Characterizing TC^0 in Terms of Infinite Groups

Andreas Krebs, Klaus-Jörn Lange, Stephanie Reifferscheid
WSI, Sand 13, D-72076 Tübingen
{krebs,lange,reiffers}@informatik.uni-tuebingen.de

Topics

Complexity, Logic, Formal Languages

Abstract

We characterize the languages in $TC^0 = \mathcal{L}(Maj[<, Bit])$ and $\mathcal{L}(Maj[<])$ as inverse morphic images of certain groups. Necessarily these are infinite, since nonregular sets are concerned. To limit the power of these infinite algebraic objects, we equip them with a finite *type set* and introduce the notion of a *finitely typed* (infinite) monoid. Following this approach we investigate *type respecting* mappings and construct a new type of block product, which is more adequate to dealing with infinite monoids. We exhibit two classes of solvable finitely typed groups which exactly characterize TC^0 and $\mathcal{L}(Maj[<])$ via inverse morphisms.

1 Introduction

There are very close relations between families of regular languages and low level complexity classes ([2, 8]). In particular the class AC^0 corresponds to the family of star-free regular languages, the class ACC^0 to the family of solvable regular languages, and NC^1 to the family of all regular languages. Unfortunately, the class TC^0 is not treatable in the framework of regular languages unless it collapse with ACC^0 or NC^1, since a regular language is NC^1-hard if its syntactic monoid contains a nonsolvable group and is contained in ACC^0 otherwise ([1, 3]).

These connections between families of regular languages and low level complexity classes are expressed in the unifying framework of logic, where their difference is reflected by the use of different numerical predicates. For instance, the star-free regular languages are precisely those acceptable by first-order formulae using only order as numerical predicate, while (uniform) AC^0 coincides with those acceptable by formulae using in addition the BIT predicate. In the same way, the difference between the solvable regular languages (all regular languages) and the class ACC^0 (NC^1) is made in using or not the BIT predicate or related ones like multiplication and addition ([2]).

Using the BIT predicate the class TC^0 is expressible as the family of all languages acceptable by formulae built of first-order and majority quantifiers using order and the BIT predicate ([2]). Hence a natural candidate for a formal language counterpart of TC^0 is the family $\mathcal{L}(Maj[<])$ of all languages represented by majority formulae using only the order predicate and not the BIT predicate. It turns out, that both first-order quantifier and addition are definable in $Maj[<]$ ([4]). Hence these formulae have an undecidable satisfiability problem (over finite words).

The various families of regular languages can be characterized algebraically, e.g. the star-free regular sets are those recognizable by aperiodic finite monoids using inverse morphisms. It was the original starting point of this work to exhibit monoids which recognize by inverse morphisms exactly the languages in $\mathcal{L}(Maj[<])$.

When characterizing languages recognized by various classes of finite monoids in terms of certain formula classes the crucial point is to exhibit algebraic objects which correspond to logic operators. For instance, the application of a modular counting quantifier is characterized by building the block product with a cyclic group. In the same way, first order quantifier are expressed by the block product with the two element monoid U_1 ([8]).

It seems natural to relate in the same way majority (and counting and threshold quantifiers as well)

to the block product with the set $\mathbb{Z}$ of integers. The problem is now, that the ordinary block product with infinite monoids results in too powerful objects and it wouldn't be possible to simulate the resulting monoids by logical formulae. Hence we need a new, more restricted version of the algebraic approach. We obtain this in two steps: First, we introduce the notion of a *finitely typed monoid*, that is a monoid which is the finite disjoint union of subsets called *types*. When recognizing languages with such a monoid the inverse morphism is in some sense not allowed to distinguish the elements inside a type. Building on that we introduce the notion of a *type respecting* mapping.

In the second step we define a restricted version of the block product $W := M \,\square\, N$ of M and N as follows: Usually, for finite N, W is the bilateral semidirect product $M^{N\times N} * * N$. For infinite N the first component is uncountable and there would be no chance to define all languages that can be recognized by this monoid with the anticipated logical formula. Instead, we build the monoid $V * * N$ for a countable submonoid V of $M^{N\times N}$ which is generated by some well behaved type respecting mappings from N^2 to M. Using this restricted version of a block product we are able to construct two families $\mathbf{M}_<$ and $\mathbf{M}_{<,Sq}$ of finitely typed infinite solvable groups, such that TC^0 coincides precisely with the family of languages recognized by the elements of $\mathbf{M}_{<,Sq}$. The same connection holds between the class $\mathcal{L}(Maj[<])$ and the family $\mathbf{M}_<$.

The paper is organized as follows: Section 2 collects the needed preliminaries. The following two sections define the new block product and use it to construct families of infinite groups which are related to the classes TC^0 and $\mathcal{L}(Maj[<])$. Section 5 summarizes our main results which are proven in the final two sections.

2 Preliminaries

2.1 Logic formulae over words

Throughout this paper we consider languages defined by logical formulae. We use essentially the notation as it is presented in the book of Straubing ([8]). In general, i, j, k, n will denote positive integers, while x, y, z will denote position variables with positive integer values. The integer n will usually denote the length of the corresponding input word. Thus, variables will range over $\{1, 2, \cdots, n\}$. The predicate $Q_a(x)$ expresses that the position a variable x is pointing to contains the symbol a.

As usual, a formula ϕ with set $\mathcal{V}$ of free variables is interpreted over words as structures $w = (a_1, \mathcal{V}_1)(a_2, \mathcal{V}_2)\cdots(a_n, \mathcal{V}_n)$ such that $\mathcal{V}$ is the union of the $\mathcal{V}_i$ and that the $\mathcal{V}_i$ are pairwise disjoint. Words of this kind are called *$\mathcal{V}$-structures*. A letter $(a, \emptyset)$ is simply denoted by a. We denote the set of all $\mathcal{V}$-structures over Σ by $\Sigma^* \otimes \mathcal{V}$ whereas $(\Sigma \times 2^{\mathcal{V}})^*$ contains also words with multiple occurrences of a variable. The set of $\mathcal{V}$-structures modeling ϕ is denoted by $L_{\phi,\mathcal{V}}$.

Remark 2.1. Is ϕ a sentence, then $L_\phi := L_{\phi,\emptyset} = \{w \in \Sigma^* \mid w \models \phi\}$.

We call this the set of words defined by ϕ. The nonemptiness problem of L_ϕ is identical to the satisfiability problem of ϕ over words.

If $w = (a_1, \mathcal{V}_1)(a_2, \mathcal{V}_2)\cdots(a_n, \mathcal{V}_n) \in \Sigma^* \otimes \mathcal{V}$ and $x \notin \mathcal{V}$ then $w_{x=i}$ denotes the word $(a_1, \mathcal{V}_1)\cdots(a_{i-1}, \mathcal{V}_{i-1})(a_i, \mathcal{V}_i \cup \{x\})(a_{i+1}, \mathcal{V}_{i+1})\cdots(a_n, \mathcal{V}_n) \in \Sigma^* \otimes (\mathcal{V} \cup \{x\})$. As abbreviation for $w_{x=i} \models \phi$ we often write $w \models \phi(x = i)$, or if x is understood, $w \models \phi(i)$.

Let $\mathcal{Q}$ be a set of quantifier types (e.g. first-order) and $\mathcal{X}$ a set of numerical predicates (possibly not containing the order predicate $<$).We denote by $\mathcal{Q}[\mathcal{X}]$ the set of all formulae built over the elements of $\mathcal{X} \cup Q_a(.)$ as atomic formulae by conjunction, negation and quantification using quantifier types from $\mathcal{Q}$. By $\mathcal{L}(\mathcal{Q}[\mathcal{X}])$ we denote the class of all languages definable by $\mathcal{Q}[\mathcal{X}]$ formulae.

Formulae not using the $Q_a(.)$ predicates are *numerical predicates*. It is possible to define in $FO[<]$ the following numerical predicates: $=, \neq, >, \leq, \geq, +1$ (successor), -1 (predecessor), MIN (first position), and MAX (last position). Barrington et al. showed that in the presence of the order predicate first-order quantifiers can define addition and multiplication by use of the BIT predicate and vice versa ([2]). A very comprehensive treatment of related results is given by Schweickard ([7]).

We will use Maj to denote the majority quantifier. $w \models Maj\ x\ \phi$ is fulfilled iff the number of all $1 \leq i \leq |w|$ such that $w_{x=i} \models \phi$ is larger than $\lfloor |w|/2 \rfloor$. The majority quantifier rejects in case of a draw.

Lautemann et al. showed that all numerical predicates definable by $Maj[<]$-formulae are even definable by $FO[+]$ formulae ([5]). As a consequence they got:

Theorem 2.2.

$$\mathcal{L}(Maj[<, +, *]) \not\subset \mathcal{L}(Maj[<])$$

The (unary) counting quantifier is denoted by $\exists^{=y}x$. $w \models \exists^{=y}x\ \phi$ is fulfilled iff there are exactly j positions $1 \leq i \leq |w|$ such that $w_{x=i} \models \phi$ where j is the numerical value of variable y, i.e. y points to the j-th symbol of w. The counting quantifier can take values in the range$\{0, 1, \ldots, n\}$ which is one mor then there are positions available in inputs of Size n. If we care for the case $y = 0$ by the formula $\forall x \neg\phi$ and only treat the case $y \geq 1$ it is possible to show:

Lemma 2.3 ([4]). *The counting quantifier is definable in $Maj[<]$.*

For the ease of handling we will use the counting quantifier without this restriction and note that more formally in all formulae using the counting quantifier the exception handling of the case $y = 0$ should be added.

2.2 Circuits

The reader is assumed to be acquainted with language classes defined by uniform circuit classes as they are presented by Barrington et al. ([2]). In particular we will use the following relation:

$$\mathrm{TC}^0 = \mathcal{L}(FO + Maj[<, +, *]).$$

2.3 TC^0 and the Square Predicate

Let $square(x)$ denote the numerical predicate that the position number the variable x is pointing to is a positive square number. It is well known, that with respect to first-order quantifiers addition and multiplication are equivalent to addition and the square predicate (see e.g. [7][Theorem 2.3.f]). Combined with results in [4] this yields

$$\mathrm{TC}^0 = \mathcal{L}(Maj[<, square]).$$

We will use Sq to denote the *square quantifier*. $w \models Sq\ x\ \phi$ is fulfilled iff the number of all $1 \leq i \leq |w|$ such that $w_{x=i} \models \phi$ is a positive square number. It is easy to show that the square predicate *square* is definable in $FO + Sq[<]$ logic. On the other hand, the square quantifier can be simulated by the counting quantifier in connection with the square predicate. Hence we have

$$\mathrm{TC}^0 = \mathcal{L}(Maj + Sq[<]).$$

2.4 List of Symbols

Symbol	Explanation	Page
$\mathbb{Z}$	Integers	
$\mathbb{Z}^+$	Positive integers	
$\mathbb{Z}_0^-$	Nonpositive integers	
$\mathbb{S}$	Positive quadratic numbers	
$\overline{\mathbb{S}}$	Nonquadratic numbers and zero	
$\Sigma \otimes 2^{\mathcal{V}}$	Set of all $\mathcal{V}$-structures over Σ	
$L_{\phi,\mathcal{V}}$	Set of $\mathcal{V}$-structures modeling ϕ	
π_i	Projection to the ith coordinate	
e_G	Neutral element of G	
2^d	$\{(a_1, \ldots, a_d) \mid a_i \in \{0, 1\}\}$	
$2^{\mathcal{V}}$	Set of all subsets of $\mathcal{V}$	
$(G, \mathfrak{G})\ \square\ (H, \mathfrak{H})$	Block product of $(G, \mathfrak{G})$ with $(H, \mathfrak{H})$	4
$H^{-1}(\mathbf{M})$	Set of languages recognized by some element of $\mathbf{M}$	5
$\mathbf{M}_<,\ \mathbf{M}_{<,Sq}$	Group classes	5
$\Gamma^{G,\mathcal{G},m}\ x\ \overrightarrow{\phi}(x)$	Group quantifier	8
$\Gamma^{G,\mathcal{G},m}_{p..q}\ x\ \overrightarrow{\phi}(x)$	Relative group quantifier	8
$(G, \mathfrak{G}) \triangleright \mathcal{Q}[\mathcal{X}]$	$(G, \mathfrak{G})$ is definable in $\mathcal{Q}[\mathcal{X}]$	8

3 Finitely typed monoids

We call a monoid M *finitely typed with type set* $\mathfrak{M} = \{\mathcal{M}_i \mid i \in I\}$ iff $M = \dot{\bigcup}_{i\in I}\mathcal{M}_i$ for a finite set I. The pairwise disjoint sets $\mathcal{M}_i,\ i \in I$, are called the *types* of M. We call the elements of $\mathcal{B}(\mathfrak{M})$, (the boolean algebra generated by $\mathfrak{M}$), *extended types* of M. If the type set $\mathfrak{M}$ of M is understood we often simply write M instead of $(M, \mathfrak{M})$.
Let $(M, \mathfrak{M}),\ (N, \mathfrak{N})$ be finitely typed monoids. We call $\phi := (\phi_M, \phi_{\mathfrak{M}})$ a *type morphism* iff

(i) $\phi_M : M \to N$ is a monoid morphism

(ii) $\phi_{\mathfrak{M}} : \mathcal{B}(\mathfrak{M}) \to \mathcal{B}(\mathfrak{N})$ is a lattice morphism

(iii) $\phi_M(\mathcal{M}) = \phi_{\mathfrak{M}}(\mathcal{M}) \cap \phi_M(M)$ for all extended types $\mathcal{M}$ of M.

A type morphism $\phi = (\phi_M, \phi_{\mathfrak{M}})$ is called injective (resp. surjective) iff $\phi_M, \phi_{\mathfrak{M}}$ are injective (resp. surjective). The image $\phi_M(M)$ of a finitely typed monoid $(M, \mathfrak{M})$ under a type morphism $\phi = (\phi_M, \phi_{\mathfrak{M}})$ can be equipped with the type set $\phi_M(\mathfrak{M})$ given by the set $\mathcal{B}(\{\phi_M(\mathcal{M}) \mid \mathcal{M} \in \mathfrak{M}\})$.

Remark 3.1. Let M be a monoid and let $\mathfrak{M}_1,\ \mathfrak{M}_2$ be type sets such that $\mathfrak{M}_1$ is coarser than $\mathfrak{M}_2$ (that is, $\mathcal{B}(\mathfrak{M}_1) \subseteq \mathcal{B}(\mathfrak{M}_2)$).
Then $\phi = (\phi_M, \phi_{\mathfrak{M}})$ with $\phi_M, \phi_{\mathfrak{M}}$ the identities, is an injective type morphism.

We call a monoid $(U, \mathfrak{U})$ a *submonoid* of $(M, \mathfrak{M})$ ($(U, \mathfrak{U}) \leq (M, \mathfrak{M})$) iff there exists an injective type morphism $\phi : (U, \mathfrak{U}) \to (M, \mathfrak{M})$. Analogously, we call a monoid $(N, \mathfrak{N})$ a *quotient* of (M,$\mathfrak{M}$) iff there is a surjective type morphism $\phi : (M, \mathfrak{M}) \to (N, \mathfrak{N})$. Finally we define the *direct product* $(M, \mathfrak{M}) \times (N, \mathfrak{N})$ of finitely typed monoids $(M, \mathfrak{M})$ and $(N, \mathfrak{N})$ as the usual Cartesian product equipped with $\mathfrak{M} \times \mathfrak{N} := \{\mathcal{M} \times \mathcal{N} \mid \mathcal{M} \in \mathfrak{M},\ \mathcal{N} \in \mathfrak{N}\}$.

Let $(M, \mathfrak{M}),\ (N, \mathfrak{N})$ be finitely typed monoids. We introduce two kinds of functions which are dealing with types. We call a function $f : N \times N \to M$ *strongly type respecting* iff $f|_{\mathcal{N}_1\times\mathcal{N}_2}$ is constant for all $\mathcal{N}_1,\ \mathcal{N}_2 \in \mathfrak{N}$. The set of all strongly type respecting functions is denoted by $STR(N, M)$. The second kind of functions are the *type dependent* functions: Assign to each $\mathcal{N} \in \mathfrak{N}$ an element $m_{\mathcal{N}} \in M$ and denote this collection by $m_{\mathfrak{N}} = (m_{\mathcal{N}})_{\mathcal{N}\in\mathfrak{N}}$. Then for $c \in N$ we denote the type dependent function $f_c^{m_{\mathfrak{N}}} : N \times N \to M$ by $f_c^{m_{\mathfrak{N}}}(x, y) = m_{\mathcal{N}}$ iff $x \cdot c \cdot y \in \mathcal{N}$. The set of all type dependent functions is denoted by $TD(N, M)$.

Definition 3.2 (Block product). Let $(M, \mathfrak{M}),\ (N, \mathfrak{N})$ be finitely typed monoids. The *finitely typed block product* $(W, \mathfrak{W}) := (M, \mathfrak{M}) \,\square\, (N, \mathfrak{N})$ of $(M, \mathfrak{M})$ with $(N, \mathfrak{N})$ is defined as the bilateral semidirect product $V * * N$ of the submonoid V generated by $V^{(1)}$ and $V^{(2)}$ with N where

- $V^{(1)} = \{f_{s,t} : N \times N \to M \mid s \cdot f \cdot t \in STR(N, M), s, t \in N\}$
- $V^{(2)} = TD(N, M)$.

We call the elements of V *type respecting* (w.r.t. N and M). The left (resp. right) action of N on V is given by $(f \cdot n)(x, y) := f(x, ny)$(resp. $(n \cdot f)(x, y) := f(xn, y)$), $n \in N,\ f \in V$.

The type set $\mathfrak{W}$ of W consists of all types

$$\mathcal{W}_{\mathcal{M}} = \{(f, n) \in W \mid f(e_N, e_N) \in \mathcal{M}\},$$

where $\mathcal{M} \in \mathfrak{M}$.

Remark 3.3. *(a)* As usual we write the operation in V additively to provide a more readable notation. Note that this does not imply that V is commutative. By definition of the bilateral semidirect product we have

$$(f_1, n_1) \dots (f_r, n_r) = (\sum_{i=1}^{r} n_1 \dots n_{i-1} \cdot f_i \cdot n_{i+1} \dots n_r, n_1 \dots n_r).$$

The neutral element of $(M, \mathfrak{M}) \,\square\, (N, \mathfrak{N})$ is $(\mathbf{e}, e_N)$ where $\mathbf{e}(x, y) = e_M$ for all $x, y \in N$. If M, N are groups, then $(M, \mathfrak{M}) \,\square\, (N, \mathfrak{N})$ is a group as well and $(f, n)^{-1} = (-n^{-1} \cdot f \cdot n^{-1}, n^{-1})$.

(b) The range of f is finite for every $f \in V$.

(c) For $f \in V$ there is a finite set $\{f_j \mid j \in J\} \subseteq V^{(1)} \cup V^{(2)}$ with $f = \sum_{j\in J} f_j$. Using the notation above we have $f_j = f_{s_j,t_j}$ iff $f_j \in V^{(1)}$ and $f_j = f_{c_j}^{m_{\mathfrak{N}}^j}$ iff $f_j \in V^{(2)}$. Thus we can determine $f(x, y)$ by determining types $\mathcal{N}_{j_1},\ \mathcal{N}_{j_2},\ \mathcal{N}_{j_3} \in \mathfrak{N}$ such that

(i) $x \cdot s_j \in \mathcal{N}_{j_1}$ for $f_j \in V^{(1)}$.

(ii) $t_j \cdot y \in \mathcal{N}_{j_2}$ for $f_j \in V^{(1)}$.

(iii) $x \cdot c_j \cdot y \in \mathcal{N}_{j_3}$ for $f_j \in V^{(2)}$.

Definition 3.4. A finitely typed monoid $(M, \mathfrak{M})$ is said to *accept* a language $L \subseteq \Sigma^*$ iff there is a morphism $h : \Sigma^* \to M$ and a subset $\{\mathcal{M}_1, \ldots \mathcal{M}_k\} \subseteq \mathfrak{M}$ of types of M such that $L = h^{-1}(\bigcup_{i=1}^{k} \mathcal{M}_i)$. If the type set $\mathfrak{M}$ of M is fixed we simply say that M accepts L.

Let $\mathbf{M}$ be a class of finitely typed monoids. We denote the set of all languages recognized by some element of $\mathbf{M}$ with $H^{-1}(\mathbf{M})$.

Remark 3.5. If a language L is accepted by a submonoid or a quotient of M, then M accepts L as well.

4 The class $\mathbf{M}_\mathfrak{Z}$

Let $\mathfrak{Z}$ be a type set of $\mathbb{Z}$. We now define the class $\mathbf{M}_\mathfrak{Z}$ recursively by starting with the integers and repeatedly applying the block product. During this process we will associate with each element G of $\mathbf{M}_\mathfrak{Z}$ the *depth* $Dp(G)$ of G and the *width* $Wd(G)$ of G.

- $(\mathbb{Z}, \mathfrak{Z})$ is in $\mathbf{M}_\mathfrak{Z}$. We set $Dp(\mathbb{Z}, \mathfrak{Z}) := 1$ and $Wd(\mathbb{Z}, \mathfrak{Z}) := 1$.
- For $(G, \mathfrak{G})$ and $(H, \mathfrak{H})$ in $\mathbf{M}_\mathfrak{Z}$ the direct product $(G, \mathfrak{G}) \times (H, \mathfrak{H})$ is in $\mathbf{M}_\mathfrak{Z}$ as well.
  The depth is set to $Dp(G \times H, \mathfrak{G} \times \mathfrak{H}) := Max\{Dp(G, \mathfrak{G}), Dp(H, \mathfrak{H})\}$. The width is set to be the sum of the widths of the components: $Wd(G \times H, \mathfrak{G} \times \mathfrak{H}) := Wd(G, \mathfrak{G}) + Wd(H, \mathfrak{H})$.
- For $(G, \mathfrak{G})$ and $(H, \mathfrak{H})$ in $\mathbf{M}_\mathfrak{Z}$ the finitely typed block product W:=$(G, \mathfrak{G}) \,\square\, (H, \mathfrak{H})$ is in $\mathbf{M}_\mathfrak{Z}$.
  The depth is set to $Dp(W, \mathfrak{W}) := Dp(G, \mathfrak{G}) + Dp(H, \mathfrak{H})$, the width is defined by $Wd((W, \mathfrak{W}) := 1$.

Note that the elements of depth 1 in $\mathbf{M}_\mathfrak{Z}$ are commutative. Moreover it is readily seen that all elements of $\mathbf{M}_\mathfrak{Z}$ are solvable where the derived length of a group $G \in \mathbf{M}_\mathfrak{Z}$ is bounded by the depth $d(G)$ of G.

Let $\mathbf{M}$ be a class of finitely typed monoids closed under forming finite direct products, We denote by $\overline{\mathbf{M}}$ the smallest class $\mathbf{C}$ of finitely typed monoids such that (i) $\mathbf{M}$ is contained in $\mathbf{C}$ (ii) $\mathbf{C}$ is closed under forming submonoids, and (iii) $\mathbf{C}$ is closed under forming quotients. As usual (see for instance [6]), the following properties of this closure operation hold:

Remark 4.1. *(a)* $\overline{\mathbf{M}}$ is the class of all finitely typed monoids $(G, \mathfrak{G})$ such that there exists $(H, \mathfrak{H}) \in \mathbf{M}$, a submonoid $(U, \mathfrak{U})$ of $(H, \mathfrak{H})$ and a surjective type morphism $\phi : (U, \mathfrak{U}) \to (G, \mathfrak{G})$.

(b) $\overline{\mathbf{M}}$ is closed under forming finite direct products.

(c) For a partition $\mathfrak{Z}$ of $\mathbb{Z}$ we have $H^{-1}(\mathbf{M}_\mathfrak{Z}) = H^{-1}(\overline{\mathbf{M}_\mathfrak{Z}})$.

(d) Let $\mathfrak{Z}_1$, $\mathfrak{Z}_2$ be partitions of $\mathbb{Z}$ such that $\mathfrak{Z}_1$ is coarser than $\mathfrak{Z}_2$. Then $\overline{\mathbf{M}_{\mathfrak{Z}_1}} \subseteq \overline{\mathbf{M}_{\mathfrak{Z}_2}}$. In particular, $H^{-1}(\mathbf{M}_{\mathfrak{Z}_1}) \subseteq H^{-1}(\mathbf{M}_{\mathfrak{Z}_2})$.

Lemma 4.2. *Let* $\mathfrak{Z}_1$, $\mathfrak{Z}_2$ *be type sets of* $\mathbb{Z}$. *Then* $\overline{\mathbf{M}_{\mathfrak{Z}_1} \times \mathbf{M}_{\mathfrak{Z}_2}} = \overline{\mathbf{M}_{\mathfrak{Z}_1 \cap \mathfrak{Z}_2}}$, *where* $\mathfrak{Z}_1 \cap \mathfrak{Z}_2 = \{\mathcal{Z}_1 \cap \mathcal{Z}_2 \mid \mathcal{Z}_i \in \mathfrak{Z}_i,\ i = 1, 2\}$, *and* $\mathbf{M}_{\mathfrak{Z}_1} \times \mathbf{M}_{\mathfrak{Z}_2} = \{M_1 \times M_2 \mid M_i \in \mathbf{M}_{\mathfrak{Z}_i}\}$.

Proof. Obvious. □

Definition 4.3. In the following we denote by $\mathbf{M}_<$ the class $\mathbf{M}_\mathfrak{Z}$ for $\mathfrak{Z} = \mathfrak{Z}_< = \{\mathbb{Z}^+, \mathbb{Z}_0^-\}$ and by $\mathbf{M}_{<,Sq}$ the class $\mathbf{M}_\mathfrak{Z}$ for $\mathfrak{Z} = \mathfrak{Z}_{<,Sq} = \{\mathbb{S}, \mathbb{Z}^+ \setminus \mathbb{S}, \mathbb{Z}_0^-\}$.

5 Main results

The central result of this work is the characterization of the languages in $\mathcal{L}(Maj[<])$ and in TC^0 as inverse morphic images of groups in $\mathbf{M}_<$ and $\mathbf{M}_{<,Sq}$. We prove this in two parts. First, in Theorem 5.1 we show how to go from logic to groups. The more difficult direction from groups to logic is done in Theorem 5.2.

Theorem 5.1. *Let ϕ be a formula of $Maj[<]$ (resp. $Maj+Sq[<]$) and $\mathcal{V}$ be a set of first-order variables. Then there is a group $(G,\mathfrak{G}) \in \mathbf{M}_<$ (resp. $\mathbf{M}_{<,Sq}$) and a monoid morphism $h : (\Sigma \times 2^{\mathcal{V}})^* \to G$ fullfilling*

$$L_{\phi,\mathcal{V}} = h^{-1}(\mathcal{G}) \cap (\Sigma^* \otimes \mathcal{V})$$

for some $\mathcal{G} \in \mathcal{B}(\mathfrak{G})$.

Theorem 5.2. *Let $(G,\mathfrak{G}) \in \mathbf{M}_<$ (resp. $\mathbf{M}_{<,Sq}$), $\mathcal{V}$ be a set of variables and let h be monoid morphism $h : (\Sigma \times 2^{\mathcal{V}})^* \to G$. Then there is for each $\mathcal{G} \in \mathfrak{G}$ a $Maj[<]$-formula (resp. $Maj + Sq[<]$-formula) with free variables $\mathcal{V}$ that defines the language $L = h^{-1}(\mathcal{G}) \cap (\Sigma^* \otimes \mathcal{V})$.*

Corollary 5.3. *$\mathcal{L}(Maj[<]) = H^{-1}(\mathbf{M}_<)$ and $TC^0 = H^{-1}(\mathbf{M}_{<,Sq})$.*

6 Proof of theorem 5.1

This proof is done by induction on the term structure of ϕ.

(1) $\phi = Q_a(x)$ for some $a \in \Sigma$:
We recognize the set $L := \{(a_1,S_1)\dots(a_n,S_n) \in (\Sigma \times 2^{\mathcal{V}})^* \mid \text{at position x there is an } a\}$ by $(\mathbb{Z}, \mathfrak{Z}_<)$ with the morphism

$$h :\ (\Sigma \times 2^{\mathcal{V}})^* \to \mathbb{Z},\ (\alpha,S) \mapsto \begin{cases} 1 & \text{if } \alpha = a \text{ and } x \in S, \\ 0 & \text{otherwise.} \end{cases}$$

Then $L = h^{-1}(\mathbb{Z}^+)$, thus $L_{\phi,\mathcal{V}} = h^{-1}(\mathbb{Z}^+) \cap (\Sigma^* \otimes \mathcal{V})$.

(2) $\phi = x < y$:
We recognize the set L of all words in $(\Sigma \times 2^{\mathcal{V}})^*$ such that x is positioned before y by the finitely typed block product $(G,\mathfrak{G}) = (\mathbb{Z},\mathfrak{Z}_<) \,\square\, (\mathbb{Z},\mathfrak{Z}_<) = V * *\mathbb{Z}$. We use the morphism $h : (\Sigma \times 2^{\mathcal{V}})^* \to G$ given by

$$h(a,S) := \begin{cases} (\mathbf{e},0) & \text{if } \{x,y\} \cap S = \emptyset \\ (\mathbf{e},1) & \text{if } x \in S \text{ and } y \notin S \\ (f_>,0) & \text{if } y \in S, \end{cases}$$

where $\mathbf{e}(x,y) = 0$ for all $x,y \in \mathbb{Z}$, and $f_>(x,y) = 1$ if $x > 0$ and $f_>(x,y) = 0$ otherwise. Obviously $f_> \in V^{(1)} \subseteq V$.
Set $\mathcal{G} := \{(f,n) \in V * *\mathbb{Z} \mid f(0,0) \in \mathbb{Z}^+\}$. Then $L = h^{-1}(\mathcal{G})$, thus $L_{\phi,\mathcal{V}} = h^{-1}(\mathcal{G}) \cap (\Sigma^* \otimes \mathcal{V})$.

Note: Let $(a_1,S_1)\dots(a_n,S_n)$ be a $\mathcal{V}$-structure and $(f_i,m_i) = h(a_i,S_i)$. Then $h(a_1,S_1)\cdots h(a_n,S_n) \in L$ iff $(f_1 + m_1 \cdot f_2 + \dots + m_1 \dots m_{n-1} \cdot f_n)(0,0) > 0$. But this holds precisely when there exist $i,j,\ i<j$ such that $m_i = 1$ (that is $x \in S_i, y \notin S_i$) and $f_j = f_>$ (that is $y \in S_j$).

(3) $\phi = \phi_1 \wedge \phi_2$:
By induction $L_{\phi_i,\mathcal{V}} = h_i^{-1}(\mathcal{G}_i) \cap (\Sigma^* \otimes \mathcal{V})$, for suitable $h_i, \mathcal{G}_i$ and $i = 1,2$, hence $L_{\phi,\mathcal{V}} = h^{-1}(\mathcal{G}) \cap (\Sigma^* \otimes \mathcal{V})$ where $h = h_1 \times h_2,\ G = G_1 \times G_2$ and $\mathcal{G} = \mathcal{G}_1 \times \mathcal{G}_2$.

(4) $\phi = \neg\psi$:
By induction $L_\psi = h^{-1}(\mathcal{G}) \cap (\Sigma^* \otimes \mathcal{V}),\ h,\ \mathcal{G}$ suitable. Hence $L_{\phi,\mathcal{V}} = h^{-1}(\overline{\mathcal{G}}) \cap (\Sigma^* \otimes \mathcal{V})$ where $\overline{\mathcal{G}} = G \setminus \mathcal{G}$.

(5) $\phi = Maj\ x\ \psi$:
By induction there exist $(H,\mathcal{H}) \in \mathbf{M}_<$ and a monoid morphism $h_0 : (\Sigma \times 2^{\mathcal{V}\cup\{x\}})^* \to H$ with $L_{\psi,\mathcal{V}\cup\{x\}} = h_0^{-1}(\mathcal{H}) \cap (\Sigma^* \otimes \mathcal{V} \cup \{x\}) \subseteq (\Sigma \times 2^{\mathcal{V}\cup\{x\}})^*$, for a suitable extended type $\mathcal{H} \in \mathcal{B}(\mathfrak{H})$.
We set $(G,\mathfrak{G}) := (\mathbb{Z},\mathfrak{Z}_<) \,\square\, (H,\mathfrak{H}) = V * *H$ and define a morphism $h : (\Sigma \times 2^{\mathcal{V}})^* \to G$ by $h(a,S) := (f_{(a,S)}, h_0(a,S))$ where

$$f_{(a,S)}(m_1,m_2) = \begin{cases} 1 & \text{if } m_1 \cdot h_0(a, S \cup \{x\}) \cdot m_2 \in \mathcal{H}, \\ -1 & \text{otherwise.} \end{cases}$$

Then $f_{(a,S)}$ is in $V^{(2)} \subseteq V$. Set $\mathcal{G} := \{(f,m) \in V * *H \mid f(e_H,e_H) > 0\}$. Then $\mathcal{G}$ is a type of G and $L_{\phi,\mathcal{V}} = h^{-1}(\mathcal{G}) \cap (\Sigma^* \otimes \mathcal{V})$.

Note: An input word $(a_1,S_1)\cdots(a_n,S_n) \in \Sigma^* \otimes \mathcal{V}$ is a model for ϕ iff the number of positions $1 \le i \le n$ fulfilling

$$h_0((a_1,S_1)\cdots(a_{i-1},S_{i-1})(a_i,S_i \cup \{x\})(a_{i+1},S_{i+1})\cdots(a_n,S_n)) \in \mathcal{H}$$

is larger than that of nonfulfilling positions. By definition we have

$$h_0((a_1,S_1)\cdots(a_{i-1},S_{i-1})(a_i,S_i\cup\{x\})(a_{i+1},S_{i+1})\cdots(a_n,S_n))\in\mathcal{H}$$

iff

$$h_0((a_1,S_1)\cdots(a_{i-1},S_{i-1}))\cdot f_{(a_i,S_i)}(e_H,e_H)\cdot h_0((a_{i+1},S_{i+1})\cdots(a_n,S_n))$$
$$=f_{(a_i,S_i)}((h_0(a_1,S_1)\cdots(a_{i-1},S_{i-1})),h_0((a_{i+1},S_{i+1})\cdots(a_n,S_n)))=1.$$

Since $h((a_1,S_1)\cdots(a_n,S_n))=(f,h_0((a_1,S_1)\cdots(a_n,S_n)))$ with

$$f(e_H,e_H)=$$
$$\sum_{i=1}^{n} f_{(a_i,S_i)}(h_0((a_1,S_1)\cdots(a_{i-1},S_{i-1})),h_0((a_{i+1},S_{i+1})\cdots(a_n,S_n)))$$
$$=\sum_{\text{“}i\text{ fulfilling }\psi\text{”}}1-\sum_{\text{“}i\text{ not fulfilling }\psi\text{”}}1$$

we have $L_{\phi,\mathcal{V}}=h^{-1}(\mathcal{G})\cap(\Sigma^*\otimes\mathcal{V})$.

The assertion follows in the $Maj[<]$-case. Since $H^{-1}(\mathbf{M}_<)\subseteq H^{-1}(\mathbf{M}_{<,Sq})$ it remains to consider the situation

(6) $\phi=Sq\ x\ \psi$:

We define the morphism h as for $Maj\ x\ \psi$, but now mapping nonfulfilling positions to 0 instead of -1. This causes that we now simply count the number of satisfying positions ignoring the unsatisfying ones. Hence $L_{\phi,\mathcal{V}}=h^{-1}(\mathbb{S})\cap(\Sigma^*\otimes\mathcal{V})$.

An Example

In the following we give as example the simulation of modular counting by groups in $\mathbf{M}_<$. We deal with the formula $\chi:=MajxMajyx\le y$ which accepts the set of all words of odd length. Since we do not use $Q_a(\cdot)$-predicates, lets simply take $\Sigma:=\{a\}$ to be singleton. The syntactic monoid of L_χ is the cyclic group with two elements.

As a first step, we simulate the formula $\psi:=x\le y$ with the set of free variables $\mathcal{V}:=\{x,y\}$. Thus we begin with the group $(G_\psi,\mathfrak{G}_\psi)=(\mathbb{Z},\mathfrak{Z}_<)\,\square\,(\mathbb{Z},\mathfrak{Z}_<)=V_\psi**\mathbb{Z}$. We simulate $\le$ (and not $<$) thus we define the morphism $h_\psi:(\{a\}\times 2^{\mathcal{V}})^*\to G_\psi$ by

$$h_\psi(a,S):=\begin{cases}(\mathbf{e_1},0) & \text{if } S=\emptyset\\ (\mathbf{e_1},1) & \text{if } S=\{x\}\\ (f_>,0) & \text{if } S=\{y\}\\ (\mathbf{e_1},1)(f_>,0)=(1\cdot f_>,1) & \text{if } S=\{x,y\}\end{cases}.$$

where $f_>(x,y)=1$ if $x>0$ and 0 otherwise, and $\mathbf{e_1}$ is the constant zero mapping from $\mathbb{Z}\times\mathbb{Z}$ to $\mathbb{Z}$. As acceptance type we use the set $\mathcal{G}_\psi:=\{(f,n)\mid f(0,0)>0\}$. Then the set of all $\mathcal{V}$-structures contained in $h_\psi^{-1}(\mathcal{G}_\psi)$ is precisely the set of words where variable x is not positioned after variable y.

In the second step we simulate the formula $\phi:=Majy\psi$ which has the set of free variable $\mathcal{V}':=\{x\}$. We set $(G_\phi,\mathfrak{G}_\phi):=(\mathbb{Z},\mathfrak{Z}_<)\,\square\,(G_\psi,\mathfrak{G}_\psi)=V_\phi**\mathbb{Z}$, and define maps $g_{(a,\emptyset)}$ and $g_{(a,\{x\})}$ from $G_\psi\times G_\psi$ to $\mathbb{Z}$ by

$$g_{(a,\emptyset)}((f_1,n_1),(f_2,n_2)):=\begin{cases}+1 & \text{if } (f_1,n_1)h_\psi(a,\{y\})(f_2,n_2)\in\mathcal{G}_\psi\\ -1 & \text{otherwise}\end{cases}$$

and

$$g_{(a,\{x\})}((f_1,n_1),(f_2,n_2))):=\begin{cases}+1 & \text{if } (f_1,n_1)h_\psi(a,\{y,x\})(f_2,n_2)\in\mathcal{G}_\psi\\ -1 & \text{otherwise}\end{cases}.$$

Thus $g_{(a,\emptyset)}((f_1,n_1),(f_2,n_2))>0$ iff $f_1(0,n_2)+f_>(n_1,n_2)+f_2(n_1,0)>0$ and $g_{(a,\{x\})}((f_1,n_1),(f_2,n_2))>0$ iff $f_1(0,n_2+1)+f_>(n_1+1,n_2)+f_2(n_1+1,0)>0$.

For the morphism $h_\phi:(\{a\}\times 2^{\{x\}})^*\to G_\phi$ defined by $h_\phi(a,\emptyset):=(g_{(a,\emptyset)},(\mathbf{e_1},0))$ and $h_\phi(a,\{x\}):=(g_{a,\{x\}},(\mathbf{e_1},1))$ and the acceptance type $\mathcal{G}_\phi:=\{(g,(f,n))\mid g((\mathbf{e_1},o),(\mathbf{e_1},0))>0\}$ the set of all $\{x\}$-structures lying in $h_\phi^{-1}(\mathcal{G}_\phi)$ is exactly the set of words where the variable x is positioned in the first half of the word and this is the language accepted by formula ϕ.

In the final step we simulate the whole formula χ which is $Majx\phi$ and has no free variables.

We set $(G_\chi, \mathfrak{G}_\chi) := (\mathbb{Z}, \mathfrak{Z}_<) \square (G_\phi, \mathfrak{G}_\phi) = V_\chi * *\mathbb{Z}$. Define $h_\chi : \{a\}^* \to G_\chi$ by $h_\chi(a, \emptyset) = (F_{(a,\emptyset)}, h_{\phi(a,\emptyset)})$ where $F_{(a,\emptyset)} : G_\phi \times G_\phi \to \mathbb{Z}$ is defined as

$$F_{(a,\emptyset)}(m_1, m_2) = \begin{cases} 1 & \text{if } m_1 \cdot h_\phi(a, \{x\}) \cdot m_2 \in \mathcal{G}_\phi \\ -1 & \text{otherwise} \end{cases}$$

We claim that $L_\chi = h^{-1}(\mathcal{G}_\chi)$ for $\mathcal{G}_\chi := \{(F, (g, (f, n))) \in V_\chi * *G_\phi \mid F((\mathbf{e_2}, (\mathbf{e_1}, 0)), (\mathbf{e_2}, (\mathbf{e_1}, 0))) > 0\}$ where $\mathbf{e_2}$ denotes the constant zero mapping from $G_\psi \times G_\psi$ to $\mathbb{Z}$. Set for the sake of brevity $\alpha = g_{(a,\emptyset)}$, $\beta = g_{(a,\{x\})}$ and $\gamma = F_{(a,\emptyset)}$. Then by definition $\gamma((g_1, (f_1, n_1)), (g_2, (f_2, n_2))) > 0$ iff $g_1((\mathbf{e_1}, 0), (1 \cdot f_2, n_2 + 1)) + \beta((f_1, n_1), (f_2, n_2)) + g_2((f_1 \cdot 1, n_1 + 1), (\mathbf{e_1}, 0)) > 0$. In particular for $m = (\alpha, (\mathbf{e}_1, 0))$ we have $\gamma(m^{i-1}, m^{n-i}) > 0$ iff $-(i-1) + \beta((\mathbf{e_1}, 0), (\mathbf{e_1}, 0)) + (n - i) > 0$. Since

$$\pi_1(h_\chi(a^n))(e_{G_\chi}, e_{G_\chi}) = \sum_{i=1}^{n} \gamma(m^{i-1}, m^{n-i}) = \lfloor \frac{n+1}{2} \rfloor - n + \lfloor \frac{n+1}{2} \rfloor$$

this yields $h^{-1}(\mathcal{G}_\chi) = (aa)^* a$ which is the language accepted by formula χ.

Remark 6.1. It is also possible to accept the language $L = \{a^n \mid n \equiv 1 \mod (k+1)\}$ for an arbitrary natural number k by the group G_χ of the above example. It is readily seen that this can be done by setting $h(a) := (\gamma, (\alpha, (\mathbf{e_1}, 0)))$ where β is as in the above example and where γ and α are essentially defined as above but now setting the positive value of α to k and the negative value of γ to $-k$, and using $\mathcal{G} = \mathcal{G}_\chi$ of the above example as acceptance type.

7 Proof of theorem 5.2

Group quantifiers turn out to be a key ingredient to prove the converse.

Definition 7.1 (Group quantifier). Let G be a group, $\mathcal{G} \subseteq G$, $d \in \mathbb{N}$, and $\phi_1(x), \ldots, \phi_d(x)$ formulae. Further let $m : 2^d \to G$ be a function such that $e_G \in \mathrm{im}(m)$. Then the following is also a formula:

$$w \models \Gamma^{G,\mathcal{G},m}\, x\, \langle \phi_1(x), \ldots, \phi_d(x) \rangle$$

which is true iff:

$$\left(\prod_{i=1}^{n} m(w \models \phi_1(i), \ldots, w \models \phi_d(i)) \right) \in \mathcal{G}.$$

Concerning the mapping m we identify $false$ with 0 and $true$ with 1.

This definition is an extension of the group quantifier over finite groups [2]. The difference to the finite case is that the map m does not need to be surjective and the accepting set cannot be any element but has to be a type. We add the map m to the notation of the group quantifier since it strongly influences its power.

Definition 7.2. $(G, \mathfrak{G})$ is definable in $\mathcal{Q}[\mathcal{X}]$ (Notation: $(G, \mathfrak{G}) \text{ ᴅ } \mathcal{Q}[\mathcal{X}]$) iff $\forall d \in \mathbb{N}$, $\forall m : 2^d \to G$ and $\forall \mathcal{G} \in \mathfrak{G}$:$\Gamma^{G,\mathcal{G},m}$ is definable in $\mathcal{Q}[\mathcal{X}]$.

Example. We can write the $\exists$ quantifier as a group quantifier over $\mathbb{Z}$.

$$\exists x\ \phi(x) = \Gamma^{\mathbb{Z},\mathbb{Z}^+,m}\, x\, \langle \phi(x) \rangle$$

where $m(false) = 0$ and $m(true) = 1$.

Example. We can write the Maj quantifier as a group quantifier over $\mathbb{Z}$.

$$Maj\ x\ \phi(x) = \Gamma^{\mathbb{Z},\mathbb{Z}^+,m}\, x\, \langle \phi(x) \rangle$$

where $m(false) = -1$ and $m(true) = 1$. This does not quite meet the definition since $0 \notin im(m)$, but we could easily extend the definition to:

$$Maj\ x\ \phi(x) = \Gamma^{\mathbb{Z},\mathbb{Z}^+,m'}\, x\, \langle \phi(x), false \rangle$$

where $m'(false, false) = -1$ and $m'(true, false) = 1$ and to meet the definition $m'(false, true) = m'(true, true) = 0$. This example demonstrates that the requirement $0 \in im(m)$ is not substantial.

Definition 7.3 (Relative group quantifier). Let the notation be as in 7.1 and let p, q be two free variables. Then

$$w \models \Gamma_{p..q}^{G,\mathcal{G},m}\ x\ \langle \phi_1(x), \ldots, \phi_d(x)\rangle$$

is a formula which is true iff:

$$\left(\prod_{i=p}^{q} m(w \models \phi_1(i), \ldots, w \models \phi_d(i))\right) \in \mathcal{G}.$$

Lemma 7.4. *If a group quantifier is definable in $\mathcal{Q}[<, \mathcal{X}]$ then the relative group quantifier is definable in $\mathcal{Q}[<, \mathcal{X}]$, as well.*

Proof. We show that we can simulate each relative group quantifier $\Gamma_{p..q}^{G,\mathcal{G},m}\ x\ \langle \phi_1(x), \ldots, \phi_d(x)\rangle$. By definition there is a $\langle v_1, \ldots, v_d\rangle$ such that $m(\langle v_1, \ldots, v_d\rangle) = e_G$. Let p, q be two free variables. We define

$$\phi_i'(x) = (x < p \wedge v_i) \vee (x > q \wedge v_i) \vee (p \leq x \leq q \wedge \phi_i(x))$$

Then $m(\langle \phi_1'(x), \ldots, \phi_d'(x)\rangle) = e_G$ if $x < p$ or $x > q$ and equals $\phi_i(x)$ otherwise. So

$$\Gamma_{p,q}^{G,\mathcal{G},m}\ x\ \langle \phi_1(x), \ldots, \phi_d(x)\rangle = \Gamma^{G,\mathcal{G},m}\ x\ \langle \phi_1'(x), \ldots, \phi_d'(x)\rangle$$

□

If the group quantifier is definable in a logic $\mathcal{Q}[\mathcal{X}]$ then all inverse morphic images of this group belong to $\mathcal{L}(\mathcal{Q}[\mathcal{X}])$:

Lemma 7.5. $(G, \mathfrak{G}) \triangleright \mathcal{Q}[\mathcal{X}]$ *implies that for each morphism $h : \Sigma^* \to G$, $h^{-1}(\mathcal{G}) \in \mathcal{L}(\mathcal{Q}[\mathcal{X}])$ for all $\mathcal{G} \in \mathfrak{G}$.*

Proof. Let h and $\mathcal{G}$ be as above, $\Sigma = \{\alpha_1, \ldots, \alpha_r\}$ and set $\phi_i(x) = Q_{\alpha_i}(x)$ for $i = 1, \ldots, r$. We define a mapping $m : 2^{|\Sigma|} \to G$ by

$$m(false, \ldots, false, true, false, \ldots, false) := h(\alpha_i)$$

where i is the position of *true*. Then $w \models \Gamma^{G,\mathcal{G},m} < \vec{\phi}(x)$ iff $h(w) \in \mathcal{G}$ for every $w \in \Sigma^*$ □

We prove by induction on the block structure that the elements of $\mathbf{M}_<$ and $\mathbf{M}_{<,Sq}$ are definable in $Maj[<]$ and $Maj + Sq[<]$.

We use *extended arithmetic* to have vectors of variables $\vec{x} = \langle x_1, x_2, \ldots, x_r\rangle, \vec{y} = \langle y_1, y_2, \ldots, y_r\rangle \ldots$ to encode numbers between 1 and n^r, where n is the length of the input word. We assume that there are no two different encodings of the same number. Then there are $FO[+]$ and hence $Maj[<]$ formulae $\phi_<(\vec{x}, \vec{y})$ and $\phi_+(\vec{x}, \vec{y}, \vec{z})$ expressing the fact that the number encoded by $\vec{x}$ is smaller than that encoded by $\vec{y}$ resp. that the sum of the number encoded by $\vec{x}$ and $\vec{y}$ coincides with that of $\vec{z}$ ([7][Theorem2.5]). It should be mentioned that these simulations hold only for sufficiently large input size. Thus there might be a finite number of exception. Formally, this means for describing languages by formulae that we would have to explicitly code these finitely many exception words in the constructed formulae by some sort of table look-up. But to ease the handling we only consider the case of sufficiently large input size.

Proposition 7.6.

$$(\mathbb{Z}, \{\mathbb{Z}^+, \mathbb{Z}_0^-\}) \triangleright Maj[<]$$

Proof. Let $\Gamma^{\mathbb{Z},\mathbb{Z}^+,m}\ x\ \langle \phi_1(x), \ldots, \phi_d(x)\rangle$, then there are finitely many possible $\{0,1\}$-vectors $\vec{\phi}(x)$, hence there are finitely many possible numbers $m(\vec{\phi}(x))$. For each binary vector $v = \langle v_1, \ldots, v_d\rangle$ we let:

$$\psi_v = \exists^{=c_v}\ x\ \left(\bigwedge_{i=1}^{d} \phi_i(x) \leftrightarrow v_i\right)$$

where c_v is a free variable.

We now construct the formula simulating the group quantifier. This formula multiplies c_v by $|m(v)|$ then it adds all products with $m(v) < 0$ and all products with $m(v) > 0$ and compares the sums to decide if the group quantifier is fulfilled.

$$\exists\, c_{\langle 0,0,\ldots,0\rangle}, \ldots, c_{\langle 1,1,\ldots,1\rangle} \bigwedge_v \psi_v \wedge \left(\sum_{m(v)<0} (|m(v)| \cdot c_v) < \sum_{m(v)>0} (|m(v)| \cdot c_v)\right)$$

Note that we use extended arithmetic in order to compute the involved sums. There are only finitely many c_v so the product c_v by $|m(v)|$ is the multiplication with a constant hence expressible by iterated addition and the sums have only a constant number of terms.

In this way we can construct a formula expressing the group quantifier with type set $\mathbb{Z}^+$. For the type $\mathbb{Z}_0^-$ we simply negate the formula. □

Proposition 7.7.

$$(\mathbb{Z}, \{\mathbb{S}, \overline{\mathbb{S}}\}) \triangleright Maj + Sq[<].$$

Proof. We first observe that it is possible to define an extended version $Sq\ \vec{x}$ of the square quantifier $Sq\ x$ just in the way Theorem 2.3.e is proven in [7].

In the following formula $\vec{x}$ ranges over the difference between the positive and the negative terms computed by the group quantifier.

$$\exists\, c_{\langle 0,0,\ldots,0\rangle}, \ldots, c_{\langle 1,1,\ldots,1\rangle} \bigwedge_v \psi_v \wedge Sq\ \vec{x} \left(\sum_{m(v)<0} (|m(v)| \cdot c_v) \leq \vec{x} < \sum_{m(v)>0} (|m(v)| \cdot c_v) \right)$$

Since the representation of number by vectors of variables is unique the number of fulfilling choices of $\vec{x}$ coincides with $\sum_{m(v)>0} (|m(v)| \cdot c_v) - \sum_{m(v)<0} (|m(v)| \cdot c_v) = \sum_v (m(v) \cdot c_v)$. This formula is now equivalent to the group quantifier using the accepting set $\mathbb{S}$. Again with negation we can simulate the group quantifier with the accepting set $\overline{\mathbb{S}}$. □

Corollary 7.8.

$$(\mathbb{Z}, \{\mathbb{S}, \mathbb{Z}^+ \setminus \mathbb{S}, \mathbb{Z}_0^-\}) \triangleright Maj + Sq[<].$$

Proof. The preceding propositions combined with lemma 4.2 yields the desired result. □

We now start with the induction. First, we simulate Cartesian products. This simply reduces to propositional conjunction:

Proposition 7.9. *Let $\mathcal{Q}[\mathcal{X}]$ be any logic. Assume $(G, \mathfrak{G}), (H, \mathfrak{H}) \triangleright \mathcal{Q}[\mathcal{X}]$, then $(G, \mathfrak{G}) \times (M, \mathfrak{M}) \triangleright \mathcal{Q}[\mathcal{X}]$.*

Proof. Let $\mathcal{A} = \mathcal{G} \times \mathcal{H}$ be a type of $\mathfrak{G} \times \mathfrak{H}$. Let $m : 2^d \to G \times H, v \mapsto (m_g(v), m_h(v))$, and $\phi_1(x), \ldots, \phi_d(x)$ be formulae in $\mathcal{Q}[\mathcal{X}]$.

$$\Gamma^{G\times H, A, m}\ x\ \langle \phi_1(x), \ldots, \phi_d(x)\rangle$$

iff

$$\Gamma^{G,\mathcal{G},m_g}\ x\ \langle \phi_1(x), \ldots, \phi_d(x)\rangle \wedge \Gamma^{H,\mathcal{H},m_h}\ x\ \langle \phi_1(x), \ldots, \phi_d(x)\rangle$$

□

Finally we show that groups constructed by the block product also belong to $Maj[<]$ respectively $Maj + Sq[<]$.

Theorem 7.10. *Let $\mathcal{Q}[\mathcal{X}]$ be any logic.*
Assume $(G, \mathfrak{G}), (H, \mathfrak{H}) \triangleright \mathcal{Q}[\mathcal{X}]$, then $(G, \mathfrak{G}) \square (H, \mathfrak{H}) \triangleright \mathcal{Q}[\mathcal{X}]$.

Proof. Set $(W, \mathfrak{W}) = (G, \mathfrak{G}) \square (H, \mathfrak{H})$ and let $\mathcal{W} \in \mathfrak{W}$ and d, $m : 2^d \to W$, $\phi_1(x), \ldots, \phi_d(x)$ as in 7.1. We need to show that $\Gamma^{W,\mathcal{W},m} x\ \vec{\phi}(x)$ is definable in $\mathcal{Q}[\mathcal{X}]$.

We need some notation first. For an arbitrary but fixed input w of length n we have $m(\vec{\phi}(\cdot)) : \{1, \ldots, n\} \to W, i \mapsto m(w \models \phi_1(i), \ldots, \phi_d(i)) =: (f_i, h_i)$. $\mathcal{W}$ is a type of $\mathfrak{W}$, thus $\mathcal{W} = \{(f, h) \in W \mid f(e_H, e_H) \in \mathcal{G}\}$ for a suitable type $\mathcal{G}$ of G.

$$\begin{aligned} w \models \Gamma^{W,\mathcal{W},m} x\ \vec{\phi}(x) &\Leftrightarrow \prod_{i=1}^{n} m(w \models \vec{\phi}(i)) \in \mathcal{W} \\ &\Leftrightarrow \pi_1 \left(\prod_{i=1}^{n} m(w \models \vec{\phi}(i)) \right) (e_H, e_H) \in \mathcal{G} \\ &\Leftrightarrow \sum_{i=1}^{n} f_i(h_1 \cdots h_{i-1}, h_{i+1} \cdots h_n) \in \mathcal{G} \end{aligned}$$

In the following we need to show that we can construct formulae $\vec{\psi}(x)$ and a mapping $m' : 2^{d'} \to H$ such that $m'(\vec{\psi}(i)) = f_i(h_1 \cdots h_{i-1}, h_{i+1} \cdots h_n)$. Then we can use the group quantifier for G and get:

$$\Gamma^{G,\mathcal{G},m'}\ x\ \vec{\psi}(x)$$

iff

$$\Gamma^{W,\mathcal{W},m} x\ \vec{\phi}(x).$$

We already have the mapping $\pi_1 \circ m$ that determines together with $\vec{\phi}(x)$ the function f_i at any position. There are only finitely many different function f_i since the image of $\pi_1 \circ m$ is finite. So lets fix a function f_i. By 3.3(c) we have $f_i = \sum_{j \in J_i} f_{i,j}$ for $f_{i,j} \in V^{(1)} \cup V^{(2)}$ and we can determine the value of f_i if we have formulae for cases (i)-(iii) of 3.3(c).
(i) Let $s \in \{s_j \mid j \in J_i\}$ and $\mathcal{H} \in \mathfrak{H}$. We define $\phi_{d+1}(x) = x = i$ with a free variable i and $m''_s(\vec{v}, false) := \pi_2 \circ m(\vec{v})$ and $m''_s(\vec{v}, true) := s$ then

$$w \models \Gamma^{H,\mathcal{H},m''_s}_{1..j}\ x\ \langle \vec{\phi}(x), \phi_{d+1}(x) \rangle$$

iff

$$h_1 \cdots h_{i-1} \cdot s \in \mathcal{H}$$

Hence we can construct a finite number of formulae to determine the case of clause (i). **(ii)** Similar to case (i).

(iii) With the same definitions as above:

$$w \models \Gamma^{H,\mathcal{H},m''_s}\ x\ \langle \vec{\phi}(x), \phi_{d+1}(x) \rangle$$

iff

$$h_1 \cdots h_{i-1} \cdot s \cdot h_{i+1} \cdots h_n \in \mathcal{H}$$

Hence we can determine the arguments of f_i up to an equivalence class that results in the same value of f_i and use a finite case differentiation to build the formulae $\vec{\psi}(x)$ and the mapping m'. This completes the proof of Theorem 5.2. □

Corollary 7.11. $H^{-1}(\mathbf{M}_<) \subseteq \mathcal{L}(Maj[<])$.

Corollary 7.12. $H^{-1}(\mathbf{M}_{<,Sq}) \subseteq \mathcal{L}(Maj + Sq[<])$.

Discussion

We characterized the languages in $\mathcal{L}(Maj[<])$ as inverse morphic images of the elements in $\mathbf{M}_<$. By using the square quantifier it was possible to represent uniform TC^0 in the same way. A corresponding characterization is possible for other complexity classes like AC^0 or ACC^0, as well. A minor problem is the simulation of addition. This has to be done by a specific group as initial building block which is not allowed to be used in the induction step.

The next step should be to prove lower bounds. This should be possible for classes like $\mathcal{L}(Maj[<])$ or $FO[+]$-uniform ACC^0. Here the representation in terms of groups might be helpful: when transforming a circuit class into a logical class like $\mathcal{L}(\mathcal{Q}(\mathcal{X}))$ we often loose a reasonable notion of depth. For example, we know that $Maj[<]$ can do counting modulo k for each fixed k, but the depth of the corresponding formula seems inherently to increase with growing k. This is not the case when using groups in $\mathbf{M}_<$ as seen in the example in section 6. Thus it might be a reasonable task to try to correlate the (block) depth notion of a group or monoid to the depth of a circuit.

Another task is to come up with an algebraic theory of finitely typed monoids and groups. For instance, it might be a reasonable notion to call a finitely typed monoid $(M, \mathfrak{M})$ *pseudo-aperiodic* if for all $m \in M$ there is a positive integer k such that the element m^k has the same type as m^{k+1}, i.e. we have

$$\forall \mathcal{M} \in \mathfrak{M}\ \forall m \in M\ \exists k \in \mathbb{Z}^+\ m^k \in \mathcal{M} \text{ iff } m^{k+1} \in \mathcal{M}.$$

It is easy to see that all elements in $\mathbf{M}_<$ of depth 2 or less are pseudo-aperiodic, while the example in section 6 demonstrates that all finite cyclic groups are divisors of groups in $\mathbf{M}_<$.

Acknowledgments

We thank Howard Straubing, Pascal Tesson, and Denis Therien for valuable discussions on this topic.

References

[1] D.A. Barrington. Bounded-width polynomial-size branching programs can recognize exactly those languages in NC^1. *J. Comp. System Sci.*, 38:150–164, 1989.

[2] D.A. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *J. Comp. System Sci.*, 41:274–306, 1990.

[3] D.A. Barrington, H. Straubing, and D. Therien. Nonuniform Automata over Groups. *Information and Computation*, 89:109–132, 1990.

[4] K.-J. Lange. Some results on majority quantifiers over words. In *Proc. of the 19th IEEE Conference on Computational Complexity*, pages 123–129, 2004.

[5] C. Lautemann, P. MCKenzie, T. Schwentick, and H. Vollmer. The descriptive complexity approach to logcfl. *J. Comp. System Sci.*, 62:629–652, 2001.

[6] M. Lawson. *Finite Automata*. Chapman & Hall/CRC, 2004.

[7] N. Schweickard. On the Expressive Power of First-Order Logic with Built-In Predicates. Dissertation, Universität Mainz, 2001.

[8] H. Straubing. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhäuser, 1994.