Linear Circuits, Two-Variable Logic and Weakly Blocked Monoids

Christoph Behle, Andreas Krebs, and Mark Mercer

WSI - University of Tuebingen, Sand 13, 72076 Tuebingen, Germany

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 formulas 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 logical formulas. For instance, the languages recognized by $FO[+,*]$-uniform AC^0 circuits are exactly those which are expressible by $FO[+,*]$ formulas. Likewise ACC^0 and $FO+MOD[+,*]$ formulas correspond to ACC^0circuits, and $FO+MAJ[+,*]$ formulas correspond to TC^0. This establishes a strong connection between circuit classes and logical formulas.

The class of languages recognized by logical formula can be also be characterized in terms of algebra. For instance, the class of languages recognized by $FO[<]$ formula corresponds exactly to the class of languages recognized by *star-free* languages, which are exactly those which are recognized via morphisms to finite aperiodic monoids, or equivalently, block products of U_1 (list some more relevant logic to algebra connections). 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 by Lautemann et al. [9], and corresponds in algebra to weakly-blocked monoids. In [21] Therién and Wilke gave for first order formulas over two variables with the order predicate an algebraic characterization as the variety **DA**. By an result of Straubing and Therién [20] this 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 formulas 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 this 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 we review circuit and logic and in Section 3 give the algebra definitions that we will require in the exposition. In Section 4 we state the main result of this paper, and in the remaining sections we prove threewe review the circuit, logic, inclusions that give us the 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 $[n]$ 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 $w = (w_1, \mathcal{V}_1)(w_2, \mathcal{V}_2)\dots(w_n, \mathcal{V}_n)$ 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 $\Sigma \times 2^{\mathcal{V}}$.

A predicate is called *numerical* if its truth value does not depend on the input. (See Definition 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 formulas $\{Q_\sigma(x)\} \cup \{P \mid P \in \mathfrak{P}\}$ and free variables $\mathcal{V}$.

Let $L_{\phi,V}$ be the set of all $\mathcal{V}$-structures modelling ϕ. Then for any first-order sentence ψ we can we can associate a language $L_\psi = L_{\psi,\emptyset}$.

There are several cases in the literature where a new quantifier has been define to obtain a correspondence between logic and algebra. For example, $Mod\, x\, \phi(x)$ [18] has been used to connect $FO{+}MOD$ formulas to ACC^0 circuits. Likewise, TC^0 was shown to correspond to logical formulas using the majority quantifier $Maj\, x\, \phi(x)$, which is true iff for more than half of the positions of 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 iff iff 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 $\phi_1(x), \dots, \phi_c(x)$ be formulas with one free variable. Then $Maj\ x\ \langle \phi_1(x), \dots, \phi_c(x)\rangle$ 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,*

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

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

Definition 2. *$FO + \widehat{MAJ}_2[<, \mathfrak{P}]$ is the class of two-variable logical sentences over words which are constructed from atomic formulas, 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 exists 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 formulas over words. As in [6], we need the following definition in order to construct a uniformity language that can be accessed by $FO[<]$ formulas: 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 $\pi_j(w_i) = \alpha \Leftrightarrow v_j \leq i$.

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. 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. The typed groups allows 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 $\mathfrak{T} = \{\mathcal{T}_i \mid i \in I\}$ 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 $\mathfrak{T} = \{\{t\} \mid t \in T\}$.

We define the *direct product* $(S, \mathfrak{S}) \times (T, \mathfrak{T})$ of finitely typed monoids $(S, \mathfrak{S})$ and $(T, \mathfrak{T})$ as the usual Cartesian product equipped with $\mathfrak{S} \times \mathfrak{T} = \{\mathcal{S} \times \mathcal{T} \mid \mathcal{S} \in \mathfrak{S},\ \mathcal{T} \in \mathfrak{T}\}$.

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 $(f \cdot n)(t_1, t_2) = f(t_1, t\ t_2)(t \cdot f)(t_1, t_2) = f(t_1 t, t_2)$, $t,\ t_1,\ t_2 \in T,\ f \in S^{T\times T}$. 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 $f : (T, \mathfrak{T}) \times (T, \mathfrak{T}) \to S$, 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 $\hat{\mathcal{S}} = \{(f, n) \in X \mid f(e_N, e_N) \in \mathcal{S}\}$, where $\mathcal{S} \in \mathfrak{S}$. 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 $(\mathbf{e}, e)$ where $\mathbf{e}$ is the function mapping all elements to the neutral element of S.

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 the 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 $\{\mathcal{T}_1, \dots, \mathcal{T}_k\} \subseteq \mathfrak{T}$ 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 $\mathcal{T} \in \mathfrak{T}$ 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 predicate that are unary and binary the previous lemma 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 restricted 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 $c, c'_1, \dots, c'_c \in \mathbb{N}$, e.g.*

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

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 an quantor 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 following 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 in terms of uniformity.

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 TC^0 is a hierarchy in the depth of the circuits. The following theorem connects the circuit depth of $FO[<, \mathfrak{P}]$-uniform LTC^0 to the quantifier depth of $FO + M\widehat{A}J_2[<, \mathfrak{P}]$.

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 formulas 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 true 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 formulas 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 formulas $\psi^{(d)}_{x_2}$ such that $\psi^{(d)}_{x_2}(x_1)$ 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:

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

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

$$\begin{aligned} Maj\ y_1\ \langle\ & C_{x_2,1}(x_1, y_1) \wedge \psi^{(d-1)}_1(y_1), \neg C_{x_2,1}(x_1, y_1) \vee \psi^{(d-1)}_1(y_1), \ldots, \\ & C_{x_2,c}(x_1, y_1) \wedge \psi^{(d-1)}_c(y_1), \neg C_{x_2,c}(x_1, y_1) \vee \psi^{(d-1)}_c(y_1) \quad \rangle. \end{aligned}$$

This is the essential step. Observe that $G^{(d)}_{x_2}(x_1)$ models a majority gate at depth d. By 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. Further the formulas 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.

In the next lemma we show that we can replace such a formula by a Q-predicate over an enhanced alphabet.

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 $\tau_{r,\alpha}^{-1}(L_{\phi'}) = L_\phi$.*

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 $L_\phi = \tau_{r,\alpha}^{-1}(L'_\phi)$.*

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 $\tau_{r,\alpha}^{-1}(L)$ 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 a $\mathcal{T} \in \mathfrak{T}$.*

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 $h : \Sigma^* \to (S, \mathfrak{S}) \boxdot (T, \mathfrak{T})$ and $L = h^{-1}(\mathcal{X})$ for some type $\mathcal{X}$ of $(S, \mathfrak{S}) \boxdot (T, \mathfrak{T})$. 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 finite chain 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 $\bigtimes_{k=1}^{c} \left((\mathbb{Z}, \mathbb{Z}^+) \boxdot \bigtimes_{l=1}^{c'_k} (T_{P_l}, \mathfrak{T}_{P_l}) \right)$.

Lemma 7. *A $FO[<, P]$-uniform LTC^0 circuit can compute the function $\tau_{r,\alpha}$ where $\alpha : \Sigma^* \to (T, \mathfrak{T}) = \bigtimes_{k=1}^{c} \left((\mathbb{Z}, \mathbb{Z}^+) \boxdot \bigtimes_{l=1}^{c'_k} (T_{P_l}, \mathfrak{T}_{P_l}) \right)$ 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':

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

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 $L = \tau_{r_1,\alpha_1}^{-1} \circ \cdots \circ \tau_{r_{d-1},\alpha_{d-1}}^{-1}(h'^{-1}(\mathcal{T}'))$.

In order to recognize $h'^{-1}(\mathcal{T}')$ we will 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 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. D. A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *J. Comp. System Sci.*, 38:150–164, 1989.
2. D.A. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *J. Comp. System Sci.*, 41:274–306, 1990.
3. D. Barrington, N. Immerman, C. Lautemann, N. Schweickardt, and D. Thérien. The Crane Beach Conjecture. In *Proc. of the 16th IEEE Symposium On Logic in Computer Science*, pages 187–196, 2001.
4. D. Barrington. and D. Thérien. Finite Monoids and the Fine Structure of NC^1. Journal of ACM, vol. 35, no.4, 1988, 941-952.
5. C. Behle, A. Krebs, and Stephanie Reifferscheid A_5 not in FO+MOD+MAJ$_2$[reg]. To appear. (http://www-fs.informatik.uni-tuebingen.de/publi/a5notinltc0.pdf)
6. C. Behle and K.-J. Lange. $FO[<]$-Uniformity. IEEE Conference on Compuatational Complexity, 2006.
7. M. Furst, J. B. Saxe, and M. Sipser. Parity circuits and the polynomial-time hierarchy. In Proc. 22th IEEE Symposium on Foundations of Computer Science, 260-270, 1981.
8. A. Krebs, K.-J. Lange und St. Reifferscheid. Characterizing TC0 in terms of Infinite Groups. Proc. of the 22nd STACS 2005,LNCS 3404, pages 496-507,2005.
9. M. Koucký, C. Lautemann, S. Poloczek, and D. Thérien Circuit lower bounds via Ehrenfeucht-Fraissé games. In *Proc. 21st Conf. on Compuatational Complexity (CCC'06).* 2006
10. K.-J. Lange. Some results on majority quantifiers over words. In *Proc. of the 19th IEEE Conference on Computational Complexity*, pages 123–129, 2004.
11. C. Lautemann, P. McKenzie, T. Schwentick, and H. Vollmer. The descriptive complexity approach to LOGCFL. *J. Comp. System Sci.*, 62:629–652, 2001.
12. M. Lawson. *Finite Automata*. Chapman & Hall/CRC, 2004.
13. J. Rhodes and B. Tilson. The Kernel of Monoid Morphisms. *J. Pure Applied Alg.*, 62:227–268, 1989.
14. A. Roy and H. Straubing. Definability of Languages by Generalized First-Order Formulas over (N,+). In *Proc. of the 23rd STACS 2006*, to appear.
15. M. Ruhl. Counting and addition cannot express deterministic transitive closure. In *Proc. of 14th IEEE Symposium On Logic in Computer Science*, pages 326–334, 1999.
16. N. Schweikardt. On the Expressive Power of First-Order Logic with Built-In Predicates. Dissertation, Universität Mainz, 2001.
17. H. Straubing. *Finite Automata, Formal Logic, and Circuit Complexity.* Birkhäuser, 1994.
18. H. Straubing, D. Thérien, and T. Wilke.
19. H. Straubing, D. Thérien. *Regular Languages Defined by Generalized First-Order Formulas with a Bounded Number of Bound Variables.* STACS 2001: 551-562
20. H. Straubing, D. Thérien. *Weakly Iterated Block Products of Finite Monoids.* LATIN 2002: 91-104
21. D. Thérien, T. Wilke. *Over Words, Two Variables are as Powerful as One Quantifier Alternation,* Proc. 30th ACM Symposium on the Theory of Computing 256-263 (1998).

A Proofs

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 $\mathcal{T} \in \mathfrak{T}$ 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*.*

Proof. We can associate to each binary predicate P the subset $\mathcal{P} \subseteq \mathbb{Z}^3$ where $(i,j,n) \in \mathcal{P}$ iff $a^n_{x=i,y=j} \models P$. We choose $T = (\mathbb{Z}^3, \{\mathcal{P}, \mathbb{Z}\setminus\mathcal{P}\}) \boxdot ((\mathbb{Z},\mathbb{Z}^+)\times(\mathbb{Z},\mathbb{Z}^+))$. We first show the existence of an appropriate morphism. The suitable morphism h we construct will satisfy $h(a,\emptyset) = (f_\emptyset,(0,0))$, $h(a,\{x\}) = (f_x,(1,0))$, $h(a,\{y\}) = (f_y,(0,1))$, $h(a,\{x,y\}) = (f_{xy},(1,0))$, so the position of the variables x and y will be encoded in the second parameter. It remains to define m and the $f_{\mathcal{S}}$ functions.

In the following we use $\mathbf{1}$ and $\mathbf{0}$ to denote the constant 1 and 0 function respectively. Define $g_x : \mathbb{Z}^2 \times \mathbb{Z}^2 \to \mathbb{Z}$ by:

$$g_x(m_1,m_2) = \begin{cases} 1 & \text{if } (m_1,m_2) = ((0,0),(0,0)),((0,1),(1,0)), \\ & \qquad \text{or } ((0,0),(1,1)) \\ 0 & \text{otherwise} \end{cases}$$

Likewise define g_y by inverting the role of the x and y position. Lastly we define:

$$\begin{aligned} f_\emptyset &= g_x \times g_y \times \mathbf{1} \\ f_x &= (g_x + \mathbf{1}) \times g_y \times \mathbf{1} \\ f_y &= g_x \times (g_y + \mathbf{1}) \times \mathbf{1} \\ f_{xy} &= (g_x + \mathbf{1}) \times (g_y + \mathbf{1}) \times \mathbf{1} \end{aligned}$$

and set $m = (f_\emptyset,(0,0))$. Consider the value of $h(w)$ on a word $w \in a^n \otimes 2^{x,y}$. Recalling equation (*), it is easy to check that $\pi_1(h(a^n))(e,e) = (i,j,n)$, where i and j are the positions of x and y respectively.

On the other hand, let h be an arbitrary restricted morphism defined by $h((a,0)) = m = (f,(0,0))$, $h((a,\{x\})) = (f_x,m_x)$, $h((a,\{y\})) = (f_y,m_y)$, $h((a,\{x,y\})) = (f_{xy},m_{xy})$. The value of $\pi_1(h(a^n))(e,e)$ falls into a finite number cases, depending on the position of x relative to y and the values of m_x, m_y, and m_{xy}. For instance, consider $\pi_1(h(a^n_{x=i,y=j}))$ for $i < j$. The sum in the equation (*) evaluates to:

$$(i-1)\cdot((0,0)f(m_x+m_y)) + (0,0)f_x m_y + (j-i-1)\cdot(m_x f m_y) + m_x f_y(0,0)$$

Observe that $m_x f m_y(e,e)$ has nonzero value if and only if $m_x = (1,0)$ and $m_y = (0,1)$, or vice versa. Likewise $(0,0)f(m_x+m_y)(e,e)$ has a nonzero value if and only if $m_x + m_y$ evaluates to $(1,0)$, $(0,0)$, $(0,1)$ or $(1,1)$. This finite set of cases can be checked by a boolean combination of queries of $\overline{\{P\}}$, and so by definition the predicate evaluated by h is in $\overline{\{P\}}$.

B Proofs for Circuits to Logic

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.*

Proof. From ϕ we construct a new formula $\phi'(z_1, \ldots, z_5)$ in the following way: take ϕ and substitute every occurrence of a $Q_{(\sigma_1,\sigma_2,\sigma_3,\sigma_4,\sigma_5)}(x)$ by $\bigwedge_{i=1}^{5} \begin{cases} x \leq z_i, \text{ if } \sigma_i = \alpha \\ x > z_i, \text{ if } \sigma_i = \beta \end{cases}$. From the definition of the uniformity language it follows easily that, for a word $w = \langle i_1, \ldots, i_5 \rangle \in \Sigma^n$:

$$a^n_{z_1=i_1,\ldots,z_5=i_5} \models \phi'(z_1, \ldots, z_5) \text{ iff } w \models \phi.$$

Now for all constants x_2, y_2, σ, we can express $C_{x_2,y_2}(x_1, x_2)$, $Inp_{\sigma,x_2}(x_1, y_1)$, and $M_{x_2}(x_1)$ as:

$$\begin{aligned} C_{x_2,y_2}(x_1, y_1) &= \exists t \quad \phi'(x_1, x_2, y_1, y_2, t), \\ Inp_{\sigma,x_2}(x_1, y_1) &= \exists y_2' \quad \phi'(x_1, x_2, y_1, y_2', Inp_\sigma), \\ M_{x_2}(x_1) &= \exists y_1' \exists y_2' \quad \phi'(x_1, x_2, y_1', y_2', Maj). \end{aligned}$$

These formulae are in the $FO[<]$-closure of $\mathfrak{P}$.

C Proofs for Logic to Algebra

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 $\tau^{-1}_{r,\alpha}(L_{\phi'}) = L_\phi$.*

Proof. Assume $\psi = Maj\, x \langle Q_{\sigma_l}(x) \wedge P_l(x, y) \rangle_{l=1,\ldots,c}$. By definition of $(T_{P_l}, \mathfrak{T}_{P_l})$ we know that for each predicate P_l we have a type morphism h_l and a type $\mathcal{P}_l$ such that $L_{P_l} = h_l^{-1}(\mathcal{P}_l)$. We let

$$\begin{aligned} a &= (h_1((a, \emptyset)), \ldots, h_c((a, \emptyset))) \in \bigtimes_{l=1}^{c} (T_{P_l}, \mathfrak{T}_{P_l}), \\ a_x &= (h_1((a, x)), \ldots, h_c((a, x))) \in \bigtimes_{l=1}^{c} (T_{P_l}, \mathfrak{T}_{P_l}), \end{aligned}$$

and a_y and a_{xy} are defined likewise. We define $\alpha(\sigma) = (g_\sigma, a)$, where

$$g_\sigma, \tilde{g}_\sigma : \left(\bigtimes_{l=1}^{c} (T_{P_l}, \mathfrak{T}_{P_l}) \right) \times \left(\bigtimes_{l=1}^{c} (T_{P_l}, \mathfrak{T}_{P_l}) \right) \to \mathbb{Z}$$

are defined as

$$\begin{aligned} g_\sigma(m_1, m_2) &= |\{l \mid \sigma = \sigma_l \wedge \pi_l(m_1)\pi_l(a_x)\pi_l(m_2) \in \mathcal{P}_l\}|, \\ \tilde{g}_\sigma(m_1, m_2) &= |\{l \mid \sigma = \sigma_l \wedge \pi_l(m_1)\pi_l(a_{xy})\pi_l(m_2) \in \mathcal{P}_l\}|. \end{aligned}$$

Also we define $r : T \times \Sigma \times T \to \Gamma$ by $r(t_1, \sigma, t_2) = (\sigma, 1)$ iff $t_1(\tilde{g}_\sigma, a_y)t_2 \in \mathbb{Z}^+$ and $r(t_1, \sigma, t_2) = (\sigma, 0)$ otherwise.

We show that this definition of $\tau_{r,\alpha}$ ensures that $\tau_{r,\alpha}^{-1}(L_{\phi'}) = L_\phi$. Let $w \in \Sigma^*$, it suffices to show that for all $j = 1, \ldots, n$ we have $w_{y=j} \models \psi$ iff $\tau_{r,\alpha}$ has the letter $(w_j, 1)$ at the position j. This depends on the value of

$$r(\alpha(w_1 \ldots w_{j-1}), w_j, \alpha(w_{j+1} \ldots w_n)).$$

By the definition of r, this value is $(w_j, 1)$ iff

$$(g_{w_1}, a) \ldots (g_{w_{j-1}}, a)(\tilde{g}_{w_j}, a_y)(g_{w_{j+1}}, a) \ldots (g_{w_n}, a) \in \hat{\mathbb{Z}}^+.$$

By a small computation we get that this is equal to:

$$\sum_{i=1}^{j-1} g_{w_i}(a^{i-1}, a^{j-i-1}a_y a^{n-j}) + \tilde{g}_{w_j}(a^{j-1}, a^{n-j}) + \sum_{i=j+1}^{n} g_{w_i}(a^{j-1}a_y a^{i-j-1}, a^{n-i}) > 0$$

The value of $g_{w_i}(a^{i-1}, a^{j-i-1}a_y a^{n-j})$ is the number of l's such that

$$\pi_l(a^{i-1}a_x a^{j-i-1}a_y a^{n-j}) \in \mathcal{P}_l$$

and $w_i = \sigma_l$. This is the case if $w_{x=i,y=j} \models P_l \wedge Q_{\sigma_l}(x)$. Hence, the value of $g_{w_i}(a^{i-1}, a^{j-i-1}a_y a^{n-j})$ is the number of l such that $w_{x=i,y=j} \models P_l \wedge Q_{\sigma_l}(x)$.

Similarly we can show this for $\tilde{g}_{w_j}(a^{j-1}, a^{n-j})$ and $g_{w_i}(a^{j-1}a_y a^{i-j-1}, a^{n-i})$, hence the sum is positive iff for the majority of the tuples (i, l) we have $w_{x=i,y=j} \models P_l \wedge Q_{\sigma_l}(x)$, but this is iff $\psi_{y=j} \models w$. So $\tau_{r,\alpha}(w)$ has $(w_j, 1)$ at position j iff $\psi_{y=j} \models w$. It is now easy to see that the claim is fullfilled by the choice of $\tau_{r,\alpha}$.

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 $L_\phi = \tau_{r,\alpha}^{-1}(L'_\phi)$.*

Proof. The proof is similar to that of the previous lemma. Let $\{\psi_1, \ldots, \psi_c\}$ be the set of innermost formulas of quantifier depth one that appear inside another quantifier, and $\Gamma = \Sigma \times \{0, 1\}^c$. The idea is to apply the previous lemma in parallel for all ψ_k. Let τ_{r_k,α_k} be the mapping of the previous lemma for the formula ψ_k, where $\alpha_k : \Sigma^* \to (\mathbb{Z}, \mathbb{Z}^+) \boxdot \bigtimes_{l=1}^{c'_k}(T_{P_{kl}}, \mathfrak{T}_{P_{kl}})$. We define a mapping $\tau_{r,\alpha}$, where

$$\alpha : \Sigma^* \to \bigtimes_{k=1}^{c} \Big((\mathbb{Z}, \mathbb{Z}^+) \boxdot \bigtimes_{l=1}^{c'_k} (T_{P_{kl}}, \mathfrak{T}_{P_{kl}}) \Big)$$

$$\alpha(\sigma) = (\alpha_1(\sigma), \ldots, \alpha_c(\sigma))$$

and the function

$$r(t_1, \sigma, t_2) = (\sigma, \pi_2(r_1(\pi_1(t_1), \sigma, \pi_1(t_2))), \ldots, \pi_2(r_c(\pi_c(t_1), \sigma, \pi_c(t_2))))$$

So the mapping $\tau_{r,\alpha}$ adds a bit vector to each letter of the input word, where the bits are 1 if the corresponding formula ψ_k is true at this position.

If we adopt all Q of ϕ to the alphabet Γ and replace all formulas ψ_k by Q-predicates as in the previous lemma, then the resulting formula ϕ' has quantifier depth one less that ϕ and $\tau_{r,\alpha}^{-1}(L_{\phi'}) = L_\phi$.

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 $\tau_{r,\alpha}^{-1}(L)$ is recognized by a morphism to $(S, \mathfrak{S}) \boxdot (T, \mathfrak{T})$.*

Proof. Let $h : \Gamma^* \to S$ be a morphism with $L = h^{-1}(S)$, and we define $f_\sigma : T \times T \to S$ by $f_\sigma(t_1, t_2) = h(r(t_1, \sigma, t_2))$. We define $h' : \Sigma^* \to S \boxdot T$ by $h'(\sigma) = (f_\sigma, \alpha(\sigma))$. We claim $\tau_{r,\alpha}^{-1}(h^{-1}(\mathcal{S})) = h'^{-1}(\hat{\mathcal{S}})$. For arbitrary $w \in \Sigma^*$, we have:

Pick $w \in \Sigma^*$.

$$\begin{aligned} w \in \tau_{r,\alpha}^{-1}(h^{-1}(\mathcal{S})) &\iff h(\tau_{r,\alpha}(w)) \in \mathcal{S} \\ &\iff \prod_{i=1}^{n} h(r(\alpha(w_1 \dots w_{i-1}), w_i, \alpha(w_{i+1} \dots w_n))) \in \hat{\mathcal{S}} \\ &\iff \prod_{i=1}^{n} f_{w_i}(\alpha(w_1 \dots w_{i-1}), \alpha(w_{i+1} \dots w_n))) \in \hat{\mathcal{S}} \\ &\iff \prod_{i=1}^{n} f_{w_i}(\alpha(w_1) \cdots \alpha(w_{i-1}), \alpha(w_{i+1}) \cdots \alpha(w_n))) \in \hat{\mathcal{S}} \\ &\iff h(w) \in \hat{\mathcal{S}}. \end{aligned}$$

D Algebra to Circuits

Lemma 6. *Let $h : \Sigma^* \to (S, \mathfrak{S}) \boxdot (T, \mathfrak{T})$ and $L = h^{-1}(\mathcal{X})$ for some type $\mathcal{X}$ of $(S, \mathfrak{S}) \boxdot (T, \mathfrak{T})$. 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.*

Proof. Let $(f_\sigma, m_\sigma) = h(\sigma)$. Since all f_σ have a finite image we can define $\Gamma = \bigcup_{\sigma \in \Sigma} im\ f_\sigma$, note that we treat the elements in the image of f_σ simply as symbols and not as group elements here. We let $r(t_1, \sigma, t_2) = f_\sigma(t_1, t_2)$ and $\alpha(\sigma) = m_\sigma$. Here the morphism α is restricted if h was restricted. Let $w \in \Sigma^*$, then $w \in L$ iff $h(w) \in \mathcal{X}$ which reduces to evaluating $\prod_{i=1}^{n} f_{w_i}(m_1 \dots m_{i-1}, m_{i+1} \dots w_n) \in \pi_1 \mathcal{X}$, but by the definition of r and α, the i-th factor of this product is the i-th letter of $\tau_{r,\alpha}$. Hence we simply define $h' : \Gamma^* \to S$, which maps a symbol of Γ to the corresponding group element of S. Then we pick $\mathcal{S} = \pi_1 \mathcal{X}$.

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

Proof. The idea is to construct a layer of circuits with gates for each element $(g_{w_1} \dots g_{w_{i-1}}, r_c, g_{w_{i+1}} \dots g_{w_n})$, each letter $\sigma \in \Sigma$, and every possible value $1 \leq i \leq n$. The input of these gates is wired to the input gates according to the predicates computed n the predicate groups. Then the output gates are just a boolean combination of those gates.

We first split the function r into $r_\sigma : (T, \mathfrak{T}) \times (T, \mathfrak{T}) \to \Gamma$, where $r_\sigma(t_1, t_2) = r(t_1, \sigma, t_2)$. The idea is to create gates for each position $j = 1, \dots, n$ and all $s \in \Sigma$ and all $\gamma \in \Gamma$ that are true iff $w_j = \sigma$ and $r_s(\alpha(w_1 \dots w_{j-1}), \alpha(w_{j+1} \dots w_n)) = t$. We assume for the moment that $c = 1$.

So fix a position j and assume $w_j = \sigma$. We create a constant size circuit that checks if the value of r_σ is the letter $\gamma \in \Gamma$. The function r_σ is type respecting, thus the statement $r_\sigma(t_1, t_2) = \gamma$ can be reformulated as a finite boolean combination

of conditions of the forms: (V1) $t_1 \cdot t_r \in \mathcal{T}$, (V2) $t_r \cdot t_2 \in \mathcal{T}$, (V3) $t_1 \cdot t_r \cdot t_2 \in \mathcal{T}$, where $t_1, t_2 \in (\mathbb{Z}, \mathbb{Z}^+) \boxdot \bigtimes_{l=1}^{c'_k}(T_{P_l}, \mathfrak{T}_{P_l})$ are the arguments of r_σ, and $t_r \in (\mathbb{Z}, \mathbb{Z}^+) \boxdot \bigtimes_{l=1}^{c'_k}(T_{P_l}, \mathfrak{T}_{P_l})$ is a constant of r_σ. Since we can easily compute a finite boolean closure, it suffices to show we can construct gates that are true if one of the conditions is true.

Assume we want to check a condition $t_1 \cdot t_r \cdot t_2 \in \mathcal{T}$. The other cases can be proven similar. By plugging in t_1, t_2 we get $h(w_1 \dots w_{j-1}) \cdot t_r \cdot h(w_{j+1} \dots w_n) \in T$, let $h(\sigma) = (g_\sigma, a)$ (we know that h is restricted hence a is the incremental element and does not depend on σ) and $(g_y, a_y) := t_r$, then we get $\prod_{i=1}^{j-1}(g_{w_i}, a) \cdot (g_y, a_y) \cdot \prod_{i=j+1}^{n}(g_{w_i}, a) \in \mathcal{T}$. By computation one gets $\sum_{i=1} g_{w_i}(a^{i-1}, a^{j-i-1}a_y a^{n-j}) + g_y(a^{j-1}, a^{n-j}) + \sum_{i=j+1}^{n} g_{w_i}(a^{j-1}a_y a^{i-j-1}, a^{n-i}) \in \pi_1\mathcal{T}$. This sum is computed in the finitely typed group $(\mathbb{Z}, \mathbb{Z}^+)$, hence we need to check if the sum is positive or not. Let m the maximum absolute value in the image of any $g_{\sigma'}$ for $\sigma' \in \Sigma$ and g_y. We can compute the sum by a weighted majority gate, where the weight is the value of the summand plus m. Weighted majority gates with integer weights in the range 0 to $2m$ can be simulated my a majority gate with multiple wires corresponding to the weight to the inputs, and additional true or false inputs for shifts.

So for a given value of i and a letter $\sigma' \in \Sigma$ we have an input gate that is true if $w_i = \sigma'$. It remains to show that we can wire the input gates correctly to our input gate. For each value v in the finite image of $g_{\sigma'}$ we will find a predicate $P(i, j)$ that is true iff $g_{\sigma'}(a^{i-1}, a^{j-i-1}a_y a^{n-j}) = v$ or $g_y(a^{j-1}, a^{n-j}) = v$ or $g_{\sigma'}(a^{j-1}a_y a^{i-j-1}, a^{n-i}) = v$. By the definition of the block product $g_{\sigma'}$ is type respecting since the value of $g_{\sigma'}(m_1, m_2)$ depends on a finite number of conditions of the forms: (V1) $m_1 \cdot a_x \in \mathcal{T}'$, (V2) $a_x \cdot m_2 \in \mathcal{T}'$, (V3) $m_1 \cdot a_x \cdot m_2 \in \mathcal{T}'$. The computation of these product is in a predicate group and since compute only products of the kind $a^{i-1}a_x a^{j-i-1}a_y a^{n-j}$ where we the factors are the incremental element a except for the position i and j by theorem 1 there is a predicate P for each type $\mathcal{P}$ that is true iff this product evaluates to an element in $\mathcal{P}$.

Hence we created a $FO[<, \bar{P}]$-uniform LTC^0 circuit in the case $c = 1$. For the case $c > 1$ we split T in the c many factors, and create a circuit for each factor. By the definition of the direct product, a boolean combination of the output gates of these circuits creates the correct value. Since we need only the i-th output gates of these circuits to compute the i-th output this boolean combination is finite and results in a finite circuit for each i.