Electronic Colloquium on Computational Complexity, Report No. 177 (2014)

Visibly Counter Languages and Constant Depth Circuits

Andreas Krebs, Klaus-Jörn Lange, and Michael Ludwig

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

Abstract

We examine visibly counter languages, which are languages recognized by visibly counter automata (a.k.a. input driven counter automata). We are able to effectively characterize the visibly counter languages in AC^0, and show that they are contained in FO[+].

Keywords and phrases visibly counter automata, constant depth circuits, AC0, FO[+]

1 Introduction

One important topic of complexity theory is the characterization of regular languages contained in constant depth complexity classes [6, 3, 5, 4]. In [4] Barrington et al. showed that the regular sets in AC^0 are exactly the languages definable by first order logic using regular predicates.

We extend this approach to certain non-regular languages. The *visibly pushdown languages* (VPL) are a sub-class of the context-free languages containing the regular sets which exhibits many of the decidability and closure properties of the regular languages. Their essential feature is that the use of the pushdown store is purely input-driven: The input alphabet is partitioned into call, return and internal letters. On a call letter, the pushdown automaton (PDA) must push a symbol onto its stack, on a return letter it must pop a symbol, and on an internal letter it cannot access the stack at all. This is a severe restriction, however it allows visibly pushdown automata (VPA) to accept non-regular languagesl, the simpelst simplest example being a^nb^n. At the same time, VPA are less powerful than all PDA: They even cannot check if a string has an equal number of a's and b's. In fact, due to the visible nature of the stack, membership testing for VPA might be easier than for general PDA. It is known to be in NC^1 [7] and hence it is NC^1-complete. On the other hand the membership problem for the context-free languages is complete for SAC^1 [19].

Visibly counter automata (VCA) [2] were introduced by Bárány et al. as a restricted model of visibly pushdown automata as they were of use to decide a certain sub-class membership problem of VPL. They still contain all regular sets.

In this paper, we show that all visible one-counter languages in AC^0 are definable by first order logic using addition as an numerical predicate. Our techniques allow us to decide whether a visible one-counter language is in fact a member of AC^0.

Examples of visible counter languages are:

- The set $\{a^nb^n \mid n \geq 0\}^*$, the Kleene-closure of $\{a^nb^n \mid n \geq 0\}$, is in FO[+] and hence in AC^0.
- The one-sided Dyck language of a single pair of parentheses. This language is not in AC^0.
- The set $\{\{a, aba\}^nb^n \mid n \geq 0\}$ is as hard as the set EQUALITY of all strings in $\{0, 1\}^*$ in which the numbers of ones coincide with those of zeroes and is thus not in AC^0.

ISSN 1433-8092

Another interest stems from descriptive complexity [12]. Here we use the model of predicate logic for language recognition where variables are associated with word positions. An interesting question is whether one can extend the conjecture of Straubing that for any set of quantifiers Q it suffices to consider regular predicates, i.e., $Q[arb] \cap \mathrm{REG} = Q[Reg]$, to a family of non-regular lanugages. This question was examined in detail in [14]. Here we show $\mathrm{FO}[arb] \cap \mathrm{VCL} \subseteq \mathrm{FO}[+]$, which shows that only the addition predicate is needed.

The rough idea of our proof is to exhibit two decidable properties of a visible counter automaton $\mathcal{A}$ which together characterize the property of the accepted language $L(\mathcal{A})$:

- The first property concers the "height behavior" of words. E.g. the Dyck set contains all possible height progressions and is not in AC^0. However the language $L = \{a^n b^n \mid n \geq 0\}$ has a very simple height behavior. The language L^* is more complicated in this respect but is still in AC^0. We introduce the notion of *simple height behavior* of a language to capture this. If $L(\mathcal{A})$ has simple height behavior, then a matching predicate is definable in AC^0. If not, then $L(\mathcal{A})$ is not in AC^0.
- The second property is a modification of quasi-aperiodicy (see [4, 18]) of regular languages fit for our needs. If $L(\mathcal{A})$ does not have the property we can reduce a language outside AC^0 to it. Otherwise we get a certain $\mathrm{FO}[Reg]$ formula.

If $L(\mathcal{A})$ has the two properties then by using the matching predicate and the $\mathrm{FO}[Reg]$ formula, we can build a $\mathrm{FO}[+]$ formula for $L(\mathcal{A})$.

We thank the anonymous referees for their helpfull feedback.

2 Preliminaries

By $\mathbb{Z}$ we denote the integers, by $\mathbb{N}$ the non-negative integers and by $\mathbb{Q}$ the rational numbers. An *alphabet* is a finite set Σ and ϵ is the empty word. A *language* is a subset of Σ^*. For a word w, $|w|$ is the length of the word and $|w|_M$ for $M \subseteq \Sigma$ is the number of letters in w which belong to M. If not locally defined otherwise, w_i is the letter in position i in w. For a language $L \subseteq \Sigma^*$, the set $F(L) \subseteq \Sigma^*$ is the set of all *factors* of words in L.

For every language $L \subseteq \Sigma^*$ we define a congruence relation, the *syntactic congruence* of L: For $x, y \in \Sigma^*$ it is $x \sim_L y$ iff for all $u, v \in \Sigma^*$ we have $uxv \in L \Leftrightarrow uyv \in L$. The *syntactic monoid* is $\mathrm{syn}(L)$, i.e. the set of equivalence classes under $\sim_L$ with the induced multiplication. The *syntactic morphism* of L is $\eta_L \colon \Sigma^* \to \mathrm{syn}(L)$.

We use circuits as a model of computation. Important complexity classes in this area include:

- AC^0 - polynomial-size circuits of constant depth with Boolean gates of arbitrary fan-in.
- ACC^0_k - AC^0 circuits plus modulo-k-gates. ACC^0 is the union of ACC^0_k for all k.
- TC^0 - polynomial-size circuits of constant depth with threshold gates of arbitrary fan-in.
- NC^1 - polynomial-size circuits of logarithmic depth with bounded fan-in.

Circuits have a vertain imput length. However it is desirable to be able to treat arbitrary long inputs. This is achieved with families of circuits which contain one circuit for each input length. 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. Consult e.g. [20] for further references on circuit complexity.

We also use the model of first-order predicate logic over finite words. Variables range over word positions. and so the numerical predicates are sets of word positions. E.g. $<$ is a predicate of arity two with obvious semantic. We allow for existential and all quantification: $\exists, \forall$. We write $\mathrm{FO}[<]$ for the set of languages we get of first-order formulas with $<$ predicate. The $+$ predicate has arity three: $(i, j, k) \in +$ iff $i + j = k$.

The class (of non-uniform) AC^0 coincides with first order logic with arbitrary numerical predicates, which we denote by $\mathrm{FO}[arb]$ [11, 9]. For the interplay of circuits and logic see [18]. A prominent theorem is the equivalence of star-free languages, $\mathrm{FO}[<]$ languages and languages with aperiodic syntactic monoid [17, 15]. A monoid M is aperiodic if for all $m \in M$ it holds that $m^i = m^{i+1}$ for some i or equivalently if no subset of M is a non-trivial group. For us a related notion is also important: The intersection of AC^0 and the regular languages is captured exactly by the set of languages which have a quasi-aperiodic syntactic morphism η_L. It is quasi-aperiodic if for all $t > 0$, $\eta_L(\Sigma^t)$ does not contain a non-trivial group. Languages with quasi-aperiodic syntactic morphisms are exactly the ones in $\mathrm{FO}[Reg]$, that is first order logic using the regular predicates, which are the order predicate and the modulo predicates [4].

We will use the following languages:

- $\textsc{Equality} = \{w \in \{0,1\}^* \,:\, |w|_0 = |w|_1\}$ is TC^0-hard.
- $\textsc{Mod}_k = \{w \in \{0,1\}^* \,:\, |w|_0 \equiv 0 \pmod k\}$ is ACC^0_k-hard.

Neither $\textsc{Equality}$ nor $\textsc{Parity} = \textsc{Mod}_2$ is in AC^0 [8, 10].

Mehlhorn [16] and independently also Alur and Madhusudan [1] introduced *input-driven* or *visibly pushdown automata*. Here, the input symbol determines the stack operation, i.e. if a symbol is pushed or popped. This leads to a partition of Σ into call, return and internal letters: $\Sigma = \Sigma_{\text{call}} \cup \Sigma_{\text{ret}} \cup \Sigma_{\text{int}}$. Then $\hat{\Sigma} = (\Sigma_{\text{call}}, \Sigma_{\text{ret}}, \Sigma_{\text{int}})$ is a visibly alphabet. In the rest of the paper we always assume that there is a visibly alphabet for Σ.

We define a function $\Delta : \Sigma^* \to \mathbb{Z}$ which gives us the *height* of a word by $\Delta(w) = |w|_{\Sigma_{\text{call}}} - |w|_{\Sigma_{\text{ret}}}$. Each word w over a visibly alphabet can be assigned its *height profile* w^Δ, which is a map $\{0,\ldots,|w|\} \to \mathbb{Z}$ with $w^\Delta(i) = \Delta(w_1 \cdots w_i)$. A word w is *well-matched* if w^Δ maps into $\mathbb{N}$ and $\Delta(w) = 0$. Two positions i, j of a word are *matched* if $w_i \in \Sigma_{\text{call}}$, $w_j \in \Sigma_{\text{ret}}$, and $w_{i+1} \ldots w_{j-1}$ is well-matched. In a well-matched word, every position i has a matching position j, unless w_i is an internal letter. Thus, positions with letters in Σ_{int} are always unmatched. We say a word w has a *non-negative height profile* if $\Delta(w_1 \cdots w_i) \geq 0$ for all $i \in \{0,\ldots,|w|\}$.

Bárány et al. [2] introduced the notion of *visibly counter automata* (VCA). Since every VPA can be determinized and this is also true for visibly counter automata, we restrict ourselves 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 $\delta_i \colon Q \times \Sigma \to Q$ is the transition functions.

A configuration is an element of $Q \times \mathbb{N}$. Note that m−VCAs, similar to VPAs, can only recognize words where the height profile is non-negative. All other words are rejected. An m−VCA $\mathcal{A}$ performs the following transition when a letter $\sigma \in \Sigma$ is read: $(q,k) \xrightarrow{\sigma} (\delta_{\min(m,k)}(q,\sigma), k + \Delta(\sigma))$. Then $w \in L(\mathcal{A})$ iff $(q_0, 0) \xrightarrow{w} (f, h)$ for $f \in F$ and $h \geq 0$. Note that we slightly modified the semantics compared to [2], as they required $h = 0$ for a word to be accepted. Our version is more general since we can accept languages which contain words w with $\Delta(w) > 0$. Bárány et al. needed that their 0−VCA only accepts languages of well-matched words. With our definition we would need an 2−VCA so simulate this.

▶ **Example 2.** Consider the language $L = \{a^n b^n \mid n \in \mathbb{N}\}$ where a is a push letter and b is a pop letter, so $\hat{\Sigma} = (\{a\}, \{b\}, \emptyset)$. Then a 2−VCA for L is $\mathcal{A} = (Q, q_0, F, \hat{\Sigma}, \delta_0, \delta_1, \delta_2)$ with $Q = \{q_0, f, q, q_{\text{dead}}\}$, $F = \{f\}$ and:

- $\delta_i(q_0, a) = q_0$ for $0 \leq i \leq 2$

- $\delta_0(q_0, b) = q_{\text{dead}}$, $\delta_1(q_0, b) = f$, $\delta_2(q_0, b) = q$
- $\delta_0(q, b) = q_{\text{dead}}$, $\delta_1(q, a) = f$, $\delta_2(q, b) = q$
- $\delta_i(q, a) = q_{\text{dead}}$ for $0 \le i \le 2$
- $\delta_i(q_{\text{dead}}, \sigma) = \delta_i(f, \sigma) = q_{\text{dead}}$ for $0 \le i \le 2$ and $\sigma \in \{a, b\}$

We say a word w loops through q if $(q, h) \xrightarrow{w} (q, h + \Delta(w))$ and $\Delta(w_1 \cdots w_i) + h > m$ for all positions i.

▶ **Definition 3** (VCL)**.** The class of the visibly counter languages (VCL) is that of languages recognized by a $m-$VCA for some m.

3 Properties of Visibly Counter Languages

We will present two properties which a VCL language L must fulfill to be in AC^0. The first property concerns the behavior of the height profiles of the words in L. Intuitively, we need to be able to compute most of the height profile in AC^0. The second property assumes that the height profile is known and is about computing the states which the automaton will pass on its run on the input. This is closely related to the property a regular language must have to be in AC^0.

In the following we will decompose the automaton so that we can handle the two properties independently. After that we treat the two properties and show that L is not in AC^0, if one of the properties is violated. On the other hand we are able to build a FO[+] formula in case L has these two properties.

3.1 Decomposition of the Automaton

We will split the computation of the automaton in two steps. The first part is the computation of the height profile. The second can be seen as the regular part of the language. Formally, we will extend the alphabet to include the stack-height up to a threshold. We will consider a new language of words over the extended alphabet. This language will be regular since the information for the decision which δ_i to use is already coded into the input.

Similar to a regular transducer we define a transduction that appends the height profile to a given word. In the following we fix a visibly alphabet $\hat{\Sigma}$ of Σ.

▶ **Definition 4** (Height transduction)**.** We let $\tau_m : \Sigma^* \to \Sigma_m^*$ where $\Sigma_m = \Sigma \times \{0, \dots, m\}$. With τ_m we assign to each position in the word its height up to the threshold m.

$$\begin{aligned}&\tau_m(w) = \tau_m(w_1 w_2 \cdots w_n) = \\ &(w_1, \Delta_m(\epsilon))(w_2, \Delta_m(w_1)) \cdots (w_i, \Delta_m(w_1 \cdots w_{i-1})) \cdots (w_n, \Delta_m(w_1 \cdots w_{n-1}))\end{aligned}$$

where $\Delta_m(w) = \min(\Delta(w), m)$. The transduction τ_m is only defined on words with a non-negative height profile. We call a word in Σ_m^* *valid* if it is in $F(\tau_m(\Sigma^*))$. We also say that i is the *label* of the letter $(a, i) \in \Sigma_m$.

▶ **Example 5.** If a is a push letter, and b a pop letter, then $\tau_2(aaba) = (a, 0)(a, 1)(b, 2)(a, 1)$.

▶ **Definition 6.** For an $m-$VCA $\mathcal{A} = (Q, q_0, F, \hat{\Sigma}, \delta_0, \dots, \delta_m)$ we define $R_{\mathcal{A}} = L(M)$ where M is a finite automaton $M = (Q, q_0, F, \Sigma_m, \delta)$, where $\delta(q, (a, i)) = \delta_i(q, a)$.

The following statement is obvious:

▶ **Lemma 7.** *If $\mathcal{A}$ is an $m-$VCA, then $w \in L(\mathcal{A})$ iff $\tau_m(w) \in R_{\mathcal{A}}$.*

3.2 Height Computation

In this section we investigate in which cases the transduction τ_m is expressible as a FO[+] formula.

For this we need to be able to count the number of call letters minus the number of return letters in the prefix, which is in general TC^0-hard. Yet, if all the states that are important to the height computation i.e. can occur in loops that have a "fixed slope", then the computation will be in AC^0.

We fix some $m-$VCA $\mathcal{A}$.

▶ **Definition 8** (fixed slope)**.** 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

We call α the *slope* and γ the *corridor* of q.

Figure 1 shows the concept of this definition.

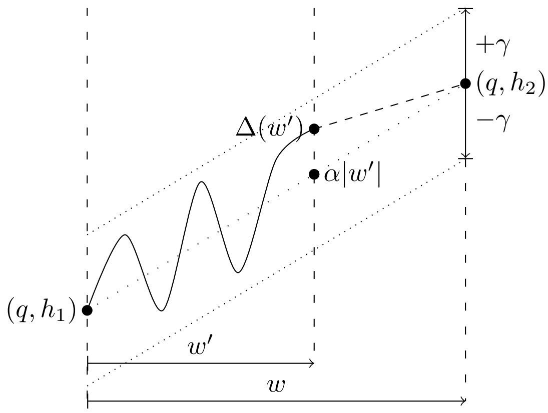

Figure 1 Visualization of a state q having fixed slope where α is the actual slope and γ is the corridor. If w' is a prefix of w then $\Delta(w')$ has to stay in the corridor.

As we will see, we can think of states with a fixed slope as of those which do not pose a problem when computing the stack height in FO[+]. However there can be states without a fixed slope, which do not make the language too hard for FO[+], since it is possible that the recognition of a word does not depend on its height profile any more if $\mathcal{A}$ has visited such a state. This happens if from this point the height of the word can never reach height levels below m any more. The next definition captures this idea by some kind of reachability property. Figure 2 visualizes the idea.

▶ **Definition 9** (active)**.** 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 $\Delta(w_1 \cdots w_i) - \Delta(w_1 \cdots w_j) > |Q|$.

Before we prove that for a VCL language L every active state needs to have a fixed slope for L to be in AC^0, we give an example of a typical case of a hard language.

▶ **Example 10.** Consider the language $L = \{(a|aba)^n b^n \mid n \in \mathbb{N}\}$. This language is clearly in VCL but there is not an $m-$VCA for L where every active state has a fixed slope. In fact

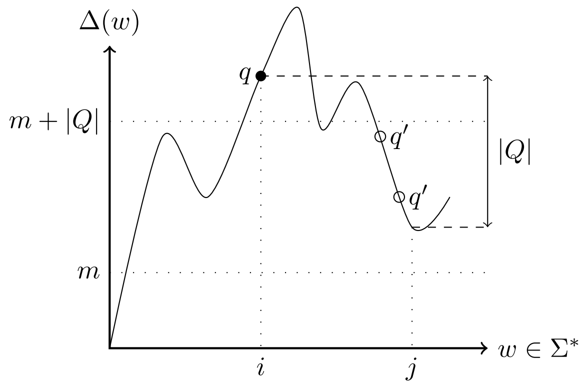

Figure 2 The state q is an active state since there is a word $w \in L$ such that q occurs at position i with a height above $m+|Q|$ and there is a word position j with a height difference of $|Q|$ compared to i. By this we know that there is a down loop - in this case through state q'. This is used in lemma 11.

L is not in AC^0 because of this. We can reduce the TC^0-hard language EQUALITY $\subseteq \{0,1\}^*$ to L. Let ϕ and ψ be morphisms with $\phi(0) = aaa$, $\phi(1) = aba$ and $\psi(0) = \psi(1) = bb$. The reduction is $f(w) = \phi(w)\psi(w)$ which is in AC^0. As one can see, the number of push letters a and pop letters b is in balance iff there are as many 0's as 1's: $|f(w)|_a = 3|w|_0 + 2|w|_1 = |f(w)|_b = |w|_1 + 2|w|$. This is equivalent to $|w|_0 = |w|_1$.

In the following lemma and its proof, we generalize the idea of the previous example.

▶ **Lemma 11.** *If a language $L \in$ VCL is recognized by an $m-$VCA which has an active state without a fixed slope, then L is not in* AC^0.

Proof. Let $\mathcal{A}$ be an $m-$VCA with $L = L(\mathcal{A})$ having a state q which is active but does not have a fixed slope. This implies that there are words besides the empty word forming a loop through q. In fact, there must be two words u and v which loop through q with $|u| = |v|$ and $\Delta(u) > \Delta(v)$. Also there must exist a word $w \in L$ with certain properties: intuitively w has a high "mountain", and thus there must be loops going up and down the "mountain" and one loop goes through q. In the following we assume q to be responsible for an up loop. The case where q is responsible for the down loop can be treated similarly. To be precise, w has the following properties:

- Using position i from the definition of *active*, in the run q appears in position i and w has a height above $m + |Q|$ in i.
- Because of the existence of position j with smaller height (a difference of at least $|Q|+1$), there must be a down loop. Let q' be a state looped when going down.
- We can partition w into $w = \alpha\beta\gamma$ with $\alpha = w_1 \cdots w_i$ and q' is reached after $\alpha\beta$ the first time, i.e. never in between α and $\alpha\beta$.
- There is a word $x \in \Sigma^*$ looping through q'. We assume that $-\Delta(x) > \Delta(\alpha\beta)$, which is equivalent to $\Delta(\alpha\beta x) < 0$.

It is important to note that between α and $\alpha\beta$ the height never falls below m.

In the following we want to reduce the language EQUALITY $\subseteq \{0,1\}^*$, which is in TC^0 but not in AC^0, to L by using the following pumping approach:

$$\alpha u^{-k\Delta(x)} \beta x^{k\Delta(u)} \gamma \in L$$

for all $k \geq 0$. This is true, since $\Delta(u^{-k\Delta(x)}) = -\Delta(x^{k\Delta(u)})$

We define the following words:

- $u' = u^{-2\Delta(x)}$
- $v' = v^{-2\Delta(x)}$
- $x' = x^{\Delta(u)+\Delta(v)}$

The key property is $\Delta(u'u') > \Delta(u'v') = -\Delta(x'x') > \Delta(v'v')$ and $|u'| = |v'|$. We define the morphisms $\phi, \psi \colon \{0,1\}^* \to \Sigma^*$ with $\phi(0) = u'$, $\phi(1) = v'$ and $\psi(0) = \psi(1) = x'$ and the map $f \colon w \mapsto \alpha\phi(w)\beta\psi(w)\gamma$. Since for all morphisms we used it holds that for two words of same length, the images have the same length, f is computable in AC^0.

For $w \in \{0,1\}^*$, let $\bar{w}$ be the word, where 0 and 1 are switched, e.g. $w = 0100$, then $\bar{w} = 1011$. We now have $w \in$ Equality $\Leftrightarrow f(w) \in L \wedge f(\bar{w}) \in L$. This is true since if w has more 0's than 1's (or vice versa) then either $\Delta(f(w))$ or $\Delta(f(\bar{w}))$ is negative (which is ensured by the condition $-\Delta(x) \geq \Delta(\alpha\beta)$ we had on x) and such words cannot be accepted by some Vca $\mathcal{A}$. So f is in fact a reduction and hence $L \notin \mathrm{AC}^0$. ◀

In the proof of the previous lemma, we saw that we get a property of the accepted language. If we have two automata for some language and one of them has an active state without a fixed slope and the other one does not then we get a contradiction using a pumping argument.

▶ **Corollary 12.** *If for some $m-$VCA $\mathcal{A}$ every active state has a fixed slope then in all* VCA *for $L(\mathcal{A})$ the active states have fixed slopes.*

This corollary justifies to formulate a property of languages:

▶ **Definition 13** (simple height behaviour)**.** If in some VCA all active states have a fixed slope, we say that the recognized language has simple height behaviour.

Figure 3 shows situations being relevant for this property.

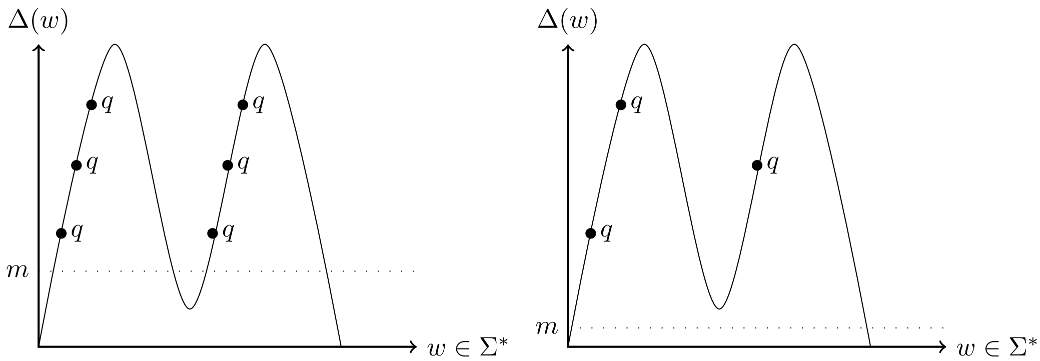

Figure 3 The left example shows an active state which might have a fixed slope (at least the pictured situation is no counter example). Through q we get an up loop and after the word has reached a height level below m, q can be reached again. In the right example however we see that q is active and does not have a fixed slope.

We now assume $L(\mathcal{A})$ has simple height behavior. In this case we can compute a sufficient approximation of the matching predicate in FO[+] and in turn use this predicate to define a stack height predicate.

We would like to define the matching predicate in FO[+] that is true for all words w with two positions i, j that are matching positions, i.e. $\Sigma_{\mathrm{call}}(w_i)$ and $\Sigma_{\mathrm{ret}}(w_j)$ and $w_{i+1} \dots w_{j-1}$

is well-matched. Even if a language L is in FO[+], the matching predicate is not necessarily in FO[+]. Hence we only approximate the matching predicate from below, i.e. we only have false negatives and recognize all matching pairs of positions that are needed later.

First, we need to define some helper predicates that allow us to verify that the height profile of some factor $w_{i+1} \dots w_j$ has a slope α and stays within a corridor $\pm\gamma$ around this slope and the height profile is above some minimal value h_l.

▶ **Definition 14.** For every $\alpha \in \mathbb{Q}$ and $\gamma \in \mathbb{N}$ we define a 5-ary predicate $B_{\alpha,\gamma}(x, y, s, t, l)$ such that: $w_{x=i,y=j,s=h_s,t=h_t,l=h_l} \models B_{\alpha,\gamma}(x, y, s, t, l)$ iff

- $\Delta(w_{i+1} \dots w_j) = h_t - h_s$,
- for all $i < k \leq j$ we have $\Delta(w_{i+1} \dots w_k) > h_l - h_s$,
- for all $i < k \leq j$ we have $\alpha|w_{i+1} \dots w_k| - \gamma \leq \Delta(w_{i+1} \dots w_k) \leq \alpha|w_{i+1} \dots w_k| + \gamma$, and
- $\Delta(w_{i+1} \dots w_j) = \alpha|w_{i+1} \dots w_j|$.

▶ **Lemma 15.** *For any $\alpha \in \mathbb{Q}$ and $\gamma \in \mathbb{N}$, the predicate $B_{\alpha,\gamma}(x, y, s, t, l)$ can be defined in* FO[+].

In order to prove that the predicate $B_{\alpha,\gamma}(x, y, s, t, l)$ can be defined in FO[+], we will define a more technical predicate. The following predicate defines a corridor with slope p/q and verifies if y points to a position after x that leaves the corridor to the top under the promise we are inside the interval before. In order to use induction we need to have different borders γ_1, γ_2 to the top and bottom. To gain intuition it helps to think of the case that the slope is 0 in all cases first, and then see that the other cases just work relative to this slope.

▶ **Definition 16.** For every $p \in \mathbb{Z}, q \in \mathbb{N}$ with p, q relative prime, $\gamma_1, \gamma_2 \in \mathbb{N}$, and $\delta \in \mathbb{Z}$ we define a binary predicate $C_{p,q,\gamma_1,\gamma_2,\delta}(x, y)$ such that: $w_{x=i,y=j} \models C_{p,q,\gamma_1,\gamma_2,\delta}(x, y)$ iff

- $i \equiv j \pmod q$,
- for all $i < k < j$ with $k \equiv i \pmod q$ we have $\alpha|w_{i+1} \dots w_k| - \gamma_1 \leq \Delta(w_{i+1} \dots w_k) \leq \alpha|w_{i+1} \dots w_k| + \gamma_2$, and
- $\Delta(w_{i+1} \dots w_j) = \alpha|w_{i+1} \dots w_j| + \delta$.

▶ **Lemma 17.** *The binary predicates $C_{p,q,0,0,\delta}(x, y)$ can be defined in* FO[+].

Proof. For $w_{x=i,y=j} \models C_{p,q,0,0,\delta}(x, y)$, it suffices test if $i \equiv j \pmod q$, for each $i < k < j$ with $k \equiv i \pmod q$ if $\alpha|w_{k-q+1} \dots w_k| = \Delta(w_{k-q+1} \dots w_k)$, and if $\alpha|w_{j-q+1} \dots w_j| + \delta = \Delta(w_{j-q+1} \dots w_j)$. Note that the second and third test only involves testing a factor word of length q, which is constant length. Hence, the predicates $C_{p,q,0,0,\delta}(x, y)$ can be defined in FO[+]. ◀

▶ **Lemma 18.** *The binary predicates $C_{p,q,\gamma_1,\gamma_2,\delta}(x, y)$ can be defined in* FO[+].

Proof. We prove this by induction on $\gamma_1 + \gamma_2$, where the base case was already handled by the previous lemma.

We need to define a FO[+] formula to test $w_{x=i,y=j} \models C_{p,q,\gamma_1,\gamma_2,\delta}(x, y)$.

There are three cases to consider:

- We never reach the top/bottom border, i.e. either for all $i < k < j$ with $k \equiv i \pmod q$ we have $\alpha|w_{i+1} \dots w_k| - \gamma_1 \leq \Delta(w_{i+1} \dots w_k) \leq \alpha|w_{i+1} \dots w_k| + \gamma_2$-1, or for all $i < k < j$ with $k \equiv i \pmod q$ we have $\alpha|w_{i+1} \dots w_k| - \gamma_1 + 1 \leq \Delta(w_{i+1} \dots w_k) \leq \alpha|w_{i+1} \dots w_k| + \gamma_2$,
- we leave the corridor, i.e. exists a $i < k < j$ with $k \equiv i \pmod q$ and $\alpha|w_{i+1} \dots w_k| - \gamma_1 \leq \Delta(w_{i+1} \dots w_k) \leq \alpha|w_{i+1} \dots w_k| + \gamma_2$ fails.
- we reach both top and bottom border, i.e. there are $i < k < j$ and $i < l < j$ with $\Delta(w_{i+1} \dots w_k) = \alpha|w_{i+1} \dots w_k| + \gamma_2$ and $\alpha|w_{i+1} \dots w_l| - \gamma_1 = \Delta(w_{i+1} \dots w_l)$,

The first case is covered by induction, we can verify it via $C_{p,q,\gamma_1-1,\gamma_2-1,\delta}(x,y)$.

For the second case either run over the top border never reaching the bottom border, i.e. $\bigvee_{a=1}^{q} C_{p,q,\gamma_1-1,\gamma_2,\gamma_2+a}$, or we reach the bottom border and then go more than $\gamma_1+\gamma_2$ steps to the top. More formally we find $i \leq i' < j' \leq j$ with $\Delta(w_{i'+1}\dots w'_j) > \alpha|w_{i'+1}\dots w'_j| + (\gamma_1+\gamma_2)$. Since we can choose i' maximal we can assume that $\Delta(w_{i'+1}\dots w_{i'+q}) > \alpha|w_{i'+1}\dots w_{i'+q}| + 1$ and $\Delta(w_{i'+q+1}\dots w_j) > \alpha|w_{i'+q+1}\dots w_j| + (\gamma_1+\gamma_2-1)$. Hence we can verify this with:

$$\begin{aligned}\exists \quad & x'\ x < x' < y \wedge x' \equiv x \pmod q \wedge \exists y'\ x' < y' < y \wedge y' \equiv x \pmod q \\ & C_{p,q,0,0,1}(x'+q,y') \wedge \bigvee_{a=1}^{q} C_{p,q,0,\gamma_1+\gamma_2-1,\gamma_1+\gamma_2-1+a}(x'+q,y')\end{aligned}$$

Similar we can test if we run below the bottom border.

For the third case we assume we stay inside the interval. This case works similar to the previous case, we find the last two positions k,l before j where we reach the top and bottom border. Assume that $l<k$. We need to verify that $\Delta(w_{l+1}\dots w_k) = \alpha|w_{l+1}\dots w_k| + (\gamma_1+\gamma_2)$, and $\Delta(w_{k+1}\dots w_j) = \alpha|w_{k+1}\dots w_j| + (\gamma_1-\delta)$.

$$\begin{aligned}\exists \quad & x'\ x < x' < y \wedge x' \equiv x \pmod q \wedge \exists y'\ x' < y' < y \wedge y' \equiv x \pmod q \\ & C_{p,q,0,0,1}(x'+q,y') \wedge C_{p,q,0,\gamma_1+\gamma_2-1,\gamma_1+\gamma_2-1}(x'+q,y') \wedge C_{p,q,\gamma_1+\gamma_2-1,0,\gamma_1-\delta}\end{aligned}$$

Combining these three cases gives us the desired formula. ◀

Proof of lemma 15. Let p,q be relative prime numbers such that $p/q=\alpha$. Lets recall the definition of $w_{x=i,y=j,s=h_s,t=h_t,l=h_l} \models B_{\alpha,\gamma}(x,y,s,t,l)$. We have four conditions that need to be verified.

- $\Delta(w_{i+1}\dots w_j) = h_t - h_s$,
- for all $i<k\leq j$ we have $\Delta(w_{i+1}\dots w_k) > h_l - h_s$,
- for all $i<k\leq j$ we have $\alpha|w_{i+1}\dots w_k| + \gamma \leq \Delta(w_{i+1}\dots w_k) \leq \alpha|w_{i+1}\dots w_k| + \gamma$, and
- $\Delta(w_{i+1}\dots w_j) = \alpha|w_{i+1}\dots w_j|$.

Using the binary predicates C we can verify condition three for all positions $k \equiv i \pmod q$. Checking locally the letters at the next $q-1$ positions we can verify condition three for all k. Similar we verify condition two.

Let $k<j$ be the largest number with $k \equiv i \pmod q$. Using the predicate C we can compute $\Delta(w_{i+1}\dots w_k)$ in FO[+]. Since $w_{k+1}\dots w_j$ is always of length less than q we can also compute $\Delta(w_{k+1}\dots w_j)$. Hence condition one and four can be verified in FO[+]. ◀

▶ **Lemma 19.** *Given an $m-$VCA $\mathcal{A}$, such that $L=L(\mathcal{A})$ has simple height behavior, we can define a binary predicate M in FO[+] such that for every $w\in\Sigma^*$ and positions i,j of w:*

- *$w_{x=i,y=j} \models M(x,y)$ implies that the position i matches the position j in w.*
- *If $w\in L$ and there is a $k>i$ with $\Delta(w_1\dots w_k)\leq m$ and the position i matches the position j then $w_{x=i,y=j}\models M(x,y)$.*

We will first define such a predicate for positions i,j that both have stack height larger than $m+|Q|$ and then define it inductively for smaller stack heights.

Fix a word w and i,j. The question is, how to verify that i and j are matching positions. To do so, we need to verify that w_i is a push letter, w_j is a pop letter and the word $z=w_{i+1}\dots w_{j-1}$ is well-matched.

For the intuition we first consider a simple case. Assume that $z \in (\Sigma_{\text{call}}^* \Sigma_{\text{ret}}^*)^k$, then we could guess the $2k-1$ positions $x_1, \dots, x_{2k-1}$ where we switch between push and pop letters. We would verify that we push more on the stack than pop for every prefix of z, i.e. $x_1 - (x_2 - x_1) \geq 0$, $x_1 - (x_2 - x_1) + (x_3 - x_2) - (x_4 - x_3) \geq 0$, $\dots$ Finally we need to test if the sum of the length of the intervals with push letters is equal to the sum of the length of the intervals with pop letters.

Unfortunately we cannot assume that there is a constant k such that all words z are of this form. But we have a similar form for each factor z where we need to test if it is a well-matched word if the whole word w belongs to L. Assume here that w is well-matched. Since $w \in L$ there is an accepting run for w, hence every state occurring in an interval of height at least $m + |Q|$ is an active state, and every active state has a fixed slope. Let q be an active state that appears more than once in the run of w inside of z. Let k, l be the first and last position inside z where the state q occurs. Then the height difference $\Delta(z_k \dots z_l) = \alpha_q \cdot (l - k)$, where α_q is the slope of the state q. Since there are only finitely many states we can split z into a fixed number of intervals such that in each interval the stack height is "nearly linear" increasing or decreasing or being constant. If we cannot find such a fixed number of intervals than w cannot be in L.

The following lemma will formalize this idea.

▶ **Lemma 20.** *Given a language $L \subseteq \Sigma^*$ in* VCL *with simple height behavior, we can define a binary predicate $M_{>m+|Q|}$ in* FO[+] *such that for every $w \in \Sigma^*$ and positions i, j of w:*

- *$w_{x=i,y=j} \models M_{>m+|Q|}(x, y)$ implies the position i matches the position j in w.*
- *If $w \in L$ and $\Delta(w_1 \dots w_i) > m + |Q|$ and there is a $k > i$ with $\Delta(w_1 \dots w_k) \leq m$ and the position i matches the position j then $w_{x=i,y=j} \models M_{>m+|Q|}(x, y)$.*

Proof. We will first give the intuition on how to define the predicate $M_{>m+|Q|}$. Then we will show if $M_{>m+|Q|}$ is true that the positions i, j are actually matching positions, and finally that for $w \in L$ and i, j matching positions with stack height at least m, the predicate is true.

Let L be accepted by some $m-$VCA $\mathcal{A}$. Following the idea above our formula will need to guess at most $n = |Q| + 1$ points $z_0, \dots, z_n$ and the "slope" between these points represented by a state $q_1, \dots, q_n$. Finally we guess the stack-height $h_0, \dots, h_n$ at the points $z_0, \dots, z_n$ relative to $\Delta(w_1 \dots w_{i-1})$.

$$\begin{aligned} M_{>m+|Q|}(x, y) = \quad & x < y \wedge \Sigma_{\text{call}}(x) \wedge \quad \Sigma_{\text{ret}}(y) \wedge \\ & \bigvee_{(q_1, \dots, q_n) \in Q^n} \quad \exists z_0 \dots \exists z_n \exists h_1 \dots \exists h_n \\ & \qquad z_0 = x \wedge z_n = y - 1 \wedge h_0 = h_n \\ & \qquad \bigwedge_{i=0}^{n-1} z_i \leq z_{i+1} \wedge A_{q_{i+1}}(z_i, z_{i+1}, h_i, h_{i+1}, h_0) \end{aligned}$$

where the formulas A are defined below.

The idea is that the formula $A_{q_{i+1}}$ needs to verify that the guess was correct in the sense that the slope in interval $z_i + 1 \dots z_{i+1}$ is equal to the slope of q_{i+1} having a stack height difference of $h_{i+1} - h_i$. Note that we do not have to guess the state of the accepting run, but only *some* state with the same slope. In the case of an interval length 0 or 1 the formula $A_{q_{i+1}}$ will ignore the state q_{i+1} and directly check if the height difference is zero respectively corresponds to the single letter. See figure 4 as a sketch.

Finally we define the formula $A_{q_{i+1}}(z_i, z_{i+1}, h_i, h_{i+1}, h_0)$:

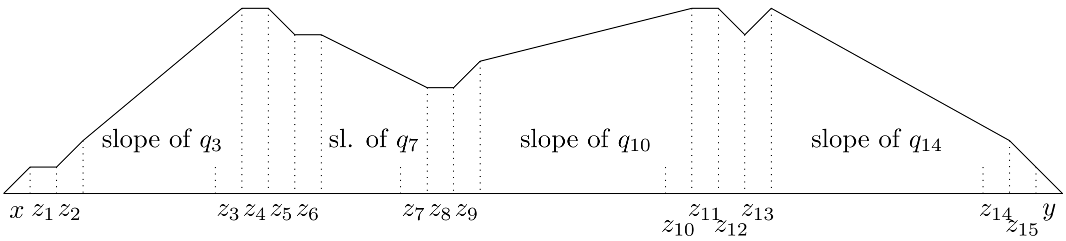

Figure 4 Example for positions of $z_1, \ldots, z_{15}$ fitting to the input word.

- If $z_i = z_{i+1} \wedge h_i = h_{i+1} \wedge h_i > h_0$ then the formula is true.
- If $z_i + 1 = z_{i+1}$ the formula is true if $h_i = h_{i+1} - 1 \wedge h_i > h_0$ (resp. $h_i = h_{i+1} \wedge h_i > h_0$ or $h_i = h_{i+1} + 1 \wedge h_{i+1} > h_0$) and $\Sigma_{\text{call}}(z_{i+1})$ (resp. $\Sigma_{\text{int}}(z_{i+1})$ or $\Sigma_{\text{ret}}(z_{i+1})$).
- In the case that $z_i + 1 < z_{i+1}$ we use the predicate $B_{\alpha,\gamma}(z_i, z_{i+1}, h_i, h_{i+1}, h_0)$ where α is the slope and γ the corridor of q.
- Otherwise the predicate is false.

Finally, we need to verify that with our definition of $M_{>m+|Q|}$ we satisfy the conditions of the lemma. The first condition is certainly true as guessing and verifying the stack height always is correct if all A predicates are true. For the second condition we need to show that is satisfies to guess n "turning points". We only consider the case $w \in L$, hence there is an accepting run of w and the sequence of states within the positions of x and y since $w \in L$ and the height profile will be below m at some point in the suffix all states of the accepting run are active states and hence have fixed slope. If the distance of x and y is less than n we could simple guess all states in this sequence. But their distance might be larger, hence we compress this sequence. For a state with a fixed slope the whole interval between the first and last occurrence should have a fixed slope and hence can be recognized by a single A predicate. So in the compressed sequence states with a fixed slope will occur only once. Hence it satisfies to guess n "turning points". ◀

At this point we have defined the predicate $M_{>m+|Q|}$. We can now define predicates M_k for height k under the assumption we have defined M_{k+1} already. This way we inductively get M_0.

▶ **Example 21.** Consider $L = \{a(a^n b^n)^* b \mid n \in \mathbb{N}\}$. If $w \in L$ then the first and the last letter of w match but the number of intervals can be arbitrary large. There is a $2-\text{VCA}$ for L but no $1-\text{VCA}$. This reflects in the matching predicates.

Proof of Lemma 19. By the previous lemma we have a predicate $M_{>m+|Q|}$. Fix a word w. Any two positions i, j are matching positions if and only if $w_i \in \Sigma_{\text{call}}$, $w_j \in \Sigma_{\text{ret}}$, and every push letter w_s with $i < s < j$ is matched to a pop letter w_t with $i < s < t < j$. Note that if i, j are at stack height h then s, t will be always at stack height $> h$, hence we can test if i, j are matched testing matching in between only for words of larger stack height.

$$
\begin{array}{rcl}
M_k(x,y) & = & M_{k+1}(x,y) \\
 & \vee & x < y \wedge \Sigma_{\text{call}}(x) \wedge \Sigma_{\text{ret}}(y) \\
 & \wedge & \forall z(x < z < y \wedge \neg\Sigma_{\text{int}}(z)) \Rightarrow (\exists z' x < z' < y \wedge (M_{k+1}(z,z') \vee M_{k+1}(z',z))
\end{array}
$$

Note that the first line in the definition of M_k, ensures that the power to recognize a matching increases from M_{k+1} to M_k. This way M_0 will be true for all matchings which can occur in a word in L. Hence $M(x,y) = M_0(x,y)$. ◀

A position is of stack height 0 if all call-positions in the prefix have matching return-positions in the prefix. Similar a position is of stack height i if there are i call-positions in the prefix with push letters and all positions are matched by positions in the same interval generated by those i positions.

▶ **Lemma 22.** *For every constant $0 \leq j < m$, we can define a monadic predicate $H_k(x)$ in $FO[+]$ such that:*
- $w_{x=i} \models H_j(x)$ *then* $\Delta(w_1 \dots w_{i-1}) = k$ *for arbitrary* $w \in \Sigma^*$.
- $w_{x=i} \models H_j(x)$ *iff* $\Delta(w_1 \dots w_{i-1}) = k$ *for all* $w \in L$.

Proof. A position i has stack height k iff all but k call letters in the prefix $w_1 \dots w_{i-1}$ match. It is obvious that this can be defined in FO[+] using our matching predicates.

The matching predicates might have false negatives resulting in false negatives of H_k. But in the case of $w \in L$ and the case that the height at position i is $k < m$ the matching predicate is exact on the prefix $w_1 \dots w_{i-1}$ and hence the height is correctly presented by H_k. ◀

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.

3.3 The Regular Part

In this section we will show a second property which in addition to the property of the previous section - simple height behavior - is sufficient to characterize the visibly counter languages in AC^0. This second property concerns $R_{\mathcal{A}}$. If $R_{\mathcal{A}}$ is in FO[*Reg*] and if L has simple height behavior, then we can build a FO[+] formula for L. Unfortunately there are cases where $R_{\mathcal{A}}$ is not in FO[*Reg*], but still L is in FO[+]. The problem here is that there can be words which are witness for $R_{\mathcal{A}}$ not being quasi-aperiodic, but which are not images of τ_m; then we cannot deduct that L is not in AC^0.

First we introduce a normal-form on visibly counter automata which concerns loops. In the following definition we call a state q *dead* if there is no $w = w_1 w_2 \in L$, so that $(q_0, 0) \xrightarrow{w_1} (q, h) \xrightarrow{w_2} (q', h')$ with $h \geq m$.

▶ **Definition 23.** 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 there is $z \in \Sigma^*$ with $xyz \in L(\mathcal{A})$ and one of the following is true, depending on $\Delta(y)$:
- If $\Delta(y) > 0$, then there is a partition of z into $z = z_1 z_3$ and a word $z_2 \in \Sigma^*$ so that for all $i \geq 0$ we have that $xyy^i z_1 z_2^i z_3 \in L(\mathcal{A})$.
- If $\Delta(y) < 0$, then there is a partition of x into $x = x_1 x_3$ and a word $x_2 \in \Sigma^*$ so that for all $i \geq 0$ we have that $x_1 x_2^i x_3 y y^i z \in L(\mathcal{A})$.
- If $\Delta(y) = 0$, then $xy^i z \in L$ for all $i \geq 0$.

We also require that $\delta_i = \delta_m$ for $m - |Q| < i < m$.

The idea of this definition is, that if a prefix reaches a state in $\mathcal{A}$ that can be completed to a word in L then no matter how many loops through this state are appended, the word can still be completed to a word in L.

▶ **Lemma 24.** *For every $m-$VCA $\mathcal{A}$ recognizing a language L there is a loop-normal $m'-$VCA $\mathcal{A}'$ recognizing L.*

Proof of lemma 24. Given an $m-$VCA $\mathcal{A}$, we construct an automaton $\mathcal{A}'$ with the same language and show that the loop-normal property is satisfied.

Let Q be the set of states and $\delta_0, \dots, \delta_m$ be the transition functions of $\mathcal{A}$. The set of states in $\mathcal{A}'$ is $Q' = Q \times \{0, \dots, |Q!| - 1\}$. The new second component is a counter modulo $|Q|!$ and does not affect the language. So if $\delta_i(q, a) = q'$ then $\delta'_i((q, k), a) = (q', k + \Delta(a) \mod |Q|!)$. Further $q'_0 = (q_0, 0)$ and $F' = F \times \{0, \dots, |Q!| - 1\}$. The threshold of $\mathcal{A}'$ is $m' = m + |Q'|$ but we do not actually use it in the transition function: $\delta'_i = \delta'_m$ for $m < i \leq m + |Q'|$.

It is easy to see that $L(\mathcal{A}) = L(\mathcal{A}')$.

Now we verify that $\mathcal{A}'$ is loop-normal. Assume x, y are given as in definition 23 with $(q_0, 0) \xrightarrow{x} (q, h_1) \xrightarrow{y} (q, h_2)$ and $\Delta(xy_1 \dots y_k) \geq m)$ for all $1 \leq k \leq |y|$. We treat the three cases for $\Delta(y)$ of the definition:

- $\Delta(y) = 0$. In this case we can obviously loop y more often.
- $\Delta(y) > 0$. Choose z so that $xyz \in L$. If in xyz after x the height profile never gets below m, we can choose $z_1 = z$ and $z_2 = z_3 = \epsilon$.
  If there is some position j in z, so that $\Delta(xyz_1 \cdots z_j) < m$, then in the run of the original automaton $\mathcal{A}$ on xyz there must be a down loop, since $h_2 > m + |Q|!$, so let be q the looped state and z'_2 be a word looping through q with $|z'_2| \leq |Q|$. Let $z = z_1 z_3$ such that $(q_0, 0) \xrightarrow{xyz_1} (q, \Delta(xyz_1))$. Note that $\Delta(y)$ is a multiple of $\Delta(z'_2)$. Now let
  $$z_2 = {z'_2}^{\frac{\Delta(y)}{\Delta(z'_2)}}.$$
  Since $\Delta(y) = \Delta(z_2)$, clearly $xyy^i z_1 z_2^i z_3 \in L$ for all $i \geq 0$.
- $\Delta(y) < 0$. This case is similar to the previous one with the only difference that we always have to consider proper up and down loops.

So in any case we have the desired property. ◀

The intersection of the regular languages and AC^0 is characterized by the quasi-aperiodicity of the syntactic morphism. A regular language R is in AC^0 iff $\eta_R(\Sigma^t)$ has only trivial groups for all t. If $R \notin \mathrm{AC}^0$ then there exist words of equal length spanning a group. In this case we can use those words to build an AC^0 reduction from an ACC^0_k-hard language to R. The same we want to do with $R_\mathcal{A}$. Unfortunately there can be words of equal length spanning a group in the syntactic monoid of $R_\mathcal{A}$ but still $L = L(\mathcal{A})$ is in AC^0. The reason is that actually we are only interested in $R_\mathcal{A} \cap \tau_m(\Sigma^*)$, i.e. in a restricted set of inputs. This intersection however is not regular any more.

In the following we use $\tau_m(\Sigma^*)$ which is the set of *restricted inputs* we are interested in. It contains prefixes of labeled well-matched words. The set $F(\tau_m(\Sigma^*))$ is the set of factors of words in $\tau_m(\Sigma^*)$. Keep in mind that τ_m is not defined for inputs with negative height-profile, e.g. $\tau_m(ba)$ is undefined if a is a push and b a pop letter.

▶ **Lemma 25.** *If $\mathcal{A}$ is a loop-normal $m-$VCA then if there is a number $t > 0$ and a set $G \subseteq \Sigma_m^t$ with $G^* \subseteq F(\tau_m(\Sigma^*))$ so that the set $\eta_{R_\mathcal{A}}(G)$ contains a non-trivial group, then $L \notin AC^0$.*

Proof of lemma 25. First we assume that L has simple height behavior. Otherwise L is not in AC^0 anyway. If there is some t and $G \subseteq \Sigma_m^t$ with $G^* \subseteq F(\tau_m(\Sigma^*))$ so that $\eta_{R_\mathcal{A}}(G)$ contains a group of order k, then the ACC^0_k-hard language Mod_k is reducible to $R_\mathcal{A}$. We show that then, Mod_k is also reducible to $L = L(\mathcal{A})$.

Let us say that G is such a set of words generating a group $\mathbb{Z}_k$ in the syntactic monoid, i.e. $\eta_{R_\mathcal{A}}(G) = \mathbb{Z}_k$. Let $g_1 \in G$ so that $\eta_{R_\mathcal{A}}(g_1)$ is the neutral element of $\mathbb{Z}_k$ and let g_2 be so that $\langle \eta_{R_\mathcal{A}}(g_2) \rangle = \mathbb{Z}_k$.

In the following, for each $g_i \in \Sigma_m^*$, $\bar{g}_i \in \Sigma^*$ denotes the projection of g_i to the first element, i.e. the label is deleted.

We have to consider several cases:

1. The first one is, that $G \subseteq (\Sigma \times \{m\})^t$ and $\Delta(\bar{g}_1) = \Delta(\bar{g}_i)$ for all $g_i \in G$.
   We get three sub-cases:
   - $\Delta(\bar{g}_1) = 0$. Let $u, v \in \Sigma_m^*$ be a context so that $ug_1v \in R_{\mathcal{A}}$ but $ug_2v \notin R_{\mathcal{A}}$ and $\bar{u}\bar{g}_1\bar{v} \in L$ but $\bar{u}\bar{g}_2\bar{v} \notin L$. Here we use that the topmost $|Q|$ transition functions are equal. This yields an AC^0-computable reduction $f\colon \{0,1\}^* \to \Sigma^*$ from MOD_k to L with $f(w) = \bar{u}\phi(w)\bar{v}$ and $\phi(0) = \bar{g}_1, \phi(1) = \bar{g}_2$.
   - $\Delta(\bar{g}_1) > 0$. In this case we have to take the stack into account and that the automaton is loop-normal. Again we choose a context u, v so that $\bar{u}\bar{g}_1\bar{v} \in L$ but $\bar{u}\bar{g}_i\bar{v} \notin L$. We can assume that $\mathcal{A}$ is in the same state after reading $\bar{u}$ and $\bar{u}\bar{g}_1$. Since the automaton is loop-normal we know that we can split v into $v = v_1v_3$ and there is a word v_2 so that $\bar{u}\bar{g}_1\bar{g}_1^i\bar{v}_1\bar{v}_2^i\bar{v}_3 \in L$. The reduction f now looks like this: $f(w) = \bar{u}\bar{g}_1\phi(w)\bar{v}_1\bar{v}_2^{|w|}\bar{v}_3$ with $\phi(0) = \bar{g}_1, \phi(1) = \bar{g}_2$.
   - $\Delta(\bar{g}_1) < 0$. This case is treated similarly as the previous one.
2. The second case is that not all g_i are in $(\Sigma \times \{m\})^t$. Since $G^* \subseteq F(\tau_m(\Sigma^*))$, $\Delta(\bar{g}) = 0$ for all $g \in G$. So this case can be treated similarly to the first sub-case above.
3. The third case is, that $G \subseteq (\Sigma \times \{m\})^t$ and not $\Delta(\bar{g}_1) = \Delta(\bar{g}_i)$ for all $g_i \in G$. This is a contradiction to simple height behavior.

◀

▶ **Lemma 26.** *If for all $t > 0$ and for all G with $G \subseteq \Sigma_m^t$ and $G^* \subseteq F(\tau_m(\Sigma^*))$ the set $\eta_{R_{\mathcal{A}}}(G)$ does not contain a non-trivial group, then there is a* FO[*Reg*] *formula ϕ with*

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

Proof of lemma 26. Our proof is very similar to the proof of the fact that languages with quasi-aperiodic syntactic morphism are in FO[*Reg*]. We adapt the proof from Straubing's text book [18]. First we will sketch the proof from the book. We stick very closely to it and adapt the notation. After, we show how so apply it to our case.

Given is some regular language - in our case $R_{\mathcal{A}}$. Let M be the syntactic monoid of $R_{\mathcal{A}}$. We look at the sequence $\eta_{R_{\mathcal{A}}}(\Sigma_m^1), \eta_{R_{\mathcal{A}}}(\Sigma_m^2), \eta_{R_{\mathcal{A}}}(\Sigma_m^3), \ldots$. Since M is finite, this sequence must begin to circle at some point. We find a natural p so that $\eta_{R_{\mathcal{A}}}(\Sigma_m^p) = \eta_{R_{\mathcal{A}}}((\Sigma_m^p)^+) = S$.

Let $B = \Sigma_m^p$. We will consider B an alphabet. Further let $\beta\colon B \to S$ with $\beta(b) = \eta_{R_{\mathcal{A}}}(b)$. Let $S_1 = S \cup \{1\}$. We can extend β to $\beta\colon B^* \to S_1$. Now

$$R_{\mathcal{A}} = \bigcup_{0 \le |w| < p} L_w w,$$

where

$$L_w = \{u \in (\Sigma_m^p)^* \mid uw \in R_{\mathcal{A}}\}.$$

It is easy to see, that if $L_w \in$ FO[*Reg*] then also $L_w w \in$ FO[*Reg*] and hence $R_{\mathcal{A}}$. So from now on we are only interested in L_w.

L_w is recognized by the monoid S_1 with $\eta_{L_w} = \beta$ and accepting set

$$X = \{m \in S_1 \mid m \cdot \eta_R(w) \in \eta_{R_{\mathcal{A}}}(R_{\mathcal{A}})\}.$$

If S_1 is aperiodic which is the case if $\eta_{R_{\mathcal{A}}}$ is quasi-aperiodic, then there is a FO[<] formula ψ for L_w over the alphabet B which can be modified to get a FO[*Reg*] formula ψ' for L_w over the original alphabet Σ_m.

Up to here, we strictly followed Straubing. Now back to our case: Since we have a weaker property than quasi-aperiodicity of $\eta_{R_\mathcal{A}}$, we cannot assume that S_1 is aperiodic. However we know that group generating inputs never occur, so we can get rid of multiplications for invalid inputs. The language does not change under the restricted inputs but the monoid becomes aperiodic. Using S_1, we construct a new monoid S'. Assume S_1 has a 0 element. Begin with S_1 and then for each $m_1, m_2 \in S_1$ we modify the multiplication for $m_1 \cdot m_2$ if the following is true: $\eta_{R_\mathcal{A}}^{-1}(g_1)\eta_{R_\mathcal{A}}^{-1}(g_2) \cap F(\tau_m(\Sigma^*)) = \emptyset$. If this is the case then we define the result of $m_1 \cdot m_2$ as 0. The result is still a monoid. If we use S' instead of S_1 for the construction of the formula for L_w, it does not change the language under the restricted inputs. What we are left with is proving that S' is indeed aperiodic.

Assume S_1 contains a non-trivial group $\mathcal{G} = \{g_1, \ldots, g_k\}$. It is sufficient if we only look at the case that $\mathcal{G}$ is cyclic. We show that this group does not exist in S' any more. Let $G = \eta_{R_\mathcal{A}}^{-1}(\mathcal{G}) \cap F(\tau_m(\Sigma^*)) \subseteq \Sigma_m^*$ and $G_i = \eta_{R_\mathcal{A}}^{-1}(g_i) \cap F(\tau_m(\Sigma^*)) \subseteq \Sigma_m^*$. If $GG \cap F(\tau_m(\Sigma^*)) = \emptyset$ we are done. This is because in this case all multiplications in the group have been set to 0 in S'.

So now we assume that $GG \cap F(\tau_m(\Sigma^*)) \neq \emptyset$. If there are $i \neq j$ so that $G_i \neq \emptyset \neq G_j$ and both $G_iG_j \cap F(\tau_m(\Sigma^*))$ and $G_jG_i \cap F(\tau_m(\Sigma^*))$ are not empty we have a contradiction to the premise of the lemma that for all $t > 0$ and for all $G \subseteq \Sigma_m^t$ with $G^* \subseteq F(\tau_m(\Sigma^*))$ the set $\eta_{R_\mathcal{A}}(G)$ does not contain a non-trivial group.

Now we have the case that there are $i \neq j$ as before but either $G_iG_j \cap F(\tau_m(\Sigma^*))$ or $G_jG_i \cap F(\tau_m(\Sigma^*))$ is empty. Assume $G_iG_j \cap F(\tau_m(\Sigma^*))$ is the empty one, then $g_i \cdot g_j = 0$ but $g_j \cdot g_i \neq 0$ in S'. Hence the group is not abelian which is a contradiction since we required it to be cyclic.

The only case left is that there is exactly one G_i, so that $G_i^* \subseteq F(\tau_m(\Sigma^*))$. Hence $G_i = G_i^*$, so the group was trivial.

◀

4 Results

If we combine our statements from the previous section, we get the following results.

▶ **Theorem 27.** *For a loop-normal $m-$VCA $\mathcal{A}$, $L = L(\mathcal{A})$ is in AC^0 if and only if*

- *$L(\mathcal{A})$ has simple height behavior and*
- *for all $t > 0$ and for all $G \subseteq \Sigma_m^t$ with $G^* \subseteq F(\tau_m(\Sigma^*))$ the set $\eta_{R_\mathcal{A}}(G)$ does not contain a non-trivial group.*

Proof. We already proved the direction from left to right with lemmas 11 and 25.

If we have $\mathcal{A}$ where all active states have a fixed slope and the formula ϕ from lemma 26, we can build a FO[+] formula for L: Begin with the FO[*Reg*] formula ϕ. This formula operates on the alphabet Σ_m and uses letter predicates $Q_{(a,k)}x$. We replace them by $(Q_a(x) \wedge H_i(x))$ if $k < m$ and by $(Q_a(x) \wedge H_{\geq m}(x))$ if $k = m$. The resulting formula is ϕ' and operates over Σ.

If a word w is in L then $w \models \phi'$.

The only thing we have to take care of are false positives in the $H_{\geq m}$ predicate which we mentioned earlier in the paper. A false positive here can only occur if there is a non-active state q without fixed slope. This state loops up and never comes down to m again (otherwise it would be active). So if we have a word which visits q but reaches a height smaller than m after q, it cannot be in L. But then there is a word $w' = w_1xw_2$, where w_1 brings $\mathcal{A}$ in the state q, x loops through q, $\Delta(x) > m$ and $w = w_1w_2$. On w', $H_{\geq m}$ will not have a false positive, since after q the word is always above m. Also w' cannot be in L and if $w' \notin L$ then also $w \notin L$. Hence if $w \notin L$ then $w \not\models \phi'$. ◀

In the proof of the previous theorem we construct a FO[+] formula for every VCL in AC^0. We can state our main result in different ways:

▶ **Corollary 28.** *The following statements are true:*
- $\mathrm{VCL} \cap \mathrm{FO}[arb] \subseteq \mathrm{FO}[+]$.
- $\mathrm{VCL} \cap \mathrm{AC}^0 \subseteq \mathrm{FO}[+]$.
- $\mathrm{VCL} \cap \mathrm{AC}^0 \subseteq \mathrm{DLOGTIME} - \mathrm{uniform}\mathrm{AC}^0$.

Since all the properties of the previous lemma are decidable we have an effective characterization.

▶ **Corollary 29.** *Given some visibly counter automaton $\mathcal{A}$, it is decidable whether $L(\mathcal{A})$ lies in AC^0, resp. in $\mathrm{FO}[+]$.*

Proof. Given $\mathcal{A}$, we have to check for all states of $\mathcal{A}$ if they are active and if they have a fixed slope. If there is an active state without fixed slope then $L(\mathcal{A}) \notin \mathrm{AC}^0$.

For deciding if a state $q \in Q$ has fixed slope, make a list of all words looping through q up to length $|Q|$. Then clalculate the slope of all words. They are all equal iff q has a fixed slope.

For deciding if a state q is active, check of there is a word x with $\Delta(x) > m + |Q|$ which brings $\mathcal{A}$ in state q. This is as hard as the membership problem for VCL. If such an x exists then check for all words $y \in \Sigma^*$ up to length $\Delta(x)|Q|$ if $xy \in L(\mathcal{A})$ and if there is a prefix y' of y with $\Delta(x) - \Delta(y') > |Q|$. If such a y exists then q is active.

Finally we have to decide our modified quasi-aperiodicy property. First of all, t can be bounded by a constant relative to the syntactic monoid of $R_{\mathcal{A}}$, where $\mathcal{A}$ is a loop-deterministic automaton for L. Then there are only finitely many sets G to consider. The requirement of $G^* \subseteq F(\tau_m(\Sigma^*))$ is equivalent to $GG \subseteq F(\tau_m(\Sigma^*))$ which then is also decidable. ◀

5 Discussion

Algebraic methods usable for languages up to now mainly pertain to finite monoids, i.e. to regular languages. We see our results as a step towards the further application of algebraic methods in the non-regular case. A natural continuation of this line of research would be an algebraic theory for visible pushdown languages and their subclasses. A promising approach to go here might be the use of forest algebras.

The characterization of the regular languages in AC^0 as the class $\mathrm{FO}[Reg]$ used the notion of *quasi-aperiodic* regular sets which is of an algebraic nature. Our result is oriented in this direction, but is not as algebraic. It still leaves open to characterize exactly the set of all visible counter languages contained in AC^0 in terms of logic. There is no obvious numerical predicate, which when used with first order formulae results in $\mathrm{VCL} \cap \mathrm{AC}^0$.

In [13] the notion of dense completeness has been introduced. A family of formal languages $\mathcal{F}$ is said to be densely complete in a complexity class $\mathcal{C}$ if both $\mathcal{F} \subset \mathcal{C}$ and for each $C \in \mathcal{C}$ there exists a $F \in \mathcal{F}$ so that $C \leq F$ and $F \leq C$, i.e.: F and C have the same complexity. While the context-free languages turn out to be densely complete in the class SAC^1, the regular languages are not densely complete in the class NC^1. As a consequence of our result we are able to show unconditionally that the visible one-counter languages, which are contained in NC^1, are not dense in NC^1. Up to now, dense families of formal languages are known for the non-deterministic classes, $\mathrm{NSPACE}(\log n)$, SAC^1, and NP, only.

In our work we explored the intersection of a formal language class and a circuit-based complexity class. Aside from the pair AC^0 and VCL, there are some other combinations worth being investigated using our methods.

References

1 Rajeev Alur and P. Madhusudan. Visibly pushdown languages. In László Babai, editor, *STOC*, pages 202–211. ACM, 2004.

2 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.

3 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.

4 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.

5 David A. Mix Barrington and Denis Thérien. Finite monoids and the fine structure of NC^1. *J. ACM*, 35(4):941–952, 1988.

6 Ashok K. Chandra, Steven Fortune, and Richard J. Lipton. Unbounded fan-in circuits and associative functions. *J. Comput. Syst. Sci.*, 30(2):222–234, 1985.

7 Patrick W. Dymond. Input-driven languages are in log n depth. *Inf. Process. Lett.*, 26(5):247–250, 1988.

8 Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. In *FOCS*, pages 260–270, 1981.

9 Yuri Gurevich and Harry R. Lewis. A logic for constant-depth circuits. *Information and Control*, 61(1):65–74, 1984.

10 Johan Håstad. Almost optimal lower bounds for small depth circuits. In *STOC*, pages 6–20. ACM, 1986.

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

12 Neil Immerman. *Descriptive Complexity.* Springer, New York, 1999.

13 Andreas Krebs and Klaus-Jörn Lange. Dense completeness. In Hsu-Chun Yen and Oscar H. Ibarra, editors, *Developments in Language Theory - 16th International Conference, DLT 2012, Taipei, Taiwan, August 14-17, 2012. Proceedings*, volume 7410 of *Lecture Notes in Computer Science*, pages 178–189. Springer, 2012.

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

15 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.

16 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.

17 Marcel Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965.

18 Howard Straubing. *Finite Automata, Formal Logic, and Circuit Complexity.* Birkhäuser, Boston, 1994.

19 H. Venkateswaran. Properties that characterize logcfl. *J. Comput. Syst. Sci.*, 43(2):380–404, 1991.

20 Heribert Vollmer. *Introduction to circuit complexity - a uniform approach.* Texts in theoretical computer science. Springer, 1999.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de