Linear Circuits, Two-Variable Logic and Weakly Blocked Monoids

Christoph Behle[1], Andreas Krebs[1], and Mark Mercer[2]

[1] WSI - University of Tuebingen, Sand 13, 72076 Tuebingen, Germany
[2] McGill University, Montreal, Canada

Abstract. Following recent works connecting two-variable logic to circuits and monoids, we establish, for numerical predicate sets $\mathfrak{P}$ satisfying a certain closure property, a one-to-one correspondence between $FO[<, \mathfrak{P}]$-uniform linear circuits, two-variable formulae with $\mathfrak{P}$ predicates, and weak block products of monoids. In particular, we consider the case of linear TC^0, majority quantifiers, and finitely typed monoids. This correspondence will hold for any numerical predicate set which is $FO[<]$-closed and whose predicates do not depend on the input length.

1 Introduction

The computational power of boolean circuits are of great interest as they are among the most powerful classes of computation devices for which we can prove nontrivial lower bounds [7]. To understand the power granted by nonuniformity in this setting, we often consider circuit families which can be generated under bounded resources.

In the case of small depth circuits, we are particularly interested in circuit families whose structure can be described in terms of some restricted class of logical formulae (Barrington, Immerman, and Straubing [2]). Such circuit families can often be characterized in terms of logic. For instance, the languages recognized by $FO[+, *]$-uniform AC^0 circuits are exactly those which are expressible by $FO[+, *]$ formulae. Likewise, $FO + MOD[+, *]$ formulae correspond to ACC^0 circuits, and $FO + MAJ[+, *]$ formulae correspond to TC^0. This establishes a strong connection between circuit classes and logical formulae.

The class of languages recognized by logical formula can be also characterized in terms of algebra. For instance, the class of languages recognized by $FO[<]$ formula corresponds exactly to the class of *star-free* languages, which are exactly those which are recognized via morphisms to finite aperiodic monoids, or equivalently, block products of U_1. This gives us a three-fold connection between circuits, logic and algebra.

In the case of AC^0 and ACC^0, restricting to linear size corresponds in logic to a restriction to using only two variables. This was shown in [9], and corresponds in algebra to weakly-blocked monoids. In [21] Therién and Wilke gave for first order formulae over two variables with the order predicate an algebraic characterization as the variety **DA**. By an result of Straubing and Therién [20] this

L. Kučera and A. Kučera (Eds.): MFCS 2007, LNCS 4708, pp. 147–158, 2007.
© Springer-Verlag Berlin Heidelberg 2007

variety, and thus $FO_2[<]$, can be characterized as weakly blocked monoids of U_1. Analogously, $FO+MOD_2[<]$ was shown in [19] to correspond to the variety $\mathbf{DO} \square \mathbf{G_{sol}}$, which is the closure of **DA** under weak block products of abelian groups.

The notion of *finitely typed monoids* was introduced in [8] to obtain an algebraic characterization for $\mathrm{TC}^0 = \mathcal{L}(MAJ[<, Sq])$ in terms of finitely typed monoids. It is clear that general numerical predicates, as well as linear TC^0, need the use of infinite monoids. In this paper, we show that types can be used to algebraically characterize logical formulae for many types predicate sets in a uniform way. Second, we apply these results to give matching logical and algebraic characterizations to a broad class of uniformity conditions for linear TC^0, which include the $FO[<]$-closure of the predicate sets $\{<\}$, $\{<, +\}$, and $\{<, arb\}$.

In particular we show, subject to a closure property of $\mathfrak{P}$, that the following properties of a language L are equivalent: (1) that it is recognized by a $FO[<, \mathfrak{P}]$-uniform family of TC^0 circuits of linear size and linear fan-in, (2) that it can be described by a $FO + M\widehat{A}J_2[<, \mathfrak{P}]$ formula, and (3) that it is recognized by a restricted type of morphism into a particular type of finitely typed group, constructed from weak block products of simpler groups. Recent results suggest that these characterizations can be used to prove lower bounds on linear sized circuits [5].

The remainder of the paper is structured as follows. In Sections 2 and 3 we review notions from circuits, logic, and algebra which we will require in the exposition. In Section 4 we state the main result of this paper, and in the remaining sections we prove three inclusions that yield our result.

2 Definitions

2.1 Logic

Following the conventions of Straubing's book [17], we express words $w \in \Sigma^*$ of length n as structures over the universe $[n] = \{1, \ldots, n\}$ in the following way. For each $\sigma \in \Sigma$ we have a unary relation Q_σ such that $Q_\sigma(x)$ is true when the value of w at the position x is σ. A formula ϕ over a set of free variables $\mathcal{V}$ is interpreted over $\mathcal{V}$*-structures*, which are strings $w = (w_1, \mathcal{V}_1)(w_2, \mathcal{V}_2) \ldots (w_n, \mathcal{V}_n)$ over $\Sigma \times 2^{\mathcal{V}}$, where the $\mathcal{V}_i$s are disjoint and $\bigcup_i \mathcal{V}_i = \mathcal{V}$. We define $\Sigma^* \otimes \mathcal{V}$ to be the set of all $\mathcal{V}$-structures over Σ^*, while we use $(\Sigma \times 2^{\mathcal{V}})^*$ to denote the set of arbitrary strings over $\Sigma \times 2^{\mathcal{V}}$. Let $L_{\phi,\mathcal{V}}$ be the set of all $\mathcal{V}$-structures modeling ϕ. Then for any first-order sentence ψ we can associate a language $L_\psi = L_{\psi,\emptyset}$.

A predicate is called *numerical* if its truth value does not depend on the input. (See Section 2.2) Let $\mathfrak{P}$ be a set of numerical predicates. A first-order formula over $\mathcal{V}$ is a first order formula built from the atomic formulae $\{Q_\sigma(x)\} \cup \{P \mid P \in \mathfrak{P}\}$ and free variables $\mathcal{V}$.

There are several cases in the literature where a new quantifier has been defined to obtain a correspondence between logic and algebra. For example, $Mod\, x\, \phi(x)$ [18] has been used to connect $FO+MOD$ formulae to ACC^0 circuits.

Likewise, TC^0 was shown to correspond to logical formulae using the majority quantifier $Maj\, x\, \phi(x)$, which is true iff for more than half of the positions x the formula $\phi(x)$ evaluates to true. This construction requires that we can use the logic to simulate *counting quantifiers* $\exists^{=y} x \phi(x)$ [10], which are true if and only if there are y many positions for x where $\phi(x)$ is true. But since a counting quantifier is defined with two variables, one has difficulties to apply this result in the case of two-variable logic. This leads to the following definition, which is equivalent in power to counting quantifiers and thus majority quantifiers if the number of variables is not restricted, but gives the right expressibility in the case of two variables to capture linear TC^0.

Definition 1 (Extended Majority Quantifier). *Let $\phi_1(x), \ldots, \phi_c(x)$ be formulae with one free variable. Then $Maj\ x\ \langle \phi_1(x), \ldots, \phi_c(x) \rangle$ is a formula. We define the semantics so that the formula is true if $w_{x=i} \models \phi_j$ for the majority of $(i, j) \in [n] \times [c]$. In other words,*

$$w \models Maj\ x\ \langle \phi_1, \ldots, \phi_c \rangle \Leftrightarrow 0 < \sum_{i=1}^{n} \sum_{j=1}^{c} \begin{cases} 1 & \text{if } w_{x=i} \models \phi_j \\ -1 & \text{otherwise} \end{cases}$$

In the case of $c = 1$ we have the old definition of the majority quantifier.

Definition 2. *$FO + M\widehat{A}J_2[<, \mathfrak{P}]$ is the class of two-variable logical sentences over words which are constructed from atomic formulae, the order predicate, numerical predicates from the set $\mathfrak{P}$, and the extended majority quantifier.*

2.2 Numerical Predicates

A c-ary predicate P is called *numerical* if the truth value of $P(x_1, \ldots, x_c)$ depends only on the the numeric value of $x_1, \ldots, x_c$ and the length of the input word. An assignment to a c-ary predicate can be expressed as a $\mathcal{V}$-structure over a unary alphabet with $\mathcal{V} = \{x_1, \ldots, x_c\}$. A predicate is said to be expressible in logic $\mathcal{Q}[\mathfrak{P}]$ if the corresponding $\mathcal{V}$-structures are expressible in first order with quantifiers $\mathcal{Q}$ and predicates $\mathfrak{P}$. We can naturally represent such predicates as subsets of $\mathbb{N}^{c+1}$. For a predicate P we have the subset $\mathcal{P} = \{(i_1, \ldots, i_c, n) \mid a^n_{x_1=i_1, \ldots, x_c=i_c} \models P\}$.

Definition 3 (Shifting Predicates). *A numerical c-ary predicate P is a shift of a numerical predicate P', if there exist integers $v_1, \ldots, v_{c+1}$ such that $\mathcal{P} = \{(i_1, \ldots, i_c, n) \mid (i_1 + v_1, \ldots, i_c + v_c, n + v_{c+1}) \in \mathcal{P}'\}$.*

Now we define the closure properties of predicates we need in this paper. For a set $\mathfrak{P}$ of numerical predicates, we say that a numerical predicate P is $FO[<]$-*constructible from* $\mathfrak{P}$ if P can be expressed by a $FO[<, \mathfrak{P}]$ formula.

Definition 4. *We denote by $\overline{\mathfrak{P}}$ the smallest set of predicates that contains $\mathfrak{P}$ and is closed under $FO[<]$-constructions and shifting.*

In the case of $\{<\}, \{<, +\}, \{<, +, *\}$ we have that $\overline{\{<\}}, \overline{\{<, +\}}, \overline{\{<, +, *\}}$ are the $FO[<]$ closure of these predicate sets, i.e. the shifting closure does not introduce

new predicates. Shifting may, in general, add extra predicates for predicates that depend on the length of the word.

2.3 Circuits

In this paper we consider circuits which compute functions $f : \Sigma^n \to \{0,1\}$. Our circuits will consist of *majority gates* and *input query gates*. A majority gate is true when more than half of the inputs are true and an $Inp_\sigma(i)$ query gate will output true when the ith letter of the input is σ.

A family $\{C_n\}_{n\in\mathbb{N}}$ of such circuits can be said to recognize a language in the usual way. The complexity class TC^0 consists of those languages recognized by families of threshold circuits of constant depth and polynomial size. We define LTC^0 to be the class of languages recognized by TC^0 circuit families of linear size and linear fan-in.

We consider the class of LTC^0 circuits with a uniformity condition that is expressed in terms of first order formulae over words. As in [6], we need the following definition in order to construct a uniformity language that can be expressed by $FO[<]$ formulae: For $v = (v_1, \ldots, v_c) \in [n]^c$, the unary shuffled encoding $\langle v_1, \ldots, v_c\rangle$ of v is the word w of length n over alphabet $\{\alpha, \beta\}^c$ defined by $\pi_j(w_i) = \alpha \Leftrightarrow v_j \leq i$, where $\pi_j((a_1, \ldots, a_c)) = a_j$.

Definition 5 (Uniformity language). *Let $C = \{C_n\}$ be an LTC^0 circuit family. Fix $c \in \mathbb{N}$, a labeling of the gates of each C_n with tuples $(x_1, x_2) \in [n]\times[c]$, and a unique identifier from $[|\Sigma| + 1]$ for each possible type of gate (i.e. Inp_α or majority). Additionally, we require $(1, 1)$ to be the output gate of the circuit. Then a uniformity language of C is the set of all shuffled encodings $\langle x_1, x_2, y_1, y_2, t\rangle$ such that if t denotes majority gate, then the gate (x_1, x_2) is a majority gate and has gate (y_1, y_2) as an input gate, or if t denotes an Inp_σ gate, then (x_1, x_2) is an $Inp_\sigma(y_1)$ query gate (y_2 is arbitrary).*

Using the definition of an uniformity language we can easily define uniform circuits for our setting.

Definition 6 (Uniform LTC^0). *$FO[<, \mathfrak{P}]$-uniform LTC^0 is the class of languages recognizable by a family of LTC^0 circuits with a uniformity language expressible in $FO[<, \mathfrak{P}]$.*

3 Finitely Typed Groups

In this section we recall the definition of finitely typed groups introduced in [8]. The motivation for finitely typed groups arises from the fact that the syntactic monoid of the majority function is infinite, yet the majority gates have a finite output. Typed groups allow us to model majority gates as morphisms in a meaningful way.

Let T be a group. A *type* of T is a collection of disjoint subsets $\mathfrak{T} = \{\mathcal{T}_i \mid i \in I\}$ of T for finite I. A *finitely typed group* is a group T equipped with a type $\mathfrak{T}$.

We call the elements of the boolean closure of $\mathfrak{T}$ the *extended types* of T. If the type set $\mathfrak{T}$ of T is understood we often simply write T instead of $(T, \mathfrak{T})$. Note that a finite monoid T can be regarded as a finitely typed monoid equipped with the type $\mathfrak{T} = \{\{t\} \mid t \in T\}$. The *direct product* $(S, \mathfrak{S}) \times (T, \mathfrak{T})$ of two finitely typed monoids $(S, \mathfrak{S})$ and $(T, \mathfrak{T})$ is the usual Cartesian product equipped with the type $\mathfrak{S} \times \mathfrak{T} = \{\mathcal{S} \times \mathcal{T} \mid \mathcal{S} \in \mathfrak{S},\ \mathcal{T} \in \mathfrak{T}\}$.

In the following we extend the notion of *block products* to finitely typed groups. Let $(S, \mathfrak{S})$, $(T, \mathfrak{T})$ be finitely typed monoids. Recall that the ordinary block product of S with T is defined as the bilateral semidirect product $S^{T\times T} * * T$ of $S^{T\times T}$, the set of all functions from $T \times T$ to S, with T, where the right (resp. left) action of T on $S^{T\times T}$ is given by $(f \cdot t)(t_1, t_2) = f(t_1, t\ t_2)$ $(t \cdot f)(t_1, t_2) = f(t_1 t, t_2)$, $t,\ t_1,\ t_2 \in T,\ f \in S^{T\times T}$. Note that this set may be uncountable in the case that S and T are infinite. As in [8], a discrete version of the block product is defined. We begin by defining a set of qualified functions:

Definition 7 (Type respecting functions). *A function $f(t_1, t_2) : (T, \mathfrak{T}) \times (T, \mathfrak{T}) \to S$, where S is any set, is called type respecting if it has a finite image and, for each $s \in S$, the preimage $f^{-1}(s)$ can be described by a finite boolean combination of conditions of the form $t_1 \cdot c_1 \in \mathcal{T}_1, c_2 \cdot t_2 \in \mathcal{T}_2, t_1 \cdot c_3 \cdot t_2 \in \mathcal{T}_3$ where c_1, c_2, c_3 are constants in T and $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ are types in $\mathfrak{T}$.*

The definition of the block product is the same as in the finite case but restraining the functions used to type respecting functions.

Definition 8 (Block product). *Let $(S, \mathfrak{S})$, $(T, \mathfrak{T})$ be finitely typed monoids and let V be the set of all type respecting functions with respect to T. The* finitely typed block product $(X, \mathfrak{X}) = (S, \mathfrak{S}) \boxdot (T, \mathfrak{T})$ *of $(S, \mathfrak{S})$ with $(T, \mathfrak{T})$ is defined as the bilateral semidirect product $V * * T$ of V with T (with respect to the actions given above). The type set $\mathfrak{X}$ of X consists of all types $\hat{\mathcal{S}} = \{(f, n) \in X \mid f(e_S, e_S) \in \mathcal{S}\}$, where $\mathcal{S} \in \mathfrak{S}$ and e_S is the neutral element of S. We also write $\pi_1\mathcal{X}$, with $\mathcal{X} \in \mathfrak{X}$, for the type $\mathcal{S} \in \mathfrak{S}$, such that $\hat{\mathcal{S}} = \mathcal{X}$.*

Note that for finite M and M' equipped with the type sets as above, every function $f : M \times M \to M'$ will be type respecting. Thus we have the ordinary definition of block product as a special case.

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:

$$(*) \quad (f_1, m_1) \dots (f_n, m_n) = (\sum_{i=1}^{n} m_1 \dots m_{i-1} \cdot f_i \cdot m_{i+1} \dots m_n, m_1 \dots m_n).$$

The neutral element of $(S, \mathfrak{S}) \boxdot (T, \mathfrak{T})$ is $(\mathbf{e}, e_T)$ where $\mathbf{e}$ is the function mapping all elements to the neutral element of S and e_T is the neutral element of T.

We also have the equivalence:

$$(f_1, m_1) \dots (f_n, m_n) \in \mathcal{X} \Leftrightarrow \sum_{i=1}^{n} f_i(m_1 \dots m_{i-1}, m_{i+1} \dots m_n) \in \pi_1\mathcal{X},$$

where $\pi_1\mathcal{X}$ is the base type as in Definition 8 above.

Definition 9. *We say that a finitely typed monoid $(T, \mathfrak{T})$ recognizes the language $L \subseteq \Sigma^*$ if there is a morphism $h : \Sigma^* \to T$ and a subset $\{\mathcal{T}_1, \ldots, \mathcal{T}_k\} \subseteq \mathfrak{T}$ of types of T such that $L = h^{-1}(\bigcup_{i=1}^{k} \mathcal{T}_i)$.*

Now we turn our attention to how we can characterize predicates via morphisms.

Theorem 1. *For each binary numerical predicate $P(x, y)$ there exists a finitely typed group $(T, \mathfrak{T})$ and a distinguished element $m \in T$ with the following properties:*

1. *there is a morphism $h : (\{a\} \times 2^{\{x,y\}})^* \to T$ with $h((a, \emptyset)) = m$ and an extended type $\mathcal{T}$ such that $a^n_{x=i,y=j} \models P(x, y)$ if and only if $h(a^n_{x=i,y=j}) \in \mathcal{T}$.*
2. *for all extended types $\mathcal{T}$ over $\mathfrak{T}$ and all morphisms $h : (\{a\} \times 2^{\{x,y\}})^* \to T$ with $h((a, \emptyset)) = m$ the predicate corresponding to the language $h^{-1}(\mathcal{T}) \cap \{a\}^* \otimes \{x, y\}$ is in $\overline{\{P\}}$.*

We call m the incremental element*.*

If $\mathfrak{P}$ is a set of predicates that are unary or binary the previous theorem is also true if we transform an unary predicate $P(x)$ into a binary predicate $P'(x, x)$. In following we always assume all predicates in the two-variable logic are binary predicates.

Definition 10 (Predicate group). *The tuple of a finitely typed group $(T, \mathfrak{T})$ and incremental element m is called a predicate group of P if it satisfies the conditions of Theorem 1.*

In the following we denote by $(T_P, \mathfrak{T}_P)$ and m_P the predicate group and incremental element for the predicate P. We define now the algebraic variety which corresponds to $FO + M\widehat{A}J_2[<, \mathfrak{P}]$.

Definition 11. *Let $\mathfrak{P}$ be a set of predicates. We let $\mathbf{W}_{\mathbb{Z}}(\mathfrak{P})$ be the smallest variety closed under weak block products with $\bigtimes_{k=1}^{c}((\mathbb{Z}, \mathbb{Z}^+) \boxdot (\bigtimes_{l=1}^{c'_k}(T_{kl}, \mathfrak{T}_{kl})))$ for $c, c'_1, \ldots, c'_c \in \mathbb{N}$, where $(T_{kl}, \mathfrak{T}_{kl})$ are predicate groups for predicates P_{kl}, i.e.*

$$G \in \mathbf{W}_{\mathbb{Z}}(\mathfrak{P}) \implies G \boxdot (\bigtimes_{k=1}^{c}((\mathbb{Z}, \mathbb{Z}^+) \boxdot \bigtimes_{l=1}^{c'_k}(T_{kl}, \mathfrak{T}_{kl}))) \in \mathbf{W}_{\mathbb{Z}}(\mathfrak{P}).$$

We now introduce *restricted elements* to ensure that the predicate groups that appear in the structure of groups of our variety cannot be "abused". If we do not restrict the class of allowable morphisms, then the typed monoids above can simulate counting quantifiers by using the predicate group to simulate a quantifier which should not be possible with two-variable majority logic. To assure the predicate groups are used in the designated way we start with the following definition:

Definition 12 (Restricted Element). *We define inductively the set of restricted elements:*

1. *All elements of* $(\mathbb{Z}, \mathbb{Z}^+)$ *are restricted.*
2. *For each predicate group* $(T_P, \mathfrak{T}_P)$ *only the incremental element* m_P *is restricted.*
3. *An element* $x \in A \times B$ *is restricted iff* $\pi_1(x)$ *and* $\pi_2(x)$ *are restricted.*
4. *An element* $x \in A \boxdot B$ *is restricted iff all elements in the image of* $\pi_1(x)$ *are restricted and* $\pi_2(x)$ *is restricted.*

Definition 13. *A morphism* $h : \Sigma^* \to G$ *is restricted if all elements of* $h(\Sigma)$ *are restricted.*

The following definition yields the characterization of the languages that we deal with in this paper.

Definition 14. *For a variety* $\mathbf{V}$, $\mathcal{H}_R^{-1}(\mathbf{V})$ *is the set of all languages that are recognized by a some* $(T, \mathfrak{T}) \in \mathbf{V}$ *with a restricted morphism.*

4 Results

The main theorem translates the well known connections between two-variable logic, weak blocked monoids, and linear size circuits [21,19,9] to the case of majority. By establishing a similar uniformity result as in [6], we can show how the predicates used in logic have their counterparts in algebra and circuits.

Theorem 2. *Let* $\mathfrak{P}$ *be a set of predicates closed under* $FO[<]$*-constructions and shifting. The following are equivalent:*

1. $L \in FO[<, \mathfrak{P}]$*-uniform* LTC^0,
2. $L \in \mathcal{L}(FO + M\widehat{A}J_2[<, \mathfrak{P}])$,
3. $L \in \mathcal{H}_R^{-1}(\mathbf{W}_{\mathbb{Z}}(\mathfrak{P}))$.

Proof. First we show that we can express a circuit family by a logic formula (Theorem 4). Then we show that a language in this logic can by recognized by a restricted morphism (Theorem 5). Finally, we show how to construct a circuit family for a restricted morphism (Theorem 6).

It is unknown whether the TC^0 depth hierarchy is strict. In the next theorem we show a relation between circuit depth and quantifier depth:

Theorem 3. *Let* $\mathfrak{P}$ *be a set of predicates closed under* $FO[<]$*-constructions and shifting.* $FO[<, \mathfrak{P}]$*-uniform* LTC^0 *circuits form a hierarchy in the circuits depth iff* $FO + M\widehat{A}J_2[<, \mathfrak{P}]$ *form a hierarchy in the quantifier depth.*

Proof. The proof of Theorem 4 translates a circuit of depth d into a formula of depth $d + c$ for a constant c. Similarly the proof for Theorem 5 translates a formula of quantifier depth d in a homomorphism into a group of weak block depth $d + c$. The construction of a circuit in Theorem 6 from a group of weak block depth d yields a circuit of depth $c \cdot d$.

5 Circuits to Logic

In this section we show how we can transform a circuit into a logical formula. We proceed inductively, starting with the input gates.

The following lemma helps us to express the uniformity:

Lemma 1. *Let ϕ be a formula in $FO[<,\mathfrak{P}]$ such that L_ϕ is the uniformity language of a family of LTC^0 circuits. Then the following predicates are in $\overline{\mathfrak{P}}$:*

1. *for all $x_2, y_2 \in [c]$ the binary predicate $C_{x_2,y_2}(x_1, y_1)$ which is true iff the gate labeled (x_1, x_2) is connected to (y_1, y_2) in C;*
2. *for all $\sigma \in \Sigma, x_2 \in [c]$ the binary predicate $Inp_{\sigma,x_2}(x_1, y_1)$ which is true iff the gate labeled (x_1, x_2) is an input gate that checks if there is an σ at position y_1 in the input; and*
3. *for all $x_2 \in [c]$ the unary predicate $M_{x_2}(x_1)$ which is true iff the gate labeled (x_1, x_2) is a majority gate.*

Now we show that, given a subset of positions by a formula $\phi(x)$, we can express if a formula $\psi(x)$ is true for the majority of these positions.

Lemma 2 (Relativization). *Let $\phi(x)$ and $\psi(x)$ be formulae in $FO + M\widehat{A}J_2[<, \mathfrak{P}]$ with one free variable. Then there exists a sentence in $FO + M\widehat{A}J_2[<,\mathfrak{P}]$ that is modeled by w iff*

$$|\{i \mid w_{x=i} \models \phi(x) \wedge w_{x=i} \models \psi(x)\}| > |\{i \mid w_{x=i} \models \phi(x) \wedge \neg w_{x=i} \models \psi(x)\}|.$$

Proof. The formula $Maj\ x\ \langle \phi(x) \wedge \psi(x), \neg\phi(x) \vee \psi(x)\rangle$ will do. If $\phi(x)$ is false, both formula add to 0 in the evaluation of the extended majority quantifier. If $\phi(x)$ is true, the contribution of the two formulae to the sum will be $+2$ or -2 depending on the value of $\psi(x)$.

Theorem 4. *If L is recognized by a $FO[<,\mathfrak{P}]$-uniform family of LTC^0-circuits, then L can be expressed as a formula in $FO + M\widehat{A}J_2[<,\overline{\mathfrak{P}}]$.*

Proof. The construction we use is standard (see e.g. [17,6]) but must be modified to work with two variables. Let $(C_n)_{n\in\mathbb{N}}$ be the LTC^0-circuit family recognizing L. By the assumption there is an $FO[<,\mathfrak{P}]$ formula ϕ that recognizes the uniformity language of $(C_n)_{n\in\mathbb{N}}$. As shown above we can assume that we have the predicates $C_{x_2,y_2}(x_1, y_1)$, $M_{x_2}(x_1)$, and $Inp_{\sigma,x_2}(x_1, y_1)$ in $\overline{\mathfrak{P}}$.

We now recursively construct a sentence ψ in $FO + M\widehat{A}J_2[<,\overline{\mathfrak{P}}]$ which describes the same language as $(C_n)_{n\in\mathbb{N}}$. We construct formulae $\psi_{x_2}^{(d)}$ such that $\psi_{x_2}^{(d)}(x_1)$ is true iff gate (x_1, x_2) outputs true and has depth at most d. For $d = 0$, (x_1, x_2) outputs true iff it is an input gate which outputs true, so:

$$\psi_{x_2}^{(0)}(x_1) = \bigvee_{\sigma\in\Sigma} \exists\ y_1\ (Inp_{\sigma,x_2}(x_1, y_1) \wedge Q_\sigma(y_1)).$$

Now let $G_{x_2}^{(d)}(x_1) =$

$$Maj\ y_1\ \langle\ C_{x_2,1}(x_1, y_1) \wedge \psi_1^{(d-1)}(y_1), \neg C_{x_2,1}(x_1, y_1) \vee \psi_1^{(d-1)}(y_1), \ldots,$$
$$C_{x_2,c}(x_1, y_1) \wedge \psi_c^{(d-1)}(y_1), \neg C_{x_2,c}(x_1, y_1) \vee \psi_c^{(d-1)}(y_1)\ \rangle.$$

This is the essential step. Observe that $G^{(d)}_{x_2}(x_1)$ models a majority gate at depth d. By the proof of Lemma 2 it evaluates to true iff the number of true predecessors is larger than the number of false predecessors. With the help of the formula $G^{(d)}_{x_2}(x_1)$, we define: $\psi^{(d)}_{x_2}(x_1) = M_{x_2}(x_1) \wedge G^{(d)}_{x_2}(x_1) \vee \psi^{(0)}_{x_2}(x_1)$. Finally, we define ψ to be the value of the gate labeled $(1,1)$, thus $\psi = \psi^{(d)}_1(1)$ where d is the depth of the circuit family.

6 Logic to Algebra

We will show that we can replace a logic formula over two variables by applying the *weak block product principle* a finite number of times. This extends the construction of [20].

Definition 15 (weak block product principle). *Let $\alpha : \Sigma^* \to (T, \mathfrak{T})$ be a morphism, Γ be a finite alphabet and $r : T \times \Sigma \times T \to \Gamma$ be a function such that $r_\sigma(t_1, t_2) = r(t_1, \sigma, t_2)$ is a type respecting function for all $\sigma \in \Sigma$. Then we define a length-preserving mapping $\tau_{r,\alpha} : \Sigma^* \to \Gamma^*$ by $\tau_{r,\alpha}(v_1 \cdots v_n) = w_1 \cdots w_n$, where $w_i = r(\alpha(v_1 \cdots v_{i-1}), v_i, \alpha(v_{i+1} \cdots v_n))$. If α is a restricted morphism, then we say $\tau_{r,\alpha}$ is restricted.*

As in the usual case [20], $\tau_{r,\alpha}$ is not a morphism. Without loss of generality we can assume an innermost formula of quantifier depth one to always be of the form $Maj\ x\ \langle Q_{\sigma_i}(x) \wedge P_i(x,y)\rangle_{i=1,\ldots,c}$. First predicates using only y can be moved out of the scope of the quantifier, and the formulae inside the quantifier can be assumed to have the form $Q_{\sigma_i}(x) \wedge P_i(x,y)$ since the predicate set is closed under boolean combinations. We now proceed by induction on the depth.

Lemma 3. *Let ϕ be a formula in $FO + M\widehat{A}J_2[<, \mathfrak{P}]$ with an innermost formula ψ of quantifier depth one over the alphabet Σ, and $\Gamma = \Sigma \times \{0,1\}$. We let ϕ' be the formula over Γ, which is ϕ if we replace $Q_\sigma(y)$ by $Q_{(\sigma,0)}(y) \vee Q_{(\sigma,1)}(y)$ and $\psi(y)$ by $\bigvee_{\sigma\in\Sigma} Q_{(\sigma,1)}(y)$. Then there exists a morphism $\alpha : \Sigma^* \to (T, \mathfrak{T}) = (\mathbb{Z}, \mathbb{Z}^+) \boxdot \bigtimes_{l=1}^{c} (T_{P_l}, \mathfrak{T}_{P_l})$ and type respecting function $r : T \times \Sigma \times T \to \Gamma$ such that $\tau^{-1}_{r,\alpha}(L_{\phi'}) = L_\phi$.*

Lemma 4. *Let ϕ be a formula in $FO + M\widehat{A}J_2[<, \mathfrak{P}]$ of quantifier depth $d > 1$ over the alphabet Σ. Then there exists a finite alphabet Γ and a restricted mapping $\tau_{r,\alpha} : \Sigma^* \to \Gamma^*$ and a formula ϕ' in $FO + M\widehat{A}J_2[<, \mathfrak{P}]$ of quantifier depth $d-1$ such that $L_\phi = \tau^{-1}_{r,\alpha}(L'_\phi)$.*

Lemma 5. *Let $\tau_{r,\alpha}$ be a restricted mapping with $\alpha : \Sigma^* \to (T, \mathfrak{T})$ and let $L \subseteq \Gamma^*$ be a language recognized by a morphism to $(S, \mathfrak{S})$. Then $\tau^{-1}_{r,\alpha}(L)$ is recognized by a morphism to $(S, \mathfrak{S}) \boxdot (T, \mathfrak{T})$.*

Theorem 5. *For each $L \in FO + M\widehat{A}J_2[<, \mathfrak{P}]$ there is a $(T, \mathfrak{T})$ in $\mathbf{W}_{\mathbb{Z}}(\mathfrak{P})$ and a restricted morphism h such that $L = h^{-1}(\mathcal{T})$ for an* extended type *T over $\mathfrak{T}$.*

Proof. Let ϕ be a $FO + M\widehat{A}J_2[<,\mathfrak{P}]$ formula of depth d with $L = L_\phi$. By applying Lemma 4 inductively we get a chain of mappings:

$$\Sigma^* \xrightarrow{\tau_{r_1,\alpha_1}} \Gamma_1^* \xrightarrow{\tau_{r_2,\alpha_2}} \Gamma_2^* \longrightarrow \cdots \longrightarrow \Gamma_{d-2}^* \xrightarrow{\tau_{r_{d-1},\alpha_{d-1}}} \Gamma_{d-1}^*$$

and a $FO + M\widehat{A}J_2[<,\mathfrak{P}]$ formula $\phi^{(d-1)}$ of depth one such that $L = \tau_{r_1,\alpha_1}^{-1} \circ \cdots \circ \tau_{r_{d-1},\alpha_{d-1}}^{-1}(L_{\phi^{(d-1)}})$.

The remaining formula ϕ' is of depth 1 and has no free variable $\phi' = Maj\ x\ \langle P_1(x) \wedge Q_{\sigma_1}(x), \ldots, P_c(x) \wedge Q_{\sigma_c}(x)\rangle$. hence it is easy to apply the construction of Lemma 3 for the morphism α. Since we do not have a free variable y we replace a_x by a_{xy} in the construction that simulates a variable y at the position x but is ignored by the formula ϕ'.

Now we have a morphism h' and a type $\mathcal{T}$ such that $L_{\phi^{(d-1)}} = h'^{-1}(\mathcal{T})$. By applying Lemma 5 inductively to $\tau_{r_{d-1},\alpha_{d-1}}$ up to τ_{r_1,α_1}, we will get a morphism $h : \Sigma^* \to (\cdots((T \,\square\, S_{d-1}) \,\square\, S_{d-2}) \cdots) \,\square\, S_1$, and a type $\mathcal{X}$ with $L = h^{-1}(\mathcal{X})$.

7 Algebra to Circuits

In order to model a morphism by a circuit, we will first split the morphism into mappings.

Lemma 6. *Let $h : \Sigma^* \to (S,\mathfrak{S}) \,\square\, (T,\mathfrak{T})$ and $L = h^{-1}(\mathcal{X})$ for some type $\mathcal{X}$ of $(S,\mathfrak{S}) \,\square\, (T,\mathfrak{T})$. Then there is a finite alphabet Γ and a map $\tau_{r,\alpha} : \Sigma^* \to \Gamma^*$ with $\alpha : \Sigma^* \to (T,\mathfrak{T})$ and a morphism $h' : \Gamma^* \to (S,\mathfrak{S})$ such that $\tau_{r,\alpha}^{-1}(h'^{-1}(\mathcal{S})) = L$ for some $\mathcal{S} \in \mathfrak{S}$. If h is restricted, then $\tau_{r,\alpha}$ and h' are also restricted.*

So if L is recognized by restricted morphism into a group $\mathbf{W}_{\mathbb{Z}}(\mathfrak{P})$, then there is a set of mappings $\tau_1, \ldots, \tau_d$ such that $L = \tau_1^{-1} \circ \cdots \circ \tau_d^{-1}(h^{-1}(\mathcal{T}))$, where all the morphisms map to a group of the form $\bigtimes_{k=1}^{c} \left((\mathbb{Z},\mathbb{Z}^+) \,\square\, \bigtimes_{l=1}^{c'_k} (T_{P_l},\mathfrak{T}_{P_l}) \right)$.

Lemma 7. *A $FO[<,P]$-uniform LTC^0 circuit can compute the function $\tau_{r,\alpha}$ where $\alpha : \Sigma^* \to (T,\mathfrak{T}) = \bigtimes_{k=1}^{c} \left((\mathbb{Z},\mathbb{Z}^+) \,\square\, \bigtimes_{l=1}^{c'_k} (T_{P_l},\mathfrak{T}_{P_l}) \right)$ is restricted. We require here for each letter $\gamma \in \Gamma$ the corresponding output gates to be labeled by (i,γ).*

Theorem 6. *Let $(T,\mathfrak{T}) \in \mathbf{W}_{\mathbb{Z}}(\mathfrak{P})$ recognize L then L is in $FO[<,\mathfrak{P}]$-uniform LTC^0.*

Proof. Let $h : \Sigma^* \to (T,\mathfrak{T})$ be a restricted morphism with $L = h^{-1}(\mathcal{T})$, where $\mathcal{T} \in \mathfrak{T}$. By applying Lemma 6 inductively we get a chain of mappings τ_{r_k,α_k} and a morphism h':

$$\Sigma^* \xrightarrow{\tau_{r_1,\alpha_1}} \Gamma_1^* \xrightarrow{\tau_{r_2,\alpha_2}} \Gamma_2^* \longrightarrow \cdots \longrightarrow \Gamma_{d-2}^* \xrightarrow{\tau_{r_{d-1},\alpha_{d-1}}} \Gamma_{d-1}^* \xrightarrow{h'} T'$$

where $T' = \bigtimes_{k=1}^{c} \left((\mathbb{Z}, \mathbb{Z}^+) \boxdot \bigtimes_{l=1}^{c'_k} (T_{P_l}, \mathfrak{T}_{P_l}) \right)$ and there is a $\mathcal{T}' \in \mathfrak{T}'$ such that $L = \tau^{-1}_{r_1,\alpha_1} \circ \cdots \circ \tau^{-1}_{r_{d-1},\alpha_{d-1}}(h'^{-1}(\mathcal{T}'))$.

To recognize $h'^{-1}(\mathcal{T}')$ we construct τ_{r_d,α_d} with $r(t_1, \sigma, t_2) = 1$ iff $t_1 \cdot t_2 \in \mathcal{T}$, $r(t_1, \sigma, t_2) = 0$ otherwise and $\alpha = h$. Then $\tau_{r_d,\alpha_d} = 1^n$ iff $w \in L$ and 0^n otherwise. Hence we can apply Lemma 7 to construct a circuit with only one output gate.

Now for each τ_{r_k,α_k} we can construct a circuit as in Lemma 7, by connecting these circuits together and also append the circuit for h' that we just created, we get a circuit that recognized L. To see that this circuit has a uniformity language in $FO[<, \mathfrak{P}]$, we label the gates (x_1, x_2) that belong to τ_{r_k,α_k} with $(x_1, (k, x_2))$ and the gates (x_1, x_2) that belong to h' by $(x_1, (d, x_2))$. Since we now have that the uniformity language for the individual circuit layers is in $FO[<, \mathfrak{P}]$, also the uniformity language for all layers is in $FO[<, \mathfrak{P}]$. The interconnection between these circuits is $FO[<]$-uniform since we always connect a series of output gates labeled by a tuple $(y_1, (d_k, y_2))$ where y_2 is a fixed constant to an input gate $(x_1, (d_{k+1}, x_2))$ where $x_1 = y_1$ and x_2 is a fixed constant.

8 Discussion

In this paper we extend the known connections between linear circuits, two-variable logic, and weakly blocked algebra from the case of linear AC^0 and linear ACC^0 to the case of linear TC^0. This algebraic characterization can be used to prove that the word problem over A_5 (known to be complete for NC^1 [1]) is not in uniform LTC^0 [5].

$FO_2[<]$ (resp. $FO + MOD_2[<]$) was linked to weakly blocked U_1 (resp. $\mathbb{Z}_p$) but no connection to circuits is known. On the other hand, $FO_2[arb]$ (resp. $FO + MOD_2[arb]$) corresponds to linear AC^0 (resp. linear ACC^0). We obtain a three-way correspondence for predicate sets respecting certain closure properties. Our proofs also hold for the case of FO_2 and $FO + MOD_2$: The group $(\mathbb{Z}, \mathbb{Z}^+)$, which simulates the quantifier, can be substituted by U_1, or by U_1 and $\mathbb{Z}_p$ to get results for those cases. In this way we obtain the possibility to handle predicate sets between the order predicate and arbitrary numerical predicates, e.g. $\overline{\{<, +\}}, \overline{\{<, +, *\}}$.

We want to thank Klaus-Jörn Lange and Stephanie Reifferscheid for helpful comments.

References

1. Barrington, D.A.: Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. J. Comp. System Sci. 38, 150–164 (1989)
2. Barrington, D.A., Immerman, N., Straubing, H.: On uniformity within NC^1. J. Comp. System Sci. 41, 274–306 (1990)
3. Barrington, D., Immerman, N., Lautemann, C., Schweickardt, N., Thérien, D.: The Crane Beach Conjecture. In: Proc. of the 16th IEEE Symposium On Logic in Computer Science, pp. 187–196. IEEE Computer Society Press, Los Alamitos (2001)

4. Barrington, D., Thérien, D.: Finite Monoids and the Fine Structure of NC^1. Journal of ACM 35(4), 941–952 (1988)
5. Behle, C., Krebs, A., Reifferscheid, S.: A_5 not in FO+MOD+MAJ$_2$[reg], `http://www-fs.informatik.uni-tuebingen.de/publi/a5notinltc0.pdf` (to appear)
6. Behle, C., Lange, K.-J.: $FO[<]$-Uniformity. In: IEEE Conference on Compuatational Complexity (2006)
7. Furst, M., Saxe, J.B., Sipser, M.: Parity circuits and the polynomial-time hierarchy. In: Proc. 22th IEEE Symposium on Foundations of Computer Science, pp. 260–270 (1981)
8. Krebs, A., Lange, K.-J., Reifferscheid, St.: In: Diekert, V., Durand, B. (eds.) STACS 2005. LNCS, vol. 3404, Springer, Heidelberg (2005)
9. Koucký, M., Lautemann, C., Poloczek, S., Thérien, D.: Circuit lower bounds via Ehrenfeucht-Fraissé games. In: Proc. 21st Conf. on Compuatational Complexity (CCC'06) (2006)
10. Lange, K.-J.: Some results on majority quantifiers over words. In: Proc. of the 19th IEEE Conference on Computational Complexity, pp. 123–129. IEEE Computer Society Press, Los Alamitos (2004)
11. Lautemann, C., McKenzie, P., Schwentick, T., Vollmer, H.: The descriptive complexity approach to LOGCFL. J. Comp. System Sci. 62, 629–652 (2001)
12. Lawson, M.: Finite Automata. Chapman & Hall/CRC (2004)
13. Rhodes, J., Tilson, B.: The Kernel of Monoid Morphisms. J. Pure Applied Alg. 62, 227–268 (1989)
14. Roy, A., Straubing, H.: Definability of Languages by Generalized First-Order Formulas over (N,+). In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, Springer, Heidelberg (to appear 2006)
15. Ruhl, M.: Counting and addition cannot express deterministic transitive closure. In: Proc. of 14th IEEE Symposium On Logic in Computer Science, pp. 326–334. IEEE Computer Society Press, Los Alamitos (1999)
16. Schweikardt, N.: On the Expressive Power of First-Order Logic with Built-In Predicates. In: Dissertation, Universität Mainz (2001)
17. Straubing, H.: Finite Automata, Formal Logic, and Circuit Complexity. Birkhäuser (1994)
18. Straubing, H., Thérien, D., Thomas, W.: Regular languages defined by generalize quantifiers. Information and Computation 118, 289–301 (1995)
19. Straubing, H., Thérien, D.: Regular Languages Defined by Generalized First-Order Formulas with a Bounded Number of Bound Variables. In: Ferreira, A., Reichel, H. (eds.) STACS 2001. LNCS, vol. 2010, pp. 551–562. Springer, Heidelberg (2001)
20. Straubing, H., Thérien, D.: Weakly Iterated Block Products of Finite Monoids. In: Rajsbaum, S. (ed.) LATIN 2002. LNCS, vol. 2286, pp. 91–104. Springer, Heidelberg (2002)
21. Thérien, D., Wilke, T.: Over Words, Two Variables are as Powerful as One Quantifier Alternation. In: Proc. 30th ACM Symposium on the Theory of Computing, pp. 256–263 (1998)