)$ are valid, we know that all $w \in \psi(G)$ start and end at the same height. Now there is a separating pair of words $u, v \in (\Sigma_m)^*$ such that, say, $u\psi(1_{\mathcal{G}})v \in R_{\mathcal{A}}$ and $u\psi(g)v \notin R_{\mathcal{A}}$, and $\overline{u}\overline{\psi(1_{\mathcal{G}})}\overline{v} \in L(\mathcal{A})$ and $\overline{u}\overline{\psi(g)}\overline{v} \notin L(\mathcal{A})$. This yields a quantifier-free-computable map $f_g : \mathcal{G}^* \to \Sigma^*$ by $w \mapsto \overline{u}\overline{\psi(w)}\overline{v}$.
- *Case II:* $\Delta(\psi(g)) \neq 0$. We assume $\Delta(\psi(g))$ to be greater 0; the other case can be seen analogously.
  Since all words over G^* are valid, we know $G \subseteq (\Sigma \times \{m\})^t$. Again, there is a separating pair of words $u, v \in (\Sigma_m)^*$ such that, say, $u\psi(1_{\mathcal{G}})v \in R_{\mathcal{A}}$ and $u\psi(g)v \notin R_{\mathcal{A}}$. Since $\Delta(g)$ is independent of g, we can choose them so that $\overline{u}\overline{\psi(1_{\mathcal{G}})}\overline{v} \in L(\mathcal{A})$ and $\overline{u}\overline{\psi(g)}\overline{v} \notin L(\mathcal{A})$. We may assume that $\mathcal{A}$ is in the same state after reading $\overline{u}$ and $\overline{u}\overline{\psi(1_{\mathcal{G}})}$. Hence, since $\mathcal{A}$ is loop-normal, v is partitioned into v_1v_3 such that there is a word v_2 such that for $g \in \mathcal{G}$, $u\psi(g)\psi(g)^n v_1 v_2^n v_3 \sim_{R_{\mathcal{A}}} u\psi(g)v_1v_3$. Hence, for $w \in \mathcal{G}^*$, $u\psi(1_{\mathcal{G}})\psi(w)v_1v_2^{|w|}v_3$ is in $L(\mathcal{A})$ if w evaluates to $1_{\mathcal{G}}$ and not in $L(\mathcal{A})$ if w evaluates to g. Then we can define a quantifier-free-computable map $f_g : \mathcal{G}^* \to \Sigma^*$ by $w \mapsto \overline{u}\overline{\psi(1_{\mathcal{G}}w)}\overline{v}\overline{v_2}^{|w|}\overline{v_3}$.

Now by Fermat's little theorem, $h^{|\mathcal{G}|-1} = g$ for each $h \neq 0 = 1_G$ and $1_\mathcal{G}$ for $h = 1_\mathcal{G}$. We now want to reduce L_p to $L(\mathcal{A})$. For a $w \in \{0,1\}^*$ we consider the word v which is the product of the conjunction $w_{i_1} \wedge w_{i_2} \cdots \wedge w_{i_{p-1}}$ over all $(p-1)$-tuples $1 \leq i_1, \cdots, i_{p-1} \leq n$. If i is the number of 1-symbols in w then v contains i^{p-1} 1-symbols. Thus we have, $w \in L_p$ iff $f_g(v) \in L(\mathcal{A})$. Since the mapping $w \to v$ is AC^0-computable, we have (many-one) reduced $WP_\mathcal{G}$ to $L(\mathcal{A})$.

Case B*:* $\mathcal{G}$ is non-abelian

Let G and t be as in the definition of Q_L. Since $\mathcal{G}$ divides $\eta_{R_\mathcal{A}}(G)$, there is a monoid $U \subseteq \eta_{R_\mathcal{A}}(G)$ and a surjective monoid morphism $\phi : U \to \mathcal{G}$.

As above, choose an assignment $\psi : \mathcal{G} \to G$ of group elements to words of length t such that $\phi(\eta_{R_\mathcal{A}}(\psi(g))) = g$ for each $g \in \mathcal{G}$.

We now modify ψ in such a way that $\Delta(\psi(g))$ is independent of g. Since $\mathcal{G} = \{g_1, ..., g_n\}$ is simple and nonabelian, it is perfect, and every element can be represented as a product of commutators. Now $[g,h] = ghg^{|\mathcal{G}|-1}h^{|\mathcal{G}|-1}$, and thus is also represented by a word of length $|\mathcal{G}| \cdot (|\psi(g)| + |\psi(h)|)$. Since $g^{|\mathcal{G}|} = 1_\mathcal{G}$, any element can be represented by a word in which each $\psi(g_i)$ occurs exactly $|\mathcal{G}|$ times. We now replace $\psi(g)$ by the words constructed in this manner. Then $|\psi(g)|$ and $\Delta(\psi(g))$ are independent of g.

We choose some element $g \neq 1_\mathcal{G}$ and construct a map f_g exactly as in the abelian case. Now consider the operation $\mathcal{G} \to \mathcal{G}$ which leaves $1_\mathcal{G}$ fixed and transforms every other element into g. Since every operation on a finite perfect group is expressible by polynomial expressions, the composition of f_g with this operation is also quantifier-free and defines a many-one reduction from the word problem of $\mathcal{G}$ to L.

□

Proof of lemma 15. Since the syntactic monoid of $R_\mathcal{A}$ is finite, there is $t > 0$ such that $\eta_{R_\mathcal{A}}(\Sigma^t) = \eta_{R_\mathcal{A}}(\Sigma^{2t})$. Now add an internal letter μ to Σ, obtaining an alphabet Σ'_m, and pad all words of $R_\mathcal{A} \subseteq (\Sigma_m)^* \subseteq (\Sigma'_m)^*$ by appending $< t$ many copies of (μ, i), where i is chosen such that the labeling is valid, so that every word has a length dividable by t We view the resulting language R' as a regular language over the alphabet $\Xi := (\Sigma'_m)^t$. We intersect $R' \subseteq \Xi^*$ with a star-free language rejecting all words containing an invalid two-character subword in $\Xi^2 = (\Sigma'_m)^{2t}$, obtaining a regular language R''. The intention of this construction is to remove groups that cannot occur in valid inputs. Let M be the syntactic monoid of $R'' \subseteq \Xi^*$. Let $\mathcal{N}$ be a simple monoid dividing M. We want to show $\mathcal{N} \in Q_L$. Let $g_1, g_2 \in M$ be elements mapped to two generators of $\mathcal{N}$ by the morphism ϕ by which $\mathcal{N}$ divides M. Let $G_i := (\Sigma_m)^t \cap \eta_{R''}^{-1}(g_i) \subseteq \Xi$. We now want to show that there are words $\gamma_i \in G_i$ such that any product over them is valid. This will then show that $\mathcal{N}$ can be obtained using valid words, and therefore is in Q_L.

– *Case I*: Both G_i contain words in $\{\Sigma \times \{m\}\}^*$. Then any product of such words is valid.

– *Case II*: One G_i, say G_1, only contains words which contain some label $< m$. Let γ_1 be one such word, and let $\gamma_2 \in G_2$. We know that $\gamma_1\gamma_2$ and $\gamma_2\gamma_1$ are valid, since the congruence class of words containing an invalid two-character subword is a zero for M and cannot be contained in $\phi^{-1}(\mathcal{N})$. We now want to deduce that all products of γ_1, γ_2 must be valid. If both words start and end with a label $< m$, any product will be valid. Now assume γ_1 starts or ends with a symbol labeled m. Since $\gamma_1\gamma_1$ is valid and γ_1 contains a position labeled $< m$, we have $\Delta(\gamma_1) = 0$. Thus γ_2 starts and ends with a position labeled m, and all products are valid. Furthermore, if γ_1 starts and ends with a symbol labeled $< m$, then γ_2 also starts and ends with this position, and all products are valid.

In both cases, $\{\gamma_1, \gamma_2\}^* \subseteq F(\tau_m(\Sigma^*))$, and $\eta_{R'}(\{\gamma_1, \gamma_2\}^*) = \mathcal{N}$. But then $\eta_{R_\mathcal{A}}(\{\gamma_1, \gamma_2\}^*)$ must be divided by $\mathcal{N}$, since $\mathcal{N}$ is simple. So $\mathcal{N} \in Q_L$. Thus, R'' is definable in $FO + Q_L[<]$ over the alphabet Ξ by a result of [15]. Hence, it is definable in $FO + Q_L[Reg]$ over the alphabet Σ_m. It is clear that removing the additional copies of (μ, i) preserves definability in this logic. Since the R' and R'' agree on the valid words, we have obtained the desired formula ϕ if $U_1 \in Q_L$. □

Proof of Lemma 16. We have an $FO + Q_L[Reg]$ formula ϕ from Lemma 15. We replace the letter predicates $Q_{(a,k)}(x)$ by $(Q_a(x) \wedge H_k(x))$ when $k < m$, and by $(Q_a(x) \wedge H_{\geq m}(x))$ when $k = m$. The resulting formula ϕ' operates on words in Σ^*. The height predicates H_k and $H_{\geq m}$ can be computed for any word in Maj[<], and since regular predicates are definable in Maj[<], we obtain a $Maj + Q_L[<]$-formula for L. Now if L has simple height behavior, we have the $FO[+]$-definable height predicates from theorem 7. For $w \in L$, we have $w \models \phi'$ since L has simple height behavior. However, for words outside the language, there may be false positives in the $H_{\geq m}$ predicate. We refer to the proof of Theorem 27 in [10] for the argument that the precise definition of the height predicates ensures that false positives do not cause problems and the formula ϕ' defines L. □

Proof of Corollary 23. Given $\mathcal{A}$, one can compute an upper bound on the logic C using Lemma 16, since the groups needed for the formula in Lemma 15 can be computed by considering the syntactic monoid of the regular language constructed in proof of that lemma. Let C' be some other logic class that can define $L(\mathcal{A})$. Since $\mathcal{A}$ has simple height behavior, $C = FO + Q_L[SL]$. C' must contain all groups in Q_L by Lemma 13 and the mentioned result of [11]. □