

Characterizing NC^1 with Typed Monoids

Anonymous author

Anonymous affiliation

Anonymous author

Anonymous affiliation

Abstract

Krebs et al. (2007) gave a characterization of the complexity class TC^0 as the class of languages recognized by a certain class of typed monoids. The notion of typed monoid was introduced to extend methods of algebraic automata theory to infinite monoids and hence characterize classes beyond the regular languages. We advance this line of work beyond TC^0 by giving a characterization of NC^1 . This is obtained by first showing that NC^1 can be defined as the languages expressible in an extension of first-order logic using only unary quantifiers over regular languages. The expressibility result is a consequence of a general result showing that finite monoid multiplication quantifiers of higher dimension can be replaced with unary quantifiers in the context of interpretations over strings.

2012 ACM Subject Classification Replace ccsdesc macro with valid one

Keywords and phrases algebraic automata theory, circuit complexity, descriptive complexity, typed monoids, semigroups, generalized quantifiers

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Acknowledgements Anonymous acknowledgements



© Anonymous author(s);

licensed under Creative Commons License CC-BY 4.0

42nd Conference on Very Important Topics (CVIT 2016).

Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Much work in theoretical computer science is concerned with studying classes of formal languages, whether these are classes defined in terms of grammars and expressions, such as the class of regular or context-free languages, or whether they are *complexity classes* such as P and NP, defined by resource bounds on machine models. Indeed, the distinction between these are largely historical as most classes of interest admit different characterizations based on machine models, grammars, logical definability, or algebraic expressions. The class of regular languages can be characterized as the languages accepted by linear-time-bounded single-tape Turing machines [9] while P can be characterized without reference to resources as the languages recognized by multi-head two-way pushdown automata [6]. The advantage of the variety of characterizations is, of course, the fact that these bring with them different mathematical toolkits that can be brought to the study of the classes.

The class of regular languages has arguably the richest theory in this sense of diversity of characterizations. Virtually all students of computer science learn of the equivalence of deterministic and nondeterministic finite automata, regular languages and linear grammars and many also know that the regular languages are exactly those definable in monadic second-order logic with an order predicate. Perhaps the most productive approach to the study of regular languages is via their connection to finite monoids. Every language L has a syntactic monoid, which is finite if, and only if, L is regular. Moreover, closure properties of classes of regular languages relate to natural closure properties of classes of monoids, via Eilenberg's Correspondence Theorem [8]. This, together with the tools of *Krohn-Rhodes theory*, gives rise to *algebraic automata theory*—which leads to the definition of natural subclasses of the class of regular languages, to effective decision procedures for automata recognizing such classes, and to separation results.

When it comes to studying computational complexity, we are mainly interested in classes of languages richer than just the regular languages. Thus the syntactic monoids of the languages are not necessarily finite any longer and the extensive tools of Krohn-Rhodes theory are not available to study them. Nonetheless, some attempts have been made to extend the methods of algebraic automata theory to classes beyond the regular languages. Most significant is the work of Krebs and collaborators [2, 3, 11, 10, 5], which introduces the notion of *typed monoids*. The idea is to allow for languages with infinite syntactic monoids, but limit the languages they recognize by associating with them a finite collection of types. This allows for the formulation of a version of Eilenberg's Correspondence theorem associating closure properties on classes of typed monoids with corresponding closure properties of classes of languages. In particular, this implies that most complexity classes of interest can be uniquely characterized in terms of an associated class of typed monoids [3]. An explicit description of the class characterizing DLOGTIME-uniform TC^0 is given in [11, 10]. This is obtained through a general method which allows us to construct typed monoids corresponding to *unary quantifiers* defined from specific languages [10] (see also Theorem 22 below).

In this paper, we extend this work to obtain a characterization of DLOGTIME-uniform NC^1 as the class of languages recognized by the collection of typed monoids obtained as the closure under *ordered strong block products* of three typed monoids: the group of integers with types for positive and negative integers; the group of natural numbers with types for the square numbers and non-square numbers; and a finite non-solvable group such as S_5 with a type for each subset of the group. Full definitions of these terms follow below. Our result is obtained by first characterizing DLOGTIME-uniform NC^1 in terms of logical definability in an extension of first-order logic with only unary quantifiers. It is known that any regular

language whose syntactic monoid is a non-solvable group is complete for NC^1 under reductions definable in first-order logic with arithmetic predicates ($\text{FO}(+, \times)$) [1]. From this, we know we can describe NC^1 as the class of languages definable in an extension of $\text{FO}(+, \times)$ with quantifiers (of arbitrary arity) associated with the regular language corresponding to the word problem for S_5 . Our main technical contribution is to show that the family of such quantifiers associated with any regular language L can be replaced with just the unary quantifiers. This also answers a question left open in [13].

In Section 2, we cover the relevant background material on semigroup theory, typed monoids, and multiplication quantifiers. In Section 3, we establish the main technical result showing that quantifiers of higher arity over a regular language L can be defined using just unary quantifiers over the syntactic monoid of L . Finally, in Section 4, we apply this to obtain the algebraic characterization of $\text{DLOGTIME-uniform NC}^1$.

2 Preliminaries

We assume the reader is familiar with basic concepts of formal language theory, automata theory, complexity theory, and logic. We quickly review definitions we need to fix notation and establish conventions.

We write \mathbb{Z} for the set of integers, \mathbb{N} for the set of natural numbers (including 0), and \mathbb{Z}^+ for the set of positive integers. We write $[n]$ for the set of integers $\{1, \dots, n\}$ and \mathbb{S} for the set of *square* integers. That is, $\mathbb{S} = \{x \in \mathbb{Z}^+ \mid x = y^2 \text{ for some } y \in \mathbb{Z}\}$.

For a fixed $n \in \mathbb{Z}^+$ and an integer $i \in [n]$, we define the *n-bit one-hot encoding* of i to be the binary string $b \in \{0, 1\}^n$ such that $b_j = 1$ if, and only if, $j = i$.

2.1 Semigroups, Monoids and Groups

A *semigroup* (S, \cdot) is a set S equipped with an *associative* binary operation. We call a semigroup *finite* if S is finite. Context permitting, we may refer to a semigroup (S, \cdot) simply by its underlying set S . A *monoid* (M, \cdot) is a semigroup with a distinguished element $1_M \in M$ such that for all $m \in M$, $1_M \cdot m = m \cdot 1_M = m$. We call 1_M the *identity* or *neutral* element of M . A *group* (G, \cdot) is a monoid such that for every $g \in G$, there exists an element $g^{-1} \in G$ such that $g \cdot g^{-1} = g^{-1} \cdot g = 1$. We call g^{-1} the *inverse* of g .

Note that $(\mathbb{Z}, +)$ is a group, $(\mathbb{N}, +)$ is a monoid but not a group and $(\mathbb{Z}^+, +)$ is a semigroup but not a monoid. In the first two cases, the identity element is 0. When we refer to the monoids \mathbb{Z} or \mathbb{N} we assume that the operation referred to is standard addition.

For a monoid (M, \cdot) , we say that a set $G \subseteq M$ *generates* M if M is equal to the closure of G under \cdot ; we denote this by $M = \langle G \rangle$, or, simply, $\langle G \rangle$ if the operation is clear from context, and call G a *generating set* of M . We say that M is *finitely generated* if there exists a finite generating set of M . All monoids we consider are finitely generated. Note that \mathbb{Z}^+ is generated by $\{1\}$, \mathbb{N} by $\{0, 1\}$, and \mathbb{Z} by $\{1, -1\}$.

We write U_1 for the monoid $(\{0, 1\}, \cdot)$ where the binary operation is the standard multiplication. Note that 1 is the identity element here. For any set S , we denote by S^+ the set of non-empty finite strings over S and by S^* the set of all finite strings over S . Equipped with the concatenation operation on strings, which we denote by either \circ or simply juxtaposition, S^* is a monoid and S^+ is a semigroup but not a monoid. We refer to these as the *free monoid* and *free semigroup* over S , respectively. Note that S is a set of generators for S^+ and $S \cup \{\epsilon\}$ is a set of generators for S^* .

A monoid homomorphism from a monoid (S, \cdot_S) to a monoid (T, \cdot_T) is a function $h : S \rightarrow T$ such that for all $s_1, s_2 \in S$, $h(s_1 \cdot_S s_2) = h(s_1) \cdot_T h(s_2)$ and $h(1_S) = 1_T$. A *congruence* on

a monoid (M, \cdot) is an equivalence relation \sim on M such that for all $a, b, c, d \in M$, if $a \sim b$ and $c \sim d$, then $a \cdot c \sim b \cdot d$. We denote by M/\sim the set of equivalence classes of \sim on M . We denote by $[a]_\sim$, or simply $[a]$, the equivalence class of $a \in M$ under \sim . Any congruence \sim gives rise to the *quotient monoid* of M by \sim , namely the monoid $(M/\sim, \star)$ where for $[a], [b] \in M/\sim$, $[a] \star [b] = [a \cdot b]$. The map $\eta : M \rightarrow M/\sim$ defined by $\eta(a) = [a]$ is then a homomorphism, known as the *canonical homomorphism* of M onto M/\sim .

For future reference, we formally define the syntactic congruence associated with a language L .

► **Definition 1.** Let L be a language. We define the syntactic congruence of L as the equivalence relation \sim_L on Σ^* such that for all $x, y \in \Sigma^*$, $x \sim_L y$ if and only if for all $w, v \in \Sigma^*$, $wxy \in L$ iff $wyv \in L$.

It is easily seen that this relation is a congruence on the free monoid Σ^* . The quotient monoid Σ^*/\sim_L is known as the *syntactic monoid* of L . More generally, we say that a monoid M recognizes the language L if there is a homomorphism $h : \Sigma^* \rightarrow M$ and a set $A \subseteq M$ such that $L = h^{-1}(A)$. It is easily seen that the syntactic monoid of L recognizes L . A language is regular if, and only if, its syntactic monoid is finite.

2.2 Logics and Quantifiers

We assume familiarity with the basic syntax and semantics of first-order logic. In this paper, the logic is always interpreted in finite relational structures. We generally denote structures by Fraktur letters, \mathfrak{A} , \mathfrak{B} , etc., and the corresponding universe of the structure is denoted $|\mathfrak{A}|$, $|\mathfrak{B}|$, etc. We are almost exclusively interested in *strings* over a finite alphabet. Thus, fix an alphabet Σ . A Σ -string is then a structure \mathfrak{A} whose universe A is linearly ordered by a binary relation $<$ and which interprets a set of unary relation symbols $(R_\sigma)_{\sigma \in \Sigma}$. For each element $a \in |\mathfrak{A}|$ there is a unique $\sigma \in \Sigma$ such that a is in the interpretation of R_σ .

More generally, let τ be any relational vocabulary consisting of a binary relation symbol $<$ and unary relation symbols R_1, \dots, R_k . We can associate with any τ -structure in which $<$ is a linear order a string over an alphabet of size 2^k as formalized in the following definition.

► **Definition 2.** For τ a relational vocabulary consisting of a binary relation symbol $<$ and unary relation symbols R_1, \dots, R_k , and \mathfrak{A} a τ -structure with n elements that interprets the symbol $<$ as a linear order of its universe, we define the string $w_{\mathfrak{A}}$ associated with \mathfrak{A} as the string of length n over the alphabet $\Sigma_k = \{0, 1\}^k$ of size 2^k so that if a is the i th element of $w_{\mathfrak{A}}$, then a is the k -tuple where $a_j = 1$ if, and only if, R_j holds at the i th element of \mathfrak{A} .

This way, we can associate a language with any isomorphism-closed class of structures over the vocabulary τ . We formalize this definition for future use.

► **Definition 3.** For τ a relational vocabulary consisting of a binary relation symbol $<$ and unary relation symbols R_1, \dots, R_k , and \mathcal{A} a class of τ -structures, we define the language $L_{\mathcal{A}}$ over the alphabet $\Sigma_k = \{0, 1\}^k$ to be

$$L_{\mathcal{A}} = \{w_{\mathfrak{A}} \mid \mathfrak{A} \in \mathcal{A}\}.$$

Conversely, for any language L over the alphabet Σ_k , we define the class of τ -structures \mathcal{S}_L to be

$$\mathcal{S}_L = \{\mathfrak{A} \mid w_{\mathfrak{A}} \in L\}.$$

As the elements of a string \mathfrak{A} are linearly ordered, we can identify them with an initial segment $\{1, \dots, n\}$ of the positive integers. In other words, we treat a string with universe $\{1, \dots, n\}$ and the standard order on these elements as a canonical representative of its isomorphism class. In addition to the order predicate, we may allow other *numerical predicates* to appear in formulas of our logics. These are predicates whose meaning is completely determined by the size n of the structure and the ordering of its elements. In particular, we have ternary predicates $+$ and \times for the partial addition and multiplication functions.

An insight due to Lindström allows us to define a *quantifier* from any isomorphism-closed class of structures (see [7]). Specifically, let Q be any isomorphism-closed class of structures in a relational vocabulary $\tau = \{R_1, \dots, R_l\}$, where for each i , R_i is a relation symbol of arity r_i . For any vocabulary σ and positive integer d , an *interpretation* of τ in σ of dimension d is a tuple of formulas $I = (\phi_1(\bar{x}_1), \dots, \phi_l(\bar{x}_l))$ of vocabulary σ where ϕ_i is associated with a tuple \bar{x}_i of variables of length dr_i . Suppose we are given a σ -structure \mathfrak{A} and an assignment α that takes variables to elements of \mathfrak{A} . Then let $\phi_i^{\mathfrak{A}, \alpha}$ denote the relation of arity dr_i consisting of the set of tuples $\{\bar{a} \in |\mathfrak{A}|^{dr_i} \mid \mathfrak{A} \models \phi_i[\alpha[\bar{x}_i/\bar{a}]]\}$. Then, the interpretation I defines a map that takes a σ -structure \mathfrak{A} , along with an assignment α to the τ -structure $I(\mathfrak{A}, \alpha)$ with universe $|\mathfrak{A}|^d$ where the interpretation of R_i is the set $\phi_i^{\mathfrak{A}, \alpha}$, seen as a relation of arity r_i on $|\mathfrak{A}|^d$.

Then, in a logic with quantifier Q , we can form formulas of the form

$$Q\bar{x}_1 \cdots \bar{x}_l(\phi_1, \dots, \phi_l)$$

in which occurrences in the subformula ϕ_i of variables among x_i are bound. The semantics of this quantifier are given by the rule that $Q\bar{x}_1 \cdots \bar{x}_l(\phi_1, \dots, \phi_l)$ is true in a structure \mathfrak{A} under some interpretation α of values to the free variables if the τ -structure $I(\mathfrak{A}, \alpha)$ is in Q . Note, we have defined what are usually called *vectorized quantifiers*, in that they can take interpretations of any dimension. Another way of formulating this is to have a separate quantifier Q_d for each dimension d . We switch between these notations when it causes no confusion and we call Q_d the *vectorization* of Q of dimension d .

We are particularly interested in interpretations I where both σ and τ are vocabularies of strings. These are also known in the literature as *string-to-string transducers*. (See [4] for an example of how transducers may have many representations.) We further restrict ourselves to interpretations in which the definition of the linear order in $I(\mathfrak{A}, \alpha)$ is always the lexicographic order on d -tuples of \mathfrak{A} induced by the order in \mathfrak{A} . This order is easily defined by a (quantifier-free) first-order formula, and we simply omit it from the description of I . Hence, we only need to specify the interpretation giving the unary relations in τ and an interpretation of dimension d has the simple form $(\phi_1(\bar{x}_1), \dots, \phi_l(\bar{x}_l))$, where all tuples of variables have length d . We can then assume, without loss of generality, that they are all the same tuple \bar{x} and we thus write a formula with a string quantifier Q as

$$Q\bar{x}(\phi_1, \dots, \phi_l).$$

Observe that a quantifier applied to an interpretation of dimension d will then bind d variables.

We say that an interpretation is *unary* if it has dimension 1. We now introduce some notation we use in the rest of the paper for various logics formed by combining particular choices of quantifiers and numerical predicates.

► **Definition 4.** For a set of quantifiers \mathfrak{Q} and numerical predicates \mathfrak{N} , we denote by $(\mathfrak{Q})[\mathfrak{N}]$ the logic constructed by extending quantifier-free first-order logic with the quantifiers in \mathfrak{Q} and allowing the numerical predicates in \mathfrak{N} .

We denote by FO the set of standard first-order quantifiers: $\{\exists, \forall\}$.

For a singleton set of quantifiers $\Omega = \{Q\}$, we sometimes denote $(\Omega)[\mathfrak{N}]$ as $(Q)[\mathfrak{N}]$. We use similar notation for the sets of numerical predicates. We use $\mathcal{L}((\Omega)[\mathfrak{N}])$ to denote the languages expressible by the logic $(\Omega)[\mathfrak{N}]$. We also use $(\Omega_1)[\mathfrak{N}]$ to denote the logic obtained as a restriction of $(\Omega)[\mathfrak{N}]$ to formulas in which quantifiers in Ω are only applied to interpretations of dimension 1.

All the logics we consider are *substitution closed* in the sense of [7]. This means in particular that if a quantifier Q is definable in a logic $(\Omega)[\mathfrak{N}]$, then extending the logic with the quantifier Q does not add to its expressive power. This is because we can replace occurrences of the quantifier Q by its definition, with a suitable substitution of the interpretation for the relation symbols. Hence, if Q is definable in $(\Omega)[\mathfrak{N}]$, then $\mathcal{L}((\Omega)[\mathfrak{N}]) = \mathcal{L}((\Omega \cup \{Q\})[\mathfrak{N}])$.

A remark is due on our notation for numerical predicates. All structures we consider are ordered, including those defining the quantifiers. Thus the order predicate is implicitly present in the collection of numerical predicates \mathfrak{N} and is used (implicitly) to define the interpretations to a quantifier. We sometimes write $(\Omega)[\emptyset]$ to indicate a logic in which this is the only use of the order that is allowed. By our choice of notation, the order symbol then does not appear explicitly in the syntax of the formulas.

2.3 Multiplication Quantifiers

The definition of multiplication quantifier has its origin in Barrington, Immerman, and Straubing [1, Section 5] where they were referred to as monoid quantifiers; the authors proved that the languages in $\text{DLOGTIME-uniform NC}^1$ are exactly those expressible by first-order logic with quantifiers whose truth-value is determined via multiplication in a finite monoid. The notion was extended by Lautemann et al. [13] to include quantifiers for the word problem over more general algebras with a binary operation. Multiplication quantifiers over a finite monoid M can be understood as generalized quantifiers corresponding to languages recognized by M , and here we define them as such.

Fix a monoid M , a set $B \subseteq M$ and a positive integer k . Let Σ_k denote the set $\{0, 1\}^k$ which we think of as an alphabet of size 2^k , and fix a function $\gamma : \Sigma_k \rightarrow M$. We extend γ to strings in Σ_k^* inductively in the standard way: $\gamma(wa) = \gamma(w)\gamma(a)$. Together these define a language

$$L_{\gamma}^{M,B} = \{x \in \Sigma_k^* \mid \gamma(x) \in B\}.$$

We can now define a *multiplication quantifier*. In the following, \mathcal{S}_L denotes the class of structures associated with a language L in the sense of Definition 3.

► **Definition 5.** Let τ be a vocabulary including an order symbol $<$ and k unary relations. For a monoid M , a set $B \subseteq M$, a positive integer k and a function $\gamma : \{0, 1\}^k \rightarrow M$, the multiplication quantifier $\Gamma_{\gamma}^{M,B}$ is the Lindström quantifier associated with the class of structures $\mathcal{S}_{L_{\gamma}^{M,B}}$.

We also write $\Gamma_{d,\gamma}^{M,B}$ for the vectorization of this quantifier of dimension d . If B is a singleton $\{s\}$, then we often write $\Gamma_{d,\gamma}^{M,s}$ for short.

Recall that U_1 denotes the two-element monoid $\{0, 1\}$ with standard multiplication. Then, it is easily seen that $\Gamma_{1,\gamma}^{U_1,0}$, where $\gamma : \{0, 1\} \rightarrow U_1$ such that $\gamma(0) = 1$ and $\gamma(1) = 0$, is the standard existential quantifier. The universal quantifier can be defined similarly.

► **Definition 6.** For a monoid M , we define the following collections of quantifiers:

$$\Gamma^M = \left\{ \Gamma_{d,\gamma}^{M,B} \mid B \subseteq M, \gamma : \{0, 1\}^k \rightarrow M, \text{ and } d, k \geq 1 \right\}$$

$$\begin{aligned}
219 \quad \Gamma_d^M &= \left\{ \Gamma_{d,\gamma}^{M,B} \mid B \subseteq M \text{ and } \gamma : \{0,1\}^k \rightarrow M \right\} \\
220 \quad \Gamma_\gamma^M &= \left\{ \Gamma_{d,\gamma}^{M,B} \mid B \subseteq M \text{ and } d \geq 1 \right\} \\
221 \quad \Gamma_{d,\gamma}^M &= \left\{ \Gamma_{d,\gamma}^{M,B} \mid B \subseteq M \right\}
\end{aligned}$$

222 Finally, let Γ^{fin} be the collection of all multiplication quantifiers over finite monoids.

223 From [1, Corollary 9.1], we know that DLOGTIME-uniform NC^1 is characterized by
 224 $(\text{FO})[+, \times]$ equipped with finite multiplication quantifiers:

225 ► **Theorem 7 ([1]).** $\text{DLOGTIME-uniform NC}^1 = \mathcal{L}((\Gamma^{\text{fin}})[+, \times])$.

226 ► **Remark 8.** In fact, simply adding multiplication quantifiers for some fixed finite, non-solvable
 227 monoid to $(\text{FO})[+, \times]$ suffices. The definition of “non-solvable monoid” is not needed for our
 228 proofs here but, for example, the *symmetric group of degree five*, denoted S_5 , is a non-solvable
 229 monoid. Therefore, we know that $\text{DLOGTIME-uniform NC}^1 = \mathcal{L}((\text{FO} \cup \Gamma^{S_5})[+, \times])$.

230 In the absence of the arithmetic predicates for addition and multiplication, the logic
 231 of multiplication quantifiers over finite monoids only allows us to define regular languages.
 232 Specifically, Barrington et al. [1] established that the regular languages are characterized
 233 by the logic using such quantifiers with only unary interpretations. We denote this logic
 234 $(\Gamma_1^{\text{fin}})[<]$.

235 ► **Theorem 9 ([1]).** $\text{REG} = \mathcal{L}((\Gamma_1^{\text{fin}})[<])$.

236 Later, Lautemann et al. [13, Theorem 5.1] showed that allowing interpretations of higher
 237 dimension to the quantifiers does not increase the expressive power when order is the only
 238 numerical predicate.

239 ► **Theorem 10.** $\text{REG} = \mathcal{L}((\Gamma^{\text{fin}})[<])$.

240 Our main technical result shows that this is true even in the presence of other numerical
 241 predicates and, therefore, Γ^{fin} can be replaced by Γ_1^{fin} even in Theorem 7.

242 2.4 Typed Monoids

243 In this subsection, we review the definitions and results from [3, 10] on typed monoids, their
 244 relationship to languages and corresponding characterizations of complexity classes.

245 A typed monoid is a monoid equipped with a collection of *types*, which form a Boolean
 246 algebra, and a set of *units*. We only deal with concrete Boolean algebras, given as collections
 247 of subsets of a fixed universe.

248 ► **Definition 11 (Boolean Algebra).** A Boolean algebra over a set S is a set $B \subseteq \wp(S)$ such
 249 that $\emptyset, S \in B$ and B is closed under union, intersection, and complementation. If B is finite,
 250 we call it a finite Boolean algebra.

251 We call \emptyset and S the trivial elements (or in some contexts, the trivial types) of B .

252 A homomorphism between Boolean algebras is defined as standard. That is, if B_1 and
 253 B_2 are Boolean algebras over sets S and T , respectively, then we call $h : B_1 \rightarrow B_2$ a
 254 *homomorphism* if $h(\emptyset) = \emptyset$, $h(S) = T$, and for all $s_1, s_2 \in B_1$, $h(s_1 \cap s_2) = h(s_1) \cap h(s_2)$,
 255 $h(s_1 \cup s_2) = h(s_1) \cup h(s_2)$, and $h(s^C) = (h(s))^C$. Now we are ready to define typed monoids.

► **Definition 12** (Typed Monoid). Let M be a monoid, G a Boolean algebra over M , and E a finite subset of M . We call the tuple $T = (M, G, E)$ a typed monoid over M and the elements of G types and the elements of E units. We call M the base monoid of T . If M is a group, then we may also call T a typed group.

Say $G = \{\emptyset, A, M - A, M\}$. Then, we often abbreviate T as (M, A, E) , i.e., the Boolean algebra is signified by an element, or elements, which generates it—in this case, A .

We say that a typed monoid (M, G, E) is finite if M is.

We also need a notion of morphism between typed monoids.

► **Definition 13.** A typed monoid homomorphism $h : (S, G, E) \rightarrow (T, H, F)$ of typed monoids is a triple (h_1, h_2, h_3) where $h_1 : S \rightarrow T$ is a monoid homomorphism, $h_2 : G \rightarrow H$ is a homomorphism of Boolean algebras, and $h_3 : E \rightarrow F$ is a mapping of sets such that the following conditions hold:

(i) For all $A \in G$, $h_1(A) = h_2(A) \cap h_1(S)$.

(ii) For all $e \in E$, $h_1(e) = h_3(e)$.

Note that h_3 is redundant in the definition as it is completely determined by h_1 . We retain it as part of the definition for consistency with [3, 10].

To motivate the definitions, recall that a language $L \subseteq \Sigma^*$ is recognized by a monoid M if there is a homomorphism $h : \Sigma^* \rightarrow M$ and a set $B \subseteq M$ such that $L = h^{-1}(B)$. When the monoid M is infinite, the languages recognized form a rather rich collection and we aim to restrict this in two ways. First, B cannot be an arbitrary set but must be an element of the algebra of types. Secondly, the homomorphism h must map the letters in Σ to units of the typed monoid. Formally, we have the following definition.

► **Definition 14.** A typed monoid $T = (M, G, E)$ recognizes a language $L \subseteq \Sigma^+$ if there exists a typed monoid homomorphism from (Σ^+, L, Σ) to T . We let $\mathcal{L}(T)$ denote the set of languages recognized by T .

When the base monoid of a typed monoid is finite, we recover the classical definition of a recognition. Hence, the languages recognized by finite typed monoids are necessarily regular.

► **Proposition 15.** If T is a finite typed monoid, then $\mathcal{L}(T) \subseteq \text{REG}$.

We can now state the definitions of the key relationships between typed monoids.

► **Definition 16.** Let (S, G, E) and (T, H, F) be typed monoids.

■ A typed monoid homomorphism $h = (h_1, h_2, h_3) : (S, G, E) \rightarrow (T, H, F)$ is injective (surjective, or bijective) if all of h_1 , h_2 , and h_3 are.

■ (S, G, E) is a typed submonoid (or, simply, “submonoid” when context is obvious) of (T, H, F) , denoted $(S, G, E) \leq (T, H, F)$, if there exists an injective typed monoid homomorphism $h : (S, G, E) \rightarrow (T, H, F)$.

■ (S, G, E) divides (T, H, F) , denoted $(S, G, E) \preceq (T, H, F)$, if there exists a surjective typed monoid homomorphism from a typed submonoid of (T, H, F) to (S, G, E) .

These have the expected properties.

► **Proposition 17** ([3]). Let T_1 , T_2 , and T_3 be typed monoids.

■ Typed monoid homomorphisms are closed under composition.

■ Division is transitive: if $T_1 \preceq T_2$ and $T_2 \preceq T_3$, then $T_1 \preceq T_3$.

■ If $T_1 \preceq T_2$, then $\mathcal{L}(T_1) \subseteq \mathcal{L}(T_2)$.

298 We can formulate the notion of the *syntactic typed monoid* of a language L as an extension
 299 of the syntactic monoid of L with a minimal collection of types and units necessary.

300 ► **Definition 18.** Let $T = (M, G, E)$ be a typed monoid. A congruence \sim over M is a typed
 301 congruence over T if for every $A \in G$ and $s_1, s_2 \in M$, if $s_1 \sim s_2$ and $s_1 \in A$, then $s_2 \in A$.

302 For a typed congruence \sim over T , let

$$303 \quad A/\sim = \{[x]_\sim \mid x \in A\} \text{ where } A \subseteq M$$

$$304 \quad G/\sim = \{A/\sim \mid A \in G\}$$

$$305 \quad E/\sim = \{[x]_\sim \mid x \in E\}.$$

306 Then, $T/\sim := (M/\sim, G/\sim, E/\sim)$ is the typed quotient monoid of T by \sim .

307 Let \sim_T denote the typed congruence on T such that for $s_1, s_2 \in S$, $s_1 \sim_T s_2$ iff for all
 308 $x, y \in S$ and $A \in G$, $xs_1y \in A$ iff $xs_2y \in A$. We then refer to the quotient monoid T/\sim_T as
 309 the minimal reduced monoid of T .

310 Recall that \sim_L is the syntactic congruence of L , defined in Definition 1.

311 ► **Definition 19.** For a language $L \subseteq \Sigma^+$, the syntactic typed monoid of L , denoted $\text{syn}(L)$,
 312 is the typed monoid $(\Sigma^+, L, \Sigma)/\sim_L$.

313 We also get the *canonical typed monoid homomorphism*, $\eta_L : (\Sigma^+, L, \Sigma) \rightarrow \text{syn}(L)$ induced
 314 by the syntactic homomorphism of L .

315 We now turn to the relationship between the expressive power of logics with multiplication
 316 quantifiers and typed monoids. A formal association is defined through the definition below.

317 ► **Definition 20.** For a multiplication quantifier $Q = \Gamma_\gamma^{M,B}$ where $\gamma : \{0, 1\}^k \rightarrow M$, we define
 318 the typed quantifier monoid of Q to be the syntactic typed monoid of the language $L_\gamma^{M,B}$.

319 It turns out that we can give a purely structural characterization of those typed monoids
 320 that are syntactic monoids.

321 ► **Proposition 21** ([10]). A typed monoid is the syntactic monoid of a language if, and only
 322 if, it is reduced, generated by its units, and has four or two types.

323 In case it has just two types, then it only recognizes the empty language or the language
 324 of all strings.

325 We now want to state the formal connection between the languages expressible in a logic
 326 with a collection of quantifiers and the corresponding class of typed monoids. For this we
 327 need the notion of the *ordered strong block product closure* of a set of typed monoids T ,
 328 which we denote $\text{sbpc}_<(T)$. The definition is technical and can be found in [10] and is also
 329 reproduced in the appendix for ease of reference.

330 From [10, Theorem 4.14], we then get the following relationship between logics and
 331 algebras:¹

332 ► **Theorem 22.** Let \mathfrak{Q} be a collection of quantifiers and \mathbf{Q} the corresponding set of typed
 333 quantifier monoids for \mathfrak{Q} . Then, $\mathcal{L}((\mathfrak{Q}_1)[<]) = \mathcal{L}(\text{sbpc}_<(\mathbf{Q}))$. Recall that “ \mathfrak{Q}_1 ” here is used to
 334 denote the restriction of our logic to only uses where quantifiers are applied to interpretations
 335 of dimension 1.

¹ The theorem in [10] is actually more general as it accounts for more predicates than just order; however,
 for our purposes, order alone suffices.

3 Simplifying Multiplication Quantifiers

To use Theorem 22 to obtain an algebraic characterization of NC^1 , we need to characterize this class in a logic with only unary quantifiers, i.e. where quantifiers are only applied to unary interpretations. Remark 8 gives us a characterization using first-order quantifiers and Γ^{S^5} . Our aim in this section is to show that we can eliminate the use of interpretations of dimension higher than 1 in this logic. As a first step, we show that we can restrict ourselves to quantifiers $\Gamma_\delta^{S^5, s}$ for a *fixed* function δ .

► **Lemma 23.** *For every finite monoid M , there exists a function $\delta : \{0, 1\}^{|M|} \rightarrow M$ such that for every $B \subseteq M$ and $\gamma : \{0, 1\}^k \rightarrow M$, the quantifier $\Gamma_\gamma^{M, B}$ is definable in $(\Gamma_\delta^{M, B})[\emptyset]$.*

Proof. We prove this in the case that $B = \{s\}$ is a singleton. Because M is finite, B is too. Therefore, we may reduce any application of $\Gamma_\gamma^{M, B}$ to a disjunction of formulas applying $\Gamma_\gamma^{M, s}$ for each $s \in B$.

Recall that $\Gamma_\gamma^{M, s}$ is the class of structures \mathfrak{A} in a vocabulary τ with one binary relation $<$ and k unary relations R_1, \dots, R_k such that $\gamma(w_{\mathfrak{A}}) = s$ where $w_{\mathfrak{A}}$ is the string associated with \mathfrak{A} as in Def. 2.

Let $c = |M|$, fix an enumeration $\{s_1, \dots, s_c\}$ of M , and let z be an arbitrary element of M . Let $\delta : \{0, 1\}^c \rightarrow M$ be the function where $\delta(w) = s_i$ if w is the one-hot encoding of i and $\delta(w) = z$ otherwise (that is, if the number of occurrences of the symbol 1 in the string w is not exactly one).

For each $t \in M$, define the formula $\psi_t(x)$ as follows:

$$\psi_t(x) := \bigvee_{w \in \{0, 1\}^k : \gamma(w) = t} \left(\bigwedge_{i \in [k] : w_i = 1} R_i(x) \wedge \bigwedge_{i \in [k] : w_i = 0} \neg R_i(x) \right).$$

It is easy to see that in a τ -structure \mathfrak{A} , we have $\mathfrak{A} \models \psi_t[a]$ if, and only if, the a th element of $w_{\mathfrak{A}}$ is mapped by γ to t . Thus, in particular, the formulas $\psi_{s_1}, \dots, \psi_{s_c}$ define disjoint sets that partition the universe of \mathfrak{A} . We now claim that the quantifier $\Gamma_\gamma^{M, s}$ is defined by the formula:

$$\Gamma_\delta^{M, s}(\psi_{s_1} \dots, \psi_{s_c}).$$

To see this, let I denote the unary interpretation $(\psi_{s_1} \dots, \psi_{s_c})$ so that $w_{I(\mathfrak{A})}$ is a string over $\{0, 1\}^c$. By the fact that the sets defined by the formulas $\psi_{s_1}, \dots, \psi_{s_c}$ partition $|\mathfrak{A}|$ it follows that each letter of $w_{I(\mathfrak{A})}$ is a vector in $\{0, 1\}^c$ with exactly one 1. Indeed, the a th element of $w_{I(\mathfrak{A})}$ is the one-hot encoding of i precisely if $\mathfrak{A} \models \psi_{s_i}[a]$. Since δ takes the one-hot encoding of i to s_i , we have for any $a \in |\mathfrak{A}|$

$$\begin{aligned} \delta((w_{I(\mathfrak{A})})_a) &= s_i \\ \text{iff } \mathfrak{A} \models \psi_i[a] \\ \text{iff } \gamma((w_{\mathfrak{A}})_a) &= s_i. \end{aligned}$$

Hence, $\delta(w_{I(\mathfrak{A})}) = \gamma(w_{\mathfrak{A}})$ and therefore $I(\mathfrak{A}) \in \Gamma_\delta^{S, B}$ if, and only if, $\mathfrak{A} \in \Gamma_\gamma^{S, B}$ as required. ◀

It then follows from Lemma 23 and the substitution property of quantifiers that the expressive power of $(\mathfrak{Q} \cup \Gamma^M)[\mathfrak{N}]$ is the same as that of $(\mathfrak{Q} \cup \Gamma_\delta^M)[\mathfrak{N}]$. Indeed, any application of a quantifier in Γ^M to an interpretation of dimension d can be replaced by an application of a quantifier in Γ_δ^M to an interpretation of the same dimension. We next aim to show that an application of a quantifier $\Gamma_\delta^{M, B}$ to an interpretation of dimension d can be replaced by d nested applications of quantifiers to interpretations of dimension 1.

371 ► **Lemma 24.** *For any collection \mathfrak{Q} of quantifiers, any formula of $(\mathfrak{Q} \cup \Gamma_{\delta}^M)[\mathfrak{N}]$ is equivalent*
 372 *to one of $(\mathfrak{Q} \cup \Gamma_{1,\delta}^M)[\mathfrak{N}]$.*

373 **Proof.** Again, fix an enumeration $M = \{s_1, \dots, s_c\}$ of M and recall that $\delta : \{0, 1\}^c \rightarrow M$
 374 takes the one-hot encoding of i to s_i .

375 We show, by induction on d that if we have a formula

$$376 \quad \Phi := \Gamma_{\delta}^{M,B} x_1, \dots, x_d(\phi_1(x_1, \dots, x_d), \dots, \phi_c(x_1, \dots, x_d))$$

377 where each formula ϕ_i is in $(\mathfrak{Q} \cup \Gamma_{1,\delta}^M)[\mathfrak{N}]$, then Φ is equivalent to a formula of $(\mathfrak{Q} \cup \Gamma_{1,\delta}^M)[\mathfrak{N}]$.
 378 The result then follows by induction on the structure of the formula.

379 The base case when $d = 1$ is trivially true. Assume then that we have the formula Φ for
 380 $d > 1$ and let I denote the interpretation $(\phi_1(x_1, \dots, x_d), \dots, \phi_c(x_1, \dots, x_d))$ of dimension d .

381 We claim that Φ is equivalent to the formula

$$382 \quad \Phi_1 := \Gamma_{\delta}^{M,B} x_1(\theta_1(x_1), \dots, \theta_c(x_1))$$

383 where θ_i is the formula

$$384 \quad \theta_i(x_i) := \Gamma_{\delta}^{M,s_i} x_2, \dots, x_d(\phi_1(x_1, \dots, x_d), \dots, \phi_c(x_1, \dots, x_d)).$$

Thus, Φ_1 is obtained by the application of $\Gamma_{\delta}^{M,B}$ to a unary interpretation

$$I_1 := (\theta_1(x_1), \dots, \theta_c(x_1))$$

385 where each formula θ_i is obtained as the application of a quantifier $\Gamma_{\delta}^{M,s}$ to an interpretation
 386 of dimension $d - 1$ defined by formulas of $(\mathfrak{Q} \cup \Gamma_{1,\delta}^M)[\mathfrak{N}]$. Thus, by the inductive hypothesis,
 387 each θ_i is equivalent to a formula of $(\mathfrak{Q} \cup \Gamma_{1,\delta}^M)[\mathfrak{N}]$ and we would be done.

388 It remains to show that Φ and Φ_1 are equivalent with respect to any string structure \mathfrak{A} .
 389 To see this, fix an assignment α of values in $|\mathfrak{A}|$ to the free variables of Φ . We need to show
 390 that $\mathfrak{A} \models \Phi[\alpha]$ if, and only if, $\mathfrak{A} \models \Phi_1[\alpha]$. Let n be the length of \mathfrak{A} and assume without loss
 391 of generality that the elements of $|\mathfrak{A}|$ are $\{1, \dots, n\}$ in that order.

392 Now, $w_{I(\mathfrak{A},\alpha)}$ denotes the string associated with the structure $I(\mathfrak{A},\alpha)$ and note that this is
 393 a string of length n^d whose elements are indexed by d -tuples of elements of \mathfrak{A} . By definition,
 394 $\mathfrak{A} \models \Phi[\alpha]$ precisely if $\delta(w_{I(\mathfrak{A},\alpha)}) \in B$. We can also regard I as an interpretation of dimension
 395 $d - 1$ obtained by treating the variable x_1 as a parameter. We write $w_{I(\mathfrak{A},\alpha[a/x_1])}$ for the
 396 string of length n^{d-1} that results from applying this interpretation with the assignment of a
 397 to the variable x_1 . Since the ordering of d -tuples in $w_{I(\mathfrak{A},\alpha)}$ is lexicographic, we have

$$398 \quad w_{I(\mathfrak{A},\alpha)} = w_{I(\mathfrak{A},\alpha[1/x_1])} \cdots w_{I(\mathfrak{A},\alpha[n/x_1])}$$

399 and thus

$$400 \quad \delta(w_{I(\mathfrak{A},\alpha)}) = \delta(w_{I(\mathfrak{A},\alpha[1/x_1])}) \cdots \delta(w_{I(\mathfrak{A},\alpha[n/x_1])}).$$

401 Now, by definition of θ_i , we have $\mathfrak{A} \models \theta_i[\alpha[a/x_1]]$ if, and only if, $\delta(w_{I(\mathfrak{A},\alpha[a/x_1])}) = s_i$.
 402 Thus, for each $a \in |\mathfrak{A}|$, there is exactly one i such that $\mathfrak{A} \models \theta_i[\alpha[a/x_1]]$. Thus, $w_{I_1(\mathfrak{A},\alpha)}$
 403 is the string of length n whose a th element is the one-hot encoding of i exactly when
 404 $\delta(w_{I(\mathfrak{A},\alpha[a/x_1])}) = s_i$. In other words, $\delta((w_{I_1(\mathfrak{A},\alpha)})_a) = \delta(w_{I(\mathfrak{A},\alpha[a/x_1])})$. Thus,

$$405 \quad \mathfrak{A} \models \Phi[\alpha]$$

$$406 \quad \text{iff } \delta(w_{I(\mathfrak{A},\alpha)}) \in B$$

23:12 Characterizing NC^1 with Typed Monoids

407 $\text{iff } \delta(w_{I(\mathfrak{A}, \alpha[1/x_1])}) \cdots \delta(w_{I(\mathfrak{A}, \alpha[n/x_1])}) \in B$
 408 $\text{iff } \delta((w_{I_1(\mathfrak{A}, \alpha)})_1) \cdots \delta((w_{I_1(\mathfrak{A}, \alpha)})_n) \in B$
 409 $\text{iff } \delta(w_{I_1(\mathfrak{A}, \alpha)}) \in B$
 410 $\text{iff } \mathfrak{A} \models \Phi_1[\alpha].$

411

412 Now we are ready to state the main theorem of this section.

413 ► **Theorem 25.** *For every finite monoid M , there exists a function $\delta : \{0, 1\}^{|M|} \rightarrow M$ such*
 414 *that for any collection of quantifiers \mathfrak{Q} and any set \mathfrak{N} of numerical predicates, every formula*
 415 *of $(\mathfrak{Q} \cup \Gamma^M)[\mathfrak{N}]$ is equivalent to a formula of $(\mathfrak{Q} \cup \Gamma_{1,\delta}^M)[\mathfrak{N}]$.*

416 **Proof.** Let ϕ be any formula of $(\mathfrak{Q} \cup \Gamma^M)[\mathfrak{N}]$. By Lemma 23 we can replace all occurrences of
 417 quantifiers $\Gamma_{\gamma}^{M,B}$ by their definitions using $\Gamma_{\delta}^{M,B}$ to get formula ϕ' of $(\mathfrak{Q} \cup \Gamma_{\delta}^M)[\mathfrak{N}]$ equivalent
 418 to ϕ . Finally, by Lemma 24, there is a formula of $(\mathfrak{Q} \cup \Gamma_{1,\delta}^M)[\mathfrak{N}]$ equivalent to ϕ' . ◀

419 Note that for a finite monoid M , while Γ^M and Γ_1^M are infinite sets, $\Gamma_{1,\delta}^M$ is a finite set.
 420 Therefore, this gives us a logic characterizing $\text{DLOGTIME-uniform NC}^1$ which not only uses
 421 unary quantifiers but also only has a finite number of quantifiers:

422 ► **Corollary 26.** *There exists a $\delta : \{0, 1\}^k \rightarrow S_5$ such that*

423 $\text{DLOGTIME-uniform NC}^1 = \mathcal{L}((\text{FO} \cup \Gamma_{1,\delta}^{S_5})[+, \times])$

424 This will simplify our construction of an algebra capturing $\text{DLOGTIME-uniform NC}^1$.

425 Moreover, this theorem serves as an alternative proof of Theorem 10 ([13, Theorem 5.1])
 426 which, unlike the original proof, does not rely on the use of automata:

427 ► **Corollary 27.** $\text{REG} = \mathcal{L}((\Gamma^{\text{fin}})[<]) = \mathcal{L}((\Gamma_1^{\text{fin}})[<]).$

428 and, furthermore, resolves an open question from [13]:

429 ► **Corollary 28.** $\mathcal{L}((\Gamma^{\text{fin}})[+, \times]) = \mathcal{L}((\Gamma_1^{\text{fin}})[+, \times])$

430 4 The Algebraic Characterization

431 Now that we have a first-order logic with only quantifiers containing interpretations of unary
 432 dimension capturing $\text{DLOGTIME-uniform NC}^1$, we are closer to applying Theorem 22 to
 433 construct an algebra for it. We now just need to convert the logic to a form whose only
 434 numerical predicate is $<$ without introducing quantifiers of a higher dimension.

435 To do this, we follow what was done for the construction of an algebra for TC^0 . First, we
 436 note that the quantifier Maj is true if the majority of the assignments to the bound variable
 437 satisfy the formula and the quantifier Sq which is true if the number of assignments to the
 438 bound variable satisfying the formula is a positive square number. In the below, we always
 439 assume the use of unary interpretations with these quantifiers.

440 The following lemma displays some known results about the expressiveness of these
 441 quantifiers:

442 ► **Lemma 29.**

- 443 (i) Maj is definable in $(\Gamma^{\text{fin}})[+, \times]$. (cf. [1])
 444 (ii) The quantifiers in FO are definable in $(\text{Maj})[<]$. ([12, Theorem 3.2])

- (iii) The numerical predicate $+$ is definable in $(\text{Maj})[<]$. ([12, Theorem 4.1])
 (iv) The numerical predicate \times is definable in $(\{\text{Maj}, \text{Sq}\})[<]$ and Sq is definable in $(\text{Maj})[<, +, \times]$. (cf. [15, Theorem 2.3.f] and [11, Section 2.3])

Bringing everything together, we get the following algebraic characterization of DLOGTIME-uniform NC^1 :

► **Theorem 30.**

$$\text{DLOGTIME-uniform NC}^1 = \mathcal{L}(\text{sbpc}_{<}(\{(\mathbb{Z}, \mathbb{Z}^+, \pm 1), (\mathbb{N}, \mathbb{S}, \{0, 1\}), (S_5, \wp(S_5), S_5)\})).$$

Proof. Let $\delta : \{0, 1\}^c \rightarrow S_5$ be as it was defined in Lemma 23. It is easy to see that the typed quantifier monoid for Maj is $(\mathbb{Z}, \mathbb{Z}^+, \pm 1)$, for Sq is $(\mathbb{N}, \mathbb{S}, \{0, 1\})$, and for $\Gamma_{1,\delta}^{S_5,s}$ is $(S_5, \{s\}, S_5)$.

By Corollary 26, we have that

$$\text{DLOGTIME-uniform NC}^1 = \mathcal{L}((\text{FO} \cup \Gamma_{1,\delta}^{S_5})[+, \times]).$$

Lemma 29 (ii)–(iv) allow us to define FO , $+$, and \times in $(\{\text{Maj}, \text{Sq}\})[<]$ and (i) and (iv) allow us to define Maj and Sq in $(\text{FO} \cup \Gamma_{1,\delta}^{S_5})[+, \times]$. Therefore,

$$\text{DLOGTIME-uniform NC}^1 = \mathcal{L}((\Gamma_{1,\delta}^{S_5} \cup \{\text{Maj}, \text{Sq}\})[<]).$$

Theorem 22 gives us the algebraic characterization of

$$\begin{aligned} \text{DLOGTIME-uniform NC}^1 = \mathcal{L}(\text{sbpc}_{<}(\{(\mathbb{Z}, \mathbb{Z}^+, \pm 1), (\mathbb{N}, \mathbb{S}, \{0, 1\})\} \\ \cup \{(S_5, A, S_5) \mid A \in \wp(S_5)\})) \end{aligned}$$

Because $(S_5, A, S_5) \prec (S_5, \wp(S_5), S_5)$ for all $A \in \wp(S_5)$, we lose no expressive power by consolidating all elements (S_5, A, S_5) with $(S_5, \wp(S_5), S_5)$. We neither gain expressive power because $\mathcal{L}((S_5, \wp(S_5), S_5)) \subseteq \text{REG} \subseteq \text{DLOGTIME-uniform NC}^1$. Therefore, we have our final characterization:

$$\text{DLOGTIME-uniform NC}^1 = \mathcal{L}(\text{sbpc}_{<}(\{(\mathbb{Z}, \mathbb{Z}^+, \pm 1), (\mathbb{N}, \mathbb{S}, \{0, 1\}), (S_5, \wp(S_5), S_5)\})).$$

5 Conclusion

In this work, we constructed a class of typed monoids exactly recognizing DLOGTIME-uniform NC^1 . To do so, we proved results regarding the expressive power of logics with quantifiers defined over finite monoids. Namely, we showed that changing the dimension of the quantifier does not increase or decrease the expressive power of the logic it is used in, regardless of numerical predicates available. Therefore, we were able to provide a logic characterizing DLOGTIME-uniform NC^1 which only uses unary quantifiers. We then used this logic to construct an algebraic characterization of DLOGTIME-uniform NC^1 . This result marks the second complexity class to be characterized in a such a way, with the first being DLOGTIME-uniform TC^0 [11].

In future work, we intend to construct similar algebraic characterization of other complexity classes beyond NC^1 . Doing so will be aided by the construction of new algebraic tools analogous to how the block product is used to characterize first-order quantification over interpretations of degree one but for first-order quantification over interpretations of higher degrees and for higher order quantifiers.

References

- 1 David A Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within NC¹. *Journal of Computer and System Sciences*, 41(3):274–306, 1990.
- 2 Christoph Behle, Andreas Krebs, and Mark Mercer. Linear circuits, two-variable logic and weakly blocked monoids. In *International Symposium on Mathematical Foundations of Computer Science*, pages 147–158. Springer, 2007.
- 3 Christoph Behle, Andreas Krebs, and Stephanie Reifferscheid. Typed monoids—An Eilenberg-like theorem for non regular languages. In *Algebraic Informatics: 4th International Conference, CAI 2011, Linz, Austria, June 21-24, 2011. Proceedings 4*, pages 97–114. Springer, 2011.
- 4 Mikolaj Bojanczyk. Transducers of polynomial growth. In *Proceedings of the 37th annual acm/ieee symposium on logic in computer science*, pages 1–27, 2022.
- 5 A Cano, J Cantero, and Ana Martínez-Pastor. A positive extension of Eilenberg’s variety theorem for non-regular languages. *Applicable Algebra in Engineering, Communication and Computing*, 32(5):553–573, 2021.
- 6 Stephen A Cook. Characterizations of Pushdown Machines in Terms of Time-Bounded Computers. *Journal of the ACM (JACM)*, 18(1):4–18, 1971.
- 7 H.-D. Ebbinghaus. Extended logics: The general framework. In J. Barwise and S. Feferman, editors, *Model-Theoretic Logics*, pages 25–76. Springer-Verlag, New York, 1985.
- 8 Samuel Eilenberg. *Automata, Languages, and Machines (Vol. B)*. Academic Press, 1976.
- 9 Fred C Hennie. One-tape, off-line turing machine computations. *Information and Control*, 8(6):553–578, 1965.
- 10 Andreas Krebs. *Typed semigroups, majority logic, and threshold circuits*. PhD thesis, Tübingen, Univ., Diss., 2008, 2008.
- 11 Andreas Krebs, Klaus-Jörn Lange, and Stephanie Reifferscheid. Characterizing TC0 in terms of infinite groups. *Theory of Computing Systems*, 40(4):303–325, 2007.
- 12 K-J Lange. Some results on majority quantifiers over words. In *Proceedings. 19th IEEE Annual Conference on Computational Complexity, 2004.*, pages 123–129. IEEE, 2004.
- 13 Clemens Lautemann, Pierre McKenzie, Thomas Schwentick, and Heribert Vollmer. The descriptive complexity approach to LOGCFL. *Journal of Computer and System Sciences*, 62(4):629–652, 2001.
- 14 John Rhodes and Bret Tilson. The kernel of monoid morphisms. *J. Pure Appl. Algebra*, 62(3):227–268, 1989.
- 15 Nicole Schweikardt. *On the Expressive Power of First-order Logic with Built in Predicates*. Logos-Verlag, 2002.

A Strong Block Product Closure

A.1 Weakly Closed Classes

► **Definition 31** (Direct Product of Monoids). *The direct product of two monoids (S, \cdot_S) and (T, \cdot_T) is the monoid $(S \times T, \cdot)$ where $(s_1, t_1) \cdot (s_2, t_2) = (s_1 \cdot_S s_2, t_1 \cdot_T t_2)$.*

► **Definition 32** (Direct Product of Boolean Algebras). *We define the direct product of Boolean algebras B_1 and B_2 , denoted $B_1 \times B_2$, to be the Boolean algebra generated by the set $\{A_1 \times A_2 \mid A_1 \in B_1 \text{ and } A_2 \in B_2\}$.*

► **Definition 33** (Direct Product of Typed Monoids). *The direct product $(S, G, E) \times (T, H, F)$ is the typed monoid $(S \times T, G \times H, E \times F)$.*

► **Definition 34** (Trivial Extension). *If there exists a surjective typed monoid homomorphism from (S, G, E) to (T, H, F) , then we say that (S, G, E) is a trivial extension of (T, H, F) .*

528 ► **Definition 35** (Weakly Closed Class). We call a set of typed monoids T a weakly closed
 529 class if it is closed under

530 ■ *Division*: If $(S, G, E) \in T$ and $(S, G, E) \preceq (T, H, F)$, then $(T, H, F) \in T$.

531 ■ *Direct Product*: If $(S, G, E), (T, H, F) \in T$, then $(S, G, E) \times (T, H, F) \in T$.

532 ■ *Trivial Extension*: If (S, G, E) is a trivial extension of (T, H, F) and $(T, H, F) \in T$, then
 533 $(S, G, E) \in T$.

534 We write $\text{wc}(T)$ to denote the smallest weakly closed set of typed monoids containing T .

535 A.2 The Block Product

536 The block product will be our main tool for the construction of algebraic characterizations of
 537 language classes via logic. Historically, the “*wreath product*” was first used for this purpose.
 538 Since [14], however, the block product has typically been the preferred and easier-to-work-with
 539 tool of choice. We now build up to its definition:

540 ► **Definition 36** (Left and Right Actions). A left action \star_l of a monoid (N, \cdot) on a monoid
 541 $(M, +)$ is a function from $N \times M$ to M such that for $n, n_1, n_2 \in N$ and $m, m_1, m_2 \in M$,

$$542 \quad n \star_l (m_1 + m_2) = n \star_l m_1 + n \star_l m_2$$

$$543 \quad (n_1 \cdot n_2) \star_l m = n_1 \star_l (n_2 \star_l m)$$

$$544 \quad n \star_l 1_M = 1_M$$

$$545 \quad 1_N \star_l m = m$$

546 The right action \star_r of (N, \cdot) on $(M, +)$ is defined dually. We say that left and right actions
 547 of (N, \cdot) on $(M, +)$ are compatible if for all $n_1, n_2 \in N$ and $m \in M$,

$$548 \quad (n_1 \star_l m) \star_r n_2 = n_1 \star_l (m \star_r n_2).$$

549 When clear from context, we may simply write nm for $n \star_l m$ and mn for $m \star_r n$.

550 ► **Definition 37** (Two-sided Semidirect Product). For a pair of compatible left and right actions,
 551 \star_l and \star_r of (N, \cdot) on $(M, +)$, the two-sided (or bilateral) semidirect product of $(M, +)$ and
 552 (N, \cdot) with respect to \star_l and \star_r is the monoid $(M \times N, \star)$ where for $(m_1, n_1), (m_2, n_2) \in M \times N$,

$$553 \quad (m_1, n_1) \star (m_2, n_2) = (m_1 n_2 + n_1 m_2, n_1 \cdot n_2).$$

554 ► **Definition 38** (Block Product). The block product of (M, \cdot_M) with (N, \cdot_N) , denoted $M \square N$,
 555 is the two-sided semidirect product of $(M^{N \times N}, +)$ and (N, \cdot) with respect to the left and right
 556 actions \star_l and \star_r where for $f, g \in M^{N \times N}$ and $n, n_1, n_2 \in N^1$,

557 ■ $(M^{N \times N}, +)$ is the monoid of all functions from $N \times N$ to M under componentwise
 558 product $+$:

$$559 \quad (f + g)(n_1, n_2) = f(n_1, n_2) \cdot_M g(n_1, n_2).$$

560 ■ The left action \star_l of (N, \cdot) on $(M^{N \times N}, +)$ is defined by

$$561 \quad (n \star_l f)(n_1, n_2) = f(n_1 \cdot_N n, n_2).$$

562 ■ The right action \star_r of (N, \cdot) on $(M^{N \times N}, +)$ is defined by

$$563 \quad (f \star_r n)(n_1, n_2) = f(n_1, n \cdot_N n_2).$$

564 **A.3 The Typed Block Product**

565 ► **Definition 39** (Typed Block Product). *Let (S, G, E) and (S', G', E') be typed monoids and*
 566 *$C \subseteq S'$ be a finite set. Then, the typed block product with C of (S, G, E) and (S', G', E') ,*
 567 *denoted $(S, G, E) \boxtimes_C (S', G', E')$, is the typed monoid (T, H, F) where*

- 568 (1) $T \leq S \boxtimes S'$ such that T is generated by the elements (f, s') such that
 569 (a) $s' \in E' \cup C$ and
 570 (b) $f \in E^{S' \times S'}$ such that for $b_1, b_2, b_3, b_4 \in S'$, if for all $c \in C$ and all $A' \in G'$,
 571 $b_1 c b_2 \in A'$ iff $b_3 c b_4 \in A'$, then $f(b_1, b_2) = f(b_3, b_4)$,
 572 (2) $H = \{\{(f, s) \mid f(1, 1) \in A\} \mid A \in G\}$ where 1 is the identity of S' ,
 573 (3) and $F = \{(f, s') \mid (f, s) \text{ is a generator of } T \text{ and } s' \in E'\}$.

574 ► **Definition 40.** *Because the typed monoid corresponding to the order predicate will be a very*
 575 *common, it is convenient to define an ordered typed block product, $(S, G, E) \boxtimes_C (S', G', E')$*
 576 *which will help simplify our algebraic representations whose numerical predicates only include*
 577 *order; this is defined the same as the typed block product above but with a change to condition*
 578 *(1)(b):*

- 579 (1)(b_<) $f \in E^{S' \times S'}$ such that for $b_1, b_2, b_3, b_4 \in S'$, if for all $c \in C$ and all $A' \in G'$,
 580 (i) $b_1 c b_2 \in A'$ iff $b_3 c b_4 \in A'$,
 581 (ii) $b_1 c \in A'$ iff $b_3 c \in A'$,
 582 (iii) and $c b_2 \in A'$ iff $c b_4 \in A'$,
 583 then $f(b_1, b_2) = f(b_3, b_4)$.

584 ► **Definition 41.** *For a set of typed monoids W , we let*

585
$$W_0 = \text{wc}(W)$$

586 and for each $k \geq 1$,

587
$$\blacksquare W_k = \{S_1 \boxtimes_C S_2 \mid S_1 \in W_0, S_2 \in W_{k-1}, \text{ and finite } C \subseteq S_2\}$$

588
$$\blacksquare W_k^< = \{S_1 \boxtimes_C S_2 \mid S_1 \in W_0, S_2 \in W_{k-1}^<, \text{ and finite } C \subseteq S_2\}$$

589 We define the (ordered) strong block product closure of W , denoted $\text{sbpc}(W)$ ($\text{sbpc}_<(W)$), as

590
$$\blacksquare \text{sbpc}(W) = \bigcup_{k \in \mathbb{N}} W_k$$

591
$$\blacksquare \text{sbpc}_<(W) = \bigcup_{k \in \mathbb{N}} W_k^<.$$