NON-DEFINABILITY OF LANGUAGES BY GENERALIZED FIRST-ORDER FORMULAS OVER ($\mathbb{N}$,+)

ANDREAS KREBS AND A V SREEJITH

Abstract. We consider first-order logic with monoidal quantifiers over words. We show that all languages with a neutral letter, definable using the addition predicate are also definable with the order predicate as the only numerical predicate. Let $\mathcal{S}$ be a subset of monoids. Let $\mathcal{L}_\mathcal{S}$ be the logic closed under quantification over the monoids in $\mathcal{S}$, and **NL** be the class of neutral letter languages. Then we prove that

$$\mathcal{L}_\mathcal{S}[<,+] \cap \mathbf{NL} = \mathcal{L}_\mathcal{S}[<] \cap \mathbf{NL}$$

Our result can be interpreted as the Crane Beach conjecture to hold for the logic $\mathcal{L}_\mathcal{S}[<,+]$. As a consequence we get the result of Roy and Straubing that FO+MOD[<,+] collapses to FO+MOD[<]. For cyclic groups, we answer an open question of Roy and Straubing, proving that MOD[<,+] collapses to MOD[<]. Our result also shows that multiplication as a numerical predicate is necessary for Barrington's theorem to hold and also to simulate majority quantifiers.

All these results can be viewed as separation results for highly uniform circuit classes. For example we separate FO[<,+]-uniform CC^0 from FO[<,+]-uniform ACC^0.

Key words. formal languages, descriptive complexity, circuit complexity, finite model theory, semigroup theory

AMS subject classifications.

1. Introduction.

1.1. Circuit Complexity. The circuit family class AC^0 is defined as the family of languages recognized by constant depth polynomial sized family of *circuits* having unbounded fan-in AND, and OR gates. Similarly $\mathrm{ACC}^0(p)$ is the family of languages recognized by constant depth polynomial sized family of circuits containing unbounded fan-in AND, OR and MOD_p for $p > 0$. Similarly $\mathrm{CC}^0(p)$ corresponds to constant depth, polynomial size circuits with only MOD_p gates. $\mathrm{ACC}^0(\mathrm{CC}^0)$ is defined as the set of languages recognized by an $\mathrm{ACC}^0(p)$ ($\mathrm{CC}^0(p)$) family of circuits for some $p > 0$. The circuit class TC^0 corresponds to circuits with constant depth, polynomial size and having in addition to AND and OR gates MAJ (majority) gate. On the other hand NC^1 circuits are defined polynomial sized, log depth circuits containing AND and OR gates. There is an alternate characterization for NC^1. It is the family of languages recognized by constant depth, polynomial sized family of circuits which uses AND, OR and finite group gates. The reader can refer to the books [Vol99], [Juk12] to know more about these classes.

Results by Razborov [Raz89] and Smolensky [Smo87] shows that:

Theorem 1.1. *[Raz89, Smo87] If p is a prime number and q is a prime other than p then the language $\mathcal{L}_q$ is not contained in* ACC^0*(p).*

Hence we can infer the following: AC^0 is separated from $\mathrm{ACC}^0(p)$ for a $p > 0$ [FSS84]; there are languages in $\mathrm{CC}^0(p)$ which are not in AC^0; the classes $\mathrm{ACC}^0(p)$ and $\mathrm{ACC}^0(q)$ are different from each other if p and q are distinct primes. But relationships between most other classes are open. For example, we do not know whether CC^0 is different from ACC^0. In fact we do not know whether $\mathrm{CC}^0(6)$ contains AC^0 or whether $\mathrm{CC}^0(6)$ is even distinct from NP. These are among the biggest unsolved problems in circuit complexity.

Each of the above circuit classes have a model-theoretic characterization. It is known, from the results of Immerman [Imm87b, Imm87a], that the set of languages accepted by *non-uniform*-AC^0 circuits are exactly those definable by first order logic which uses an order and some *arbitrary* relations. We denote this logic by $\mathrm{FO}[<, \mathrm{Arb}]$, where Arb is the class of all relations possible on $\mathbb{N}$. On the other hand *dlogtime-uniform*-AC^0 circuits are exactly those definable (see Barrington et.al[BIS90]) by first order logic which uses order, addition and multiplication relations (denoted by $\mathrm{FO}[<, +, \times]$). First order logic with different built-in predicates can be seen as the complexity class AC^0 with different uniformity conditions. From here onwards we consider only *dlogtime-uniform* circuits and hence any circuit family we mention will be *dlogtime-uniform* unless otherwise stated. Behle and Lange [BL06] gives a notion of interpreting $\mathrm{FO}[<, +]$ as highly uniform circuit classes. Other circuit families also have model theoretic characterization. We have that the circuit family CC^0corresponds to $\mathrm{MOD}[<, +, \times]$, ACC^0 corresponds to $\mathrm{FOMOD}[<, +, \times]$, TC^0 corresponds to $\mathrm{MAJ}[<, +, \times]$, and NC^1 corresponds to $\mathrm{GROUP}[<, +, \times]$. The above characterization of the circuit classes come under Descriptive complexity (it studies how different complexity classes can be captured by different logics) of circuit classes. The books by Immerman [Imm99], Vollmer [Vol99] and Straubing [Str94] show the close connection between logics with monoid quantifiers and circuit classes. Table 1 identifies the language/complexity classes for logics with different quantifiers and relations.

		Relations		
		$[<]$	$[<, +]$	$[<, +, \times]$
Quantifiers	$\exists$	Aperiodic	Our Study	AC^0
	MOD_p	p-Solvable groups		$\mathrm{CC}^0(p)$
	MOD	Solvable Groups		CC^0
	$\exists, MOD_p$	p-Solvable Monoids		$\mathrm{ACC}^0(p)$
	$\exists, MOD$	Solvable Monoids		ACC^0
	S_5	Symmetric Group, S_5		NC^1
	$Group$	Groups		NC^1
	$\exists, Group$	Monoids		NC^1
		Algebraic characterization		**Circuit Complexity**

Table 1
Generalized quantifiers and Expressiveness

This helps us to look at the questions regarding separation of circuit classes from the descriptive complexity perspective. But no separation result has been made after the announcement of Smolensky's result. Razborov and Rudich [RR97] has analyzed the reason why these questions are hard. They show that no "*natural proof*" can prove the separation between these classes. Hence, as a first step, one can ask the question of separating the logics when the multiplication relation is not available. That is, can one separate $\mathrm{MOD}[<, +]$ from $\mathrm{FOMOD}[<, +]$? Is $\mathrm{GROUP}[<, +]$ different from $\mathrm{FOMOD}[<, +]$? Table 1 shows the algebraic characterization for each of the logic classes if only the linear order is present. Algebraic techniques can be used to show that these classes are separated from each other [Str94]. Hence the most natural question would be to understand the classes of languages accepted by the various logics when addition is also present.

In Section 6, we give a powerful technique to prove lower bound results for $\mathrm{FO}[<, +]$ extended with regular quantifiers. In fact we show that most of these classes are separated. The separation corresponds to algebraic properties of the quantifiers. The corollary **??** puts this in perspective.

1.2. The Crane Beach conjecture. As we have seen, there are just a handful of techniques available for proving lower bound results for circuit families. Most of these techniques are combinatorial in nature. Secondly, there has been very little understanding of the power the built-in predicates give to the logic classes. For example, how do we show that a certain language in *non-uniform*-$\mathrm{AC}^0$is infact not in *dlogtime-uniform*-AC^0. In order to understand the expressive power of different relations, Thérien proposed (see Barrington et.al[BIL+05]), what came to be called the **Crane Beach conjecture**. In order to state the conjecture, we need to define a *neutral letter language*.

Definition 1.2. *Let $L \subseteq \Sigma^*$ be a language over the alphabet Σ. We say that a letter $\lambda \in \Sigma$ is a* neutral letter *for the language L if*

$$u\lambda v \in L \Leftrightarrow uv \in L$$

That is we can insert or delete the letter λ from any word, $w \in \Sigma^*$ without affecting its membership in L. Let us look at an example.

Example 1. *(Parity) The following language $\mathcal{L}_2$ is a language with a neutral letter, where b is a neutral letter.*

$$\mathcal{L}_2 = \{w \in \{a, b\}^* \mid |w|_a \equiv (0 \ mod \ 2)\}$$

Here is another example:

Example 2. *(word problem over group) Look at a word problem over a group G. Then clearly the identity element of the group, 1_G is a neutral letter for the language.*

The *Crane Beach conjecture* states that

Conjecture 1. *A language with a neutral letter is definable in* FO*[*$<, \mathrm{Arb}$*] iff it is definable in* FO*[*$<$*].* The conjecture says that first order logic with arbitrary numerical predicates will collapse to first order logic with only linear ordering in the presence of a neutral letter. The idea is that, in the presence of a neutral letter, formulas cannot rely on the precise location of input letters and hence numerical predicates will be of little use. Nevertheless the conjecture was refuted by Barrington et. al [BIL+05]. In fact they show, using the fact (see Ajtai and Ben-Or [ABO84]) that *dlogtime-uniform*-AC^0 can count the number of occurrences of the letter a up to log of the input size, that the conjecture does not hold for the logic $\mathrm{FO}[<, +, \times]$, i.e. first order logic with a linear order, addition, and multiplication relations. The search was then to find out for what logics and relations does the conjecture hold. So, in the most general form, the *Crane Beach conjecture* can be stated as follows. Let **NL** denote the class of languages with neutral letters. Let $\mathcal{R}$ be a set of relations on $\mathbb{N}$. Let $\mathcal{S}$ be a subset of monoids. Then the Crane Beach conjecture says that [1]

Conjecture 2.

$$\mathcal{L}_\mathcal{S}[<, \mathcal{R}] \cap \mathbf{NL} = \mathcal{L}_\mathcal{S}[<] \cap \mathbf{NL}$$

In the same paper [BIL+05], the authors identify various logics where the *CBC* (short for Crane Beach conjecture) hold and various other logics where the *CBC* does

[1] $\mathcal{L}_\mathcal{S}$ be the logic closed under quantification, where the quantifiers are Lindström quantifiers are over some monoid in $\mathcal{S}$.

not hold. For example. The Boolean closure of the Σ_1-fragment of FO[Arb] does satisfy the conjecture. That is $\mathcal{B}(\Sigma_1)[\text{Arb}] \cap \mathbf{NL} = \mathcal{B}(\Sigma_1)[<] \cap \mathbf{NL}$. Lautemann, Tesson and Thérien [LTT06] considered modulo counting quantifiers. They show that $\mathcal{B}(\Sigma_1^{0,p})[\text{Arb}] \cap \mathbf{NL} = \mathcal{B}(\Sigma_1^{0,p})[<] \cap \mathbf{NL}$. This is equivalent to showing that

Theorem 1.3. *[LTT06] Let p be a prime number. Then*

$$\text{MOD}_p[\text{Arb}] \cap \mathbf{NL} = \text{MOD}_p[<] \cap \mathbf{NL}$$

Benedikt and Libkin [BL00b], in the context of collapse results in database theory, had shown that first order logic with only the addition and order relation satisfies the Crane Beach conjecture. A different proof of the result can be found in [BIL+05]. We show that this result can be generalized to any monoid quantifier. Let $\mathcal{S}$ be a subset of monoids. Our main result (Theorem 3.1) shows that the Crane Beach conjecture hold for the logic $\mathcal{L}_{\mathcal{S}}[<,+]$. That is:

$$\mathcal{L}_{\mathcal{S}}[<,+] \cap \mathbf{NL} = \mathcal{L}_{\mathcal{S}}[<] \cap \mathbf{NL}.$$

If S is an aperiodic monoid, then the Theorem is equivalent to the result of Benedikt and Libkin. Roy and Straubing [RS07] (used ideas of Benedikt and Libkin to) show that FOMOD[<, +] in the presence of neutral letters collapse to FOMOD[<]. In the same paper they posed the question

Conjecture 3. *(posed in [RS07])*

$$\text{MOD}[<,+] \cap \mathbf{NL} = \text{MOD}[<] \cap \mathbf{NL}$$

This is proved by a corollary of our Theorem 3.1.

Our main Theorem can also be viewed from the Circuit complexity perspective. Our results therefore can be summarized as: every FO[<, +] uniform constant depth polynomial size circuit with gates that compute a product in $\mathcal{S}$ and that recognize a language with a neutral letter can be made FO[<]-uniform.

As a consequence of our Theorem 3.1 we are able to separate these highly uniform versions of circuit classes. For example: The theorem states that MOD[<, +] definable languages with a neutral letter are also definable in MOD[<]. Since MOD[<] cannot simulate existential quantifiers [Str94] we have that FO[<, +] and MOD[<, +] are incomparable. In fact we show that no group quantifier can simulate existential quantifier if only addition is available.

Another corollary gives an alternate proof of the known result [RS07] that FO-MOD(m)[<, +] cannot count modulo a prime p, which does not divide m.

Another corollary shows that the *majority quantifier* cannot be simulated by group quantifiers if multiplication is not available, thus separating MAJ[<, +] from FOGRP[<,+]. Barrington's theorem [Bar89] says that word problems over any finite group can be defined by the logic which uses only the S_5 group quantifier (the group whose elements are the set of all permutations over 5 elements) if addition and multiplication predicates are available. Our result show multiplication is necessary for Barrington's theorem to hold. In other words S_5 cannot define word problems over S_6 if only addition is available.

The interesting thing to note is how the *neutral* letter concept has turned out to useful for proving lower bound results for "highly" uniform circuit classes. The neutral letter has also been used in the past for showing non-expressibility results. It had been used for showing super linear lower bounds for bounded-width branching programs [BS91], super linear wires in circuit classes [KPT05] and in communication complexity [CKK+07]. The neutral letter concept is also closely related to *collapse-results* in database theory [BL00a].

2. Preliminaries.

2.1. Words and Languages. An *alphabet* Σ is a finite set of symbols. The set of all finite words over Σ is denoted by Σ^*, the set of all right infinite words is denoted by Σ^ω. Let $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$, denote the set of all finite words and right infinite words. A word $w \in \Sigma^\infty$ starts from position 0. For a word $w \in \Sigma^\infty$ the notation $w(i)$ denotes the i^{th} letter in w, i.e. $w = w(0)w(1)w(2)\dots$.

Consider a language $L \subseteq \Sigma^\infty$ and a letter $\lambda \in \Sigma$. We say that λ is a *neutral letter* for L if for all $u, v \in \Sigma^\infty$ we have that $u\lambda v \in L \Leftrightarrow uv \in L$. The *neutral letter languages*, or the set of all languages with a neutral letter, is denoted by **NL**. Henceforth, if otherwise not stated, λ will be the neutral letter for a language in **NL**. For a word w in a language $L \in \textbf{NL}$, we define the *non-neutral positions* $\text{nnp}(w)$ of w to be the set of all positions which do not have the neutral letter.

2.2. Monoids. A *monoid* is a set closed under a binary associative operation and has an identity element. All monoids we consider except for Σ^* and Σ^∞ will be finite. A monoid M and $S \subseteq M$ defines a *word problem*. Its language is composed of words $w \in M^*$, such that when the elements of w are multiplied in order we get an element in S. N is a *submonoid* of M, if N is a subset of M and N is itself a monoid. We say that a monoid M *divides* a monoid N if there exists a submonoid N' of N and a surjective morphism from N' to M. A monoid M *recognizes* a language $L \subseteq \Sigma^*$ if there exists a morphism $h : \Sigma^* \to M$ and a subset $T \subseteq M$ such that $L = h^{-1}(T)$. It is known that finite monoids recognize exactly regular languages [Str94]. We denote by $\mathcal{M}$ the set of all finite monoids, $\mathcal{G} \subset \mathcal{M}$ the set of all finite groups and MOD the set of all finite cyclic group. We denote by U_1 the monoid consisting of elements $\{0, 1\}$ under multiplication. For a monoid M, the element $1 \in M$ will denote its *identity element*. We also use the *block product* of monoids, whose definition can be found in [Str94]. For a set S of monoids, $bpc(S)$ denotes the smallest set which contains S and is closed under block products.

TODO: Should define block product.

$s \cdot s$	0	1
0	0	0
1	0	1

TABLE 2
U_1 monoid

2.3. Logics. Given a formula ϕ with free variables $x_1, \dots, x_k$, we write $w, i_1, \dots, i_k \models \phi$ if w is a model for the formula ϕ when the free variables x_j is assigned to i_j for $j = 1, \dots, k$. We abuse notation and let $c \in \Sigma$ also be the unary predicate symbols of the logic we consider. That is $w, i \models c(x)$ iff $w(i) = c$. Let $\mathcal{V}$ be a set of variables, $\mathcal{R}$ be a set of numerical predicates and $\mathcal{S} \subseteq \mathcal{M}$. We define the logic $\mathcal{L}_\mathcal{S}[\mathcal{R}]$ to be built from the unary predicate symbols c, where $c \in \Sigma$, the binary predicate $\{=\}$, the predicates in $\mathcal{R}$, the variable symbols $\mathcal{V}$, the Boolean connectives $\{\neg, \vee, \wedge\}$, and the monoid quantifiers Q_M^m, where $M \in \mathcal{S}$ is a monoid and $m \in M$. We also identify the logic class $\mathcal{L}_\mathcal{S}[\mathcal{R}]$ with the set of all languages definable in it.

Our definition of monoid quantifiers is a special case of Lindström quantifiers [Lin66]. The formal definition of a monoid quantifier [BIS90] is as follows. Let $M = \{m_1, \dots, m_K, 1\}$ be a monoid with $K+1$ elements. For an $m \in M$, the quantifier Q_M^m is applied on K formulas. Let x be a free variable and $\phi_1(x), \dots, \phi_K(x)$ be K

formulas. Then $u \models Q_M^m x\langle \phi_1(x), \ldots, \phi_K(x)\rangle$ iff the word u when multiplied gives the element m, i.e. $\prod_i u(i) = m$, where the i^{th} letter of u, $0 \leq i < |u|$, is

$$u(i) = \begin{cases} m_1 & \text{if } w, i \models \phi_1 \\ m_2 & \text{if } w, i \models \neg\phi_1 \wedge \phi_2 \\ & \vdots \\ m_K & \text{if } w, i \models \neg\phi_1 \wedge \cdots \wedge \neg\phi_{K-1} \wedge \phi_K \\ 1 & \text{otherwise} \end{cases}$$

The following "shorthand" notation is used to avoid clutter. We denote by $Q_M^m x \ \phi \ \langle \alpha_1, \ldots, \alpha_K \rangle$, the formula $Q_M^m x\langle \phi \wedge \alpha_1, \ldots, \phi \wedge \alpha_K\rangle$. Informally, this relativizes the quantifier to the positions where ϕ is true, by multiplying the neutral element in all other places.

Our Notation	Common Notation
$\mathcal{L}_{U_1}$	FO
$\mathcal{L}_{\mathcal{C}}$	MOD
$\mathcal{L}_{\mathcal{G}}$	GROUP
$\mathcal{L}_{\{\mathcal{C}, U_1\}}$	FOMOD
$\mathcal{L}_{\mathcal{M}}$	FOGROUP

TABLE 3
Logics: Comparing our and common notation.

The following result gives an algebraic characterization for the logic $\mathcal{L}_{\mathcal{S}}[<]$.

LEMMA 2.1 ([Str94]). *Let $\mathcal{S} \subseteq \mathcal{M}$. Let $L \subseteq \Sigma^*$ such that M is the smallest monoid which recognizes L. Then L is definable in $\mathcal{L}_{\mathcal{S}}[<]$ iff M divides a monoid in bpc(S).*

2.4. Examples. Consider the monoid U_1. It is easy to see that the word problem defined by U_1 and the set $\{0\}$ defines the regular language $1^*0(0+1)^*$. Then $Q_{U_1}^0$ is same as the existential quantifier $\exists$, since any formula of the form $\exists x\phi$ is equivalent to $Q_{U_1}^0 x \ \langle\phi\rangle$. So the logic $\mathcal{L}_{U_1}[<]$ denotes first-order logic, FO[<]. Let C_q stand for the cyclic group with q elements. Then the quantifiers $Q_{C_q}^1$ corresponds to modulo quantifiers [STT95]. Thus $\mathcal{L}_{\text{MOD}}[<]$ corresponds to all regular languages whose syntactic monoids are solvable groups [Str94]. For a sentence $\phi \in \mathcal{L}_{\mathcal{S}}[\mathcal{R}]$ we define $L(\phi) = \{w \mid w \vDash \phi\}$.

3. Results.
Recall Monoid quantifiers from Preliminaries.

Let $\mathcal{S} \subseteq \mathcal{M}$ be any set of monoids. We show that the Crane Beach conjecture is true for the logic $\mathcal{L}_{\mathcal{S}}[<, +]$.

THEOREM 3.1 (Main Theorem). *Let $\mathcal{S} \subseteq \mathcal{M}$. Then*

$$\mathcal{L}_{\mathcal{S}}[<, +] \cap \mathbf{NL} = \mathcal{L}_{\mathcal{S}}[<] \cap \mathbf{NL}$$

The proof of this theorem is given in Section 5.

3.1. Non definability Results.
Theorem 3.1 give us the following corollaries.

COROLLARY 3.2. *All languages with a neutral letter in $\mathcal{L}_{\mathcal{M}}[<, +]$ are regular.*

Proof. By Theorem 3.1 we know that all languages with a neutral letter in $\mathcal{L}_{\mathcal{M}}[<, +]$ can be defined in $\mathcal{L}_{\mathcal{M}}[<]$ which by Lemma 2.1 is the set of all regular languages. □

Recall that a monoid M *divides* a monoid N if M is a morphic image of a submonoid of N.

COROLLARY 3.3. *Let $\mathcal{S} \subseteq \mathcal{G}$. Let G be a simple group that does not divide any monoid M in $\mathcal{S}$. Then the word problem over G is not definable in $\mathcal{L}_{\mathcal{S}}[<,+]$.*

Proof. The word problem over G has a neutral letter. The result now follows from Theorem 3.1 and Lemma 2.1. □

It is known that the majority quantifier can be simulated by the non-solvable group S_5 if both multiplication and addition are available [Vol99]. We show that multiplication is necessary to simulate majority quantifiers.

COROLLARY 3.4. $\mathrm{MAJ}[<] \nsubseteq \mathcal{L}_{\mathcal{M}}[<,+]$.

Proof. Consider the language $L \subseteq \{a,b,c\}^*$ consisting of all words with an equal number of a's and b's. L can be proven to be definable in MAJ[<]. Also note that c is a neutral element for L. By Corollary 3.2, and the fact that L is nonregular, we know that L is not definable in $\mathcal{L}_{\mathcal{M}}[<,+]$. □

Barrington's theorem [Bar89] says that the word problem of any finite group can be defined in the logic $\mathcal{L}_{S_5}[<,+,*]$. The following theorem shows that multiplication is necessary for Barrington's theorem to hold.

COROLLARY 3.5. *The word problem over the group S_6 is not definable in $\mathcal{L}_{S_5}[<,+]$. Infact there does not exist any one finite monoid M such that all regular languages can be defined in $\mathcal{L}_M[<,+]$.*

Proof. A_6 is a simple subgroup of S_6, which does not divide S_5. From Corollary 3.3 it follows that the word problem over S_6 is not definable in $\mathcal{L}_{S_5}[<,+]$.
For any finite monoid M, there exists a simple group G such that G does not divide M and hence the word problem over G is not definable in $\mathcal{L}_M[<,+]$. □

Let L_p be the set of all words $w \in \{0,1\}^*$ such that the number of occurrences of 1 in w is equal to $0 \pmod{p}$. Then we get the result in [RS07] that L_p is not definable in $\mathrm{FO}+\mathrm{MOD}_m[<,+]$, if p is a prime which does not divide m.

COROLLARY 3.6 ([RS07]). *If p is a prime which does not divide m, then L_p is not definable in* $\mathrm{FO}+\mathrm{MOD}_m[<,+]$.

Proof. Let L_p be definable in $\mathrm{FO}+\mathrm{MOD}_m[<,+]$. Since 0 is a neutral letter in L_p, Theorem 3.1 says L_p is also definable in $\mathrm{FO}+\mathrm{MOD}_m[<]$. Due to Lemma 2.1 and [Str94], this is a contradiction. □

It is known that languages accepted by CC^0 circuits are exactly those which are definable by $\mathcal{L}_{\mathrm{MOD}}[<,+,*]$ formulas [Vol99]. On the other hand, it is an open question whether the language 1^* can be accepted by the circuit complexity class CC^0 [Str94].

To progress in this direction Roy and Straubing [RS07] had posed the question of whether $1^* \notin \mathcal{L}_{\mathrm{MOD}}[<,+]$. Below we show that this is the case.

COROLLARY 3.7. $1^* \notin \mathcal{L}_{\mathrm{MOD}}[<,+]$. *In fact* $1^* \notin \mathcal{L}_{\mathcal{G}}[<,+]$.

Proof. The minimal monoid which can accept 1^* is U_1 and clearly the language is in **NL**. By Theorem 3.1 if there is a formula in $\mathcal{L}_{\mathcal{G}}[<,+]$ which can define 1^*, then $\mathcal{L}_{\mathcal{G}}[<]$ can also define 1^*. From Lemma 2.1 it follows that the monoid U_1 divides a group. But this is a contradiction [Str94]. □

Behle and Lange [BL06] give a notion of interpreting $\mathcal{L}_{\mathcal{S}}[<,+]$ as highly uniform circuit classes. As a consequence we can interpret the following results as a separation of the corresponding circuit classes.

COROLLARY 3.8. *The following separation results hold, for all $m > 1$*

- $\mathrm{FO}[<,+] \nsubseteq \mathrm{MOD}[<,+]$.
- $\mathrm{MOD}_m[<,+] \nsubseteq \mathrm{FO}[<,+]$.
- $\mathrm{FO}[<,+] \subsetneq \mathrm{FO}+\mathrm{MOD}_m[<,+] \subsetneq \mathrm{FO}+\mathrm{MOD}[<,+]$

- $\mathrm{FO} + \mathrm{MOD}[<, +] \subsetneq \mathrm{FO} + \mathrm{GROUP}[<, +]$
- $\mathrm{MAJ}[<, +] \not\subseteq \mathrm{FO} + \mathrm{GROUP}[<, +]$

3.2. Decidability of Regular languages in $L_{\mathcal{S}}[<, +]$. We now look at regular languages definable by the logic $L_{\mathcal{S}}[<, +]$, for an $\mathcal{S} \subseteq \mathcal{M}$. We first show that this logic is closed under quotienting and under inverse length preserving morphims. We dont give the proofs of these claims, since they follow the standard technique. One can refer to [RS07] for the proof.

Lemma 3.9. *Let $S \subseteq \mathcal{M}$ and Σ be a finite alphabet. Let $L \subseteq \Sigma^*$ be definable in $L_{\mathcal{S}}[<, +]$ and $u, v \in \Sigma^*$. Then $u^{-1}Lv^{-1}$ is also definable in $L_{\mathcal{S}}[<, +]$.*

Lemma 3.10. *Let $S \subseteq \mathcal{M}$. Let Σ, Γ be finite alphabets and let $h : \Gamma^* \to \Sigma^*$ be a homomorphism such that $h(\Gamma) \subseteq \Sigma^r$ for some fixed $r > 0$. If $L \subseteq \Sigma^*$ is definable in $L_{\mathcal{S}}[<, +]$, then $h^{-1}(L) \subseteq \Gamma^*$ is also definable in $L_{\mathcal{S}}[<, +]$.*

We now give an algebraic characterization for regular languages definable by $\mathcal{L}_{\mathcal{S}}[<, +]$. Recall that $\mathcal{L}_{\mathcal{S}}[\mathrm{REG}]$ is defined as $\mathcal{L}_{\mathcal{S}}[<, succ, \equiv]$ where $\equiv$ are modulo predicates.

Theorem 3.11. *Let $\mathcal{S} \subseteq \mathcal{M}$ be a set of monoids. Let $L \subseteq \Sigma^*$ be a regular language, which is accepted by a morphism $h : \Sigma^* \to \mathcal{V}$, where $\mathcal{V}$ is a semigroup. Then the following are equivalent.*

1. *L is definable in $\mathcal{L}_{\mathcal{S}}[<, +]$*
2. *For all $k \in \mathbb{N}$, every group in $h(\Sigma^k)$ divides a monoid in $bpc(\mathcal{S})$.*
3. *L is definable in $\mathcal{L}_{\mathcal{S}}[\mathrm{REG}]$*

Proof. $(1 \Rightarrow 2)$: Consider a $k \in \mathbb{N}$ and a $G \in h(\Sigma^k)$. We first look at the language $h^{-1}(1_G)$. It is well known (for example, [Str94]) that $h^{-1}(1_G)$ can be written as a finite boolean combination of languages of the form $u^{-1}Lv^{-1}$, for strings $u, v \in \Sigma^*$. Now consider a new alphabet

$$\Gamma = \{a_w \mid w \in \Sigma^k, h(w) \in G\}$$

We can now define a mapping $f : \Gamma \to \Sigma^k$ as $f(a_w) = w$. Consider the language $L' = f^{-1}(h^{-1}(1_G))$. From Lemma 3.9 and Lemma 3.10 we know that L' is definable in $L_{\mathcal{S}}[<, +]$. But note that L' is a language with a neutral letter. The letter a_w, where $h(w) = 1_G$ acts as a neutral letter. Therefore L' is a language definable in $L_{\mathcal{S}}[<]$, which from Krohn Rhodes theorem **??** implies that G divides a monoid in $bpc(\mathcal{S})$.
$(2 \Rightarrow 3)$: Let us list down all the sets, $h(\Sigma), h(\Sigma^2), \ldots$. Since the subsets of $\mathcal{V}$ are finite, there exists $q, r \in \mathbb{N}$ such that $h(\Sigma^q) = h(\Sigma^{q+r})$. Choosing an $l = \mathrm{lcm}\{q, r\}$ we get that $h(\Sigma^l) = h(\Sigma^{2l})$. We can now split L into the following parts.

$$L = \bigcup_{u \in \Sigma^{<l}} u.\left(u^{-1}L \cap (\Sigma^l)^*\right)$$

Let us denote by $L_u = u^{-1}L \cap (\Sigma^l)^*$. First let us show that, if we can write L_u in $\mathcal{L}_{\mathcal{S}}[\mathrm{REG}]$, then we can write $u.L_u$ also in $\mathcal{L}_{\mathcal{S}}[\mathrm{REG}]$. Let us assume that the sentence $\phi_u \in \mathcal{L}_{\mathcal{S}}[\mathrm{REG}]$ models the language L_u. Let $u = a_1 \ldots a_{l'}$, where $l' < l$. Then the following formula models the language $u.L_u$.

$$\phi_u[> l'] \wedge a_1(1) \wedge \cdots \wedge a_{l'}(l')$$

Now since L is a disjunction of languages of the form uL_u we have that L can be written in $\mathcal{L}_{\mathcal{S}}[\mathrm{REG}]$.

We need to show that L_u is also definable. Consider the following alphabet $\Gamma = \{a_w \mid w \in \Sigma^l\}$. Let $M = h(\Sigma^l)$. Since $M = h(\Sigma^{2l})$ we have that M is closed

under concatenation and therefore a monoid and hence a monoid. Therefore we have that word problems over M are definable in $\mathcal{L}_M[<]$. Finally using Lemma 3.10 we get that L_u is definable in $\mathcal{L}_M[REG]$.
$(3 \Rightarrow 1)$: This holds, since addition can simulate all the regular predicates. □

Then we can identify the set of regular languages definable in $\mathcal{L}_\mathcal{S}[<,+]$ when $\mathcal{S}$ is a set of monoids.

COROLLARY 3.12. *Let $\mathcal{S}$ be a set of monoids. Then*

$$\mathcal{L}_\mathcal{S}[<,+] \cap \text{ REG} = \mathcal{L}_\mathcal{S}[\text{REG}]$$

Let $\mathcal{S}$ be a set of monoids such that, given a group G, it is decidable if G divides a monoid in *bpc*($\mathcal{S}$). Then, given a regular language L, it is decidable if $L \in \mathcal{L}_\mathcal{S}[<]$. Then our main theorem gives us that it is decidable if $L \in \mathcal{L}_\mathcal{S}[<,+]$.

COROLLARY 3.13. *Let $\mathcal{S}$ be a set of monoids such that, given a monoid G, it is decidable if G divides a monoid in bpc($\mathcal{S}$). Then, given a regular language L, it is decidable if $L \in L_\mathcal{S}[<,+]$.*

Proof. Since L is a regular language definable in $L_\mathcal{S}[<,+]$. Then L has the alternate characterization given by Theorem 3.11. Let h be a morphism which accepts L. Then we know that there exists $t, r \in \mathbb{N}$, such that $h(\Sigma^t) = h(\Sigma^{t+r})$, which implies that we can list down all the sets in $h(\Sigma^k)$, for $k \leq t+r$ and the groups in these sets. The claim now follows from the fact that there exists an algorithm to check whether each of these groups divide a monoid in *bpc*($\mathcal{S}$). □

For FO+MOD$[<,+]$ this was proved in [RS07] and the question when $\mathcal{S} = MOD$ was left open. The following corollary answers this special case.

COROLLARY 3.14. *Given a regular language L, the question whether L is definable in* MOD$[<,+]$ *is decidable.*

4. Proof Strategy.

For the purpose of proof we work over infinite strings which contain finite number of non-neutral letters. Our general proof strategy is similar to Benedikt and Libkin [BL00b] or Roy and Straubing [RS07] and consists of three main steps.

1. Given a formula $\phi \in \mathcal{L}_\mathcal{S}[<,+]$, we show that ϕ is "*weakly equivalent*" to an "active domain formula", that is, a formula which quantify only over *non-neutral letter positions*. Our major contribution (Theorem 5.6) is this step.
2. An active domain formula in $\mathcal{L}_\mathcal{S}[<,+]$ is weakly equivalent to an active domain formula in $\mathcal{L}_\mathcal{S}[<]$. This step (Theorem 5.7) follows from an application of Ramsey theory.
3. All active domain formulas in $\mathcal{L}_\mathcal{S}[<]$ accept languages with a neutral letter. This is an easy observation given by Lemma 5.8.

Using these three steps we show that if a formula in $\mathcal{L}_\mathcal{S}[<,+]$ is weakly equivalent to an active domain formula in $\mathcal{L}_\mathcal{S}[<]$, then it is infact equivalent. This proves our main Theorem.

The major part of the work is, Step (1). There we show how to simulate a general quantifier by an active domain formula. In the case of FO$[<,+]$, the existential quantifiers considered as Lindström quantifiers, have a commutative and idempotent monoid. Hence neither the order in which the quantifier runs over the positions of the word is important, nor does it matter if positions are queried multiple times. In Roy and Straubing this idea was extended in such a way that in the simulation of the MOD quantifier (again a commutative monoid), every position is taken into account exactly once. In their construction while replacing a MOD quantifier they need to

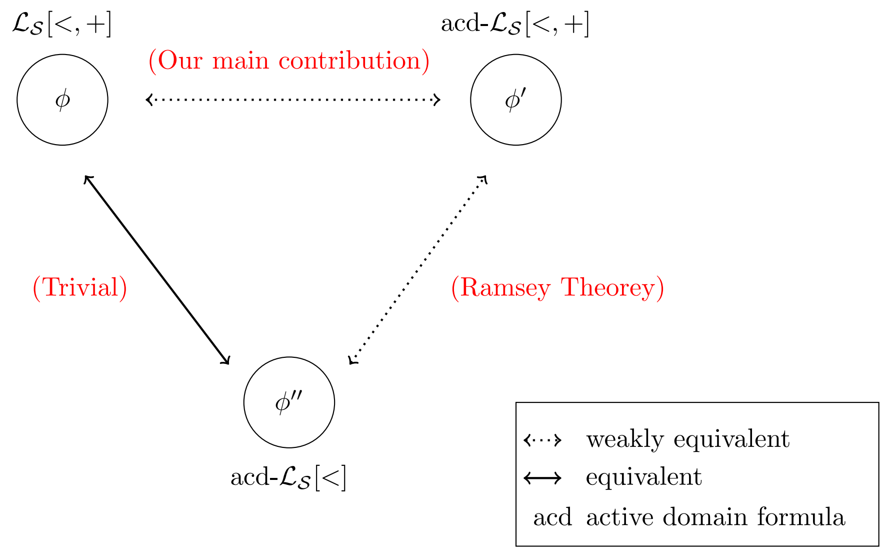
$\mathcal{L}_{\mathcal{S}}[<,+]$
acd-$\mathcal{L}_{\mathcal{S}}[<,+]$
(Our main contribution)
ϕ
ϕ'
(Trivial)
(Ramsey Theorey)
ϕ''
acd-$\mathcal{L}_{\mathcal{S}}[<]$
weakly equivalent
equivalent
acd active domain formula

FIG. 1. *The three step proof strategy.*

add additional FO quantifiers and hence their construction only allows to replace a MOD$[<,+]$ formula by an active domain FOMOD$[<,+]$ formula. In this paper, we construct a formula that takes every position into account exactly once and in the correct order. Moreover we do not introduce any new quantifier, but use only the quantifier that is replaced. This enables us to show the Crane Beach conjecture for logics whose quantifiers have a non-commutative monoid or are groups. For example MOD$[<,+]$, GROUP$[<,+]$, and FOGRP$[<,+]$.

In contrast to previous work, we do not construct an equivalent active domain formula, but only a formula that is equivalent for certain domains (or weakly equivalent formulas). We show that it is in general sufficient to show this for one infinite domain.

5. Proof of the Main Theorem. In this section we handle the general proof steps as in Libkin or Roy and Straubing of removing the plus predicate from the formula in the presence of a neutral letter. We show that all these results go through even in the presence of general Lindström quantifiers. The new crucial step is Lemma 5.4 where we convert a group quantifier to an active domain formula without introducing any other quantifiers. The proof of this lemma is deferred to the next section.

5.1. Definitions. Let $\mathcal{S} \subseteq \mathcal{M}$ be any nonempty set. To prove Theorem 3.1 we will consider the more general logic, $\mathcal{L}_{\mathcal{S}}[<,+,\{\equiv_q : q > 1\}]$ over the alphabet Σ. In this logic $+$ is a binary function, and $a \equiv_q b$ means q divides $b-a$. We will denote this logic by $\mathcal{L}_{\mathcal{S}}[<,+,\equiv]$. The relations $<$ and $\equiv_q$ are both definable using $+$. All languages recognized by this logic are definable in $\mathcal{L}_{\mathcal{S}}[<,+]$. The reason for introducing these new relations is to use a quantifier elimination procedure.

For the purpose of the proof we assume that the neutral letter language defined by a formula $\phi \in \mathcal{L}_{\mathcal{S}}[<,+]$ is a subset of $\Sigma^*\lambda^\omega$. The idea is to work with infinite words, where the arguments are easier, since the variable range is not bounded by the word length.

Definition 5.1. *Let $X \subseteq \mathbb{N}$ be an infinite set. We say that a formula ϕ' is* X-weakly equivalent *to a formula $\phi(x_1, \ldots, x_t)$ if there exists an infinite set $Y \subseteq X$, such that for all words $w \in \Sigma^*\lambda^\omega$, with $nnp(w) \subseteq Y$ and for all $a_1, \ldots, a_t \in \mathbb{N}$, we have that*

$$w \models \phi(a_1, \ldots, a_t) \Leftrightarrow w \models \phi'(a_1, \ldots, a_t)$$

In the above definition we say that Y collapse *ϕ to ϕ'.*

We say that a formula ϕ' is weakly equivalent *to a formula $\phi(x_1, \ldots, x_t)$ if for all infinite subsets $X \subseteq \mathbb{N}$, there exists an infinite set $Y \subseteq X$, such that for all words $w \in \Sigma^*\lambda^\omega$, with $nnp(w) \subseteq Y$ and for all $a_1, \ldots, a_t \in \mathbb{N}$, we have that*

$$w \models \phi(a_1, \ldots, a_t) \Leftrightarrow w \models \phi'(a_1, \ldots, a_t)$$

In this latter case we say that ϕ collapses *to ϕ'.* It is clear from the definition that if ϕ' is weakly equivalent to ϕ, then ϕ' is X-weakly equivalent to ϕ for any infinite set X. The other direction, namely if ϕ' is X-weakly equivalent to ϕ need not imply that ϕ' is weakly equivalent to ϕ.

We now define a special class of formulas, defined syntactically.

Definition 5.2. *Let an* active domain formula *over a letter $\lambda \in \Sigma$ be a formula where all quantifiers are of the form: $Q_M^m x\ \neg\lambda(x)\langle\phi_1, \ldots, \phi_K\rangle$. That is the quantifiers, quantify only over the "*active domain*", the positions which does not contain the letter λ.* Note that an active domain formula is defined with respect to a particular letter. Here we use the letter $\lambda \in \Sigma$, since the active domain formulas we will consider will always be with respect to the neutral letter of the language we will be looking at. Observe also that an active domain formula need not always define a language with a neutral letter.

Example 3. *Consider the following formula in* FO*[+].*

$$\exists x\ \neg\lambda(x)\ \wedge\ \exists y\ \neg\lambda(y)\ \wedge\ \exists z\ \neg\lambda(z)\ \ ((a(x) \wedge b(y) \wedge c(z)) \wedge x = y + z)$$

Clearly it is an active domain formula but does not define a language with a neutral letter.

The positions in a word which the quantifiers in an active domain formula access are going to be positions which have a non-neutral letter in it.

Definition 5.3. *The non-neutral letter positions of a word, w denoted by $nnp(w)$ is the set of all positions where the letter λ does not appear.*

$$nnp(w) = \{i \mid w(i) \neq \lambda\}$$

Observe that $nnp(w)$ is also defined with respect to a letter, namely λ. Let us look at the following example.

Example 4. *Let $w = a\lambda\lambda\lambda ba\lambda b\lambda\lambda a$. Then $nnp(w) = \{1, 5, 6, 8, 11\}$.*

5.2. The Proof. This subsection follows the proof outline as given in Figure 1.

We first show that any formula $\phi \in \mathcal{L}_{\mathcal{S}}[<, +, \equiv]$ will be weakly equivalent to an active domain formula $\phi' \in \mathcal{L}_{\mathcal{S}}[<, +, \equiv]$. The results by Benedikt and Libkin [BL00b], and Roy and Straubing [RS07] show that for all formulas $\phi \in \mathcal{L}_{\mathrm{MOD} \cup U_1}[<, +]$ there exists an active domain formula ϕ' in that logic, such that for all words $w \in \Sigma^*\lambda^\omega$, $w \vDash \phi \Leftrightarrow w \vDash \phi'$. They assume no restriction on the non-neutral positions of w. Observe that our collapse result is different from theirs. We prove that if we consider only words, whose non-neutral positions come from a particular subset of $\mathbb{N}$, then $\phi \in \mathcal{L}_{\mathcal{S}}[<,+]$ is equivalent to an active domain formula $\phi' \in \mathcal{L}_{\mathcal{S}}[<,+,\equiv]$. That is, we are not concerned about the satisfiability of those words whose non-neutral positions are not from that particular subset.

Let us first consider formulas with an outermost group quantifier, $G \in \mathcal{S}$.

Lemma 5.4. *Let $\phi = Q^m_G z\langle\phi_1, \dots, \phi_K\rangle$ be in $\mathcal{L}_{\mathcal{S}}[<,+,\equiv]$. Let us assume that there exists active domain formulas $\phi'_1, \dots, \phi'_k$ in the same logic such that for all $i \leq K$ we have ϕ_i is weakly equivalent to ϕ'_i.*
Then ϕ is weakly equivalent to an active domain formula $\phi' \in \mathcal{L}_{\mathcal{S}}[<,+,\equiv]$.

The proof of Lemma 5.4 will be given in Section 6. Benedikt and Libkin [BL00b] gives a similar theorem for the monoid U_1 (the existential quantifier).

Lemma 5.5 ([BL00b]). *Let $\phi = Q^1_{U_1} z\langle\phi_1\rangle$ be a formula in $\mathcal{L}_{\mathcal{S}}[<,+,\equiv]$. Let us assume that the formula ϕ_1 is weakly equivalent to an active domain formula ϕ'_1 in the same logic.*
Then ϕ is weakly equivalent to an active domain formula $\phi' \in \mathcal{L}_{\mathcal{S}}[<,+,\equiv]$.

The following theorem proves the first step of our 3 step proof strategy of Figure 1.

Theorem 5.6. *Let $\phi \in \mathcal{L}_{\mathcal{S}}[<,+,\equiv]$. Then there exists an active domain formula $\phi' \in \mathcal{L}_{\mathcal{S}}[<,+,\equiv]$ such that ϕ is weakly equivalent to ϕ'.*

Proof. Let $\phi \in \mathcal{L}_{\mathcal{S}}[<,+,\equiv]$. We first claim that we can convert ϕ into a formula which uses only groups and U_1 as quantifiers. This follows from the Krohn-Rhodes decomposition theorem for monoids that every monoid can be decomposed into block products over groups and U_1. This decomposition can then be converted back into a formula using the groups and U_1 as quantifiers [Str94].

So without loss of generality we can assume ϕ has only group or U_1 quantifiers. The proof is by induction on the quantifier depth. For the base case, let ϕ be a quantifier free formula. It is an active domain formula and therefore the claim holds. Let the claim be true for all formulas with quantifier depth $< d$. Lemma 5.4 and Lemma 5.5 show that the claim is true for formulas of type $\phi = Q^m_M z\langle\phi_1, \dots, \phi_K\rangle$ with quantifier depth d, when M is a group or U_1 respectively. We are now left with proving that the claim is closed under conjunction and negation. So assume that formulas ϕ_1, ϕ_2 is weakly equivalent to ϕ'_1, ϕ'_2 respectively. That is, for all $X \subseteq \mathbb{N}$, there exist $\mathcal{R}_{\phi_1} \subseteq X, \mathcal{R}_{\phi_2} \subseteq \mathcal{R}_{\phi_1}$ such that $\mathcal{R}_{\phi_1}$ is weakly equivalent to ϕ_1 to ϕ'_1 and $\mathcal{R}_{\phi_2}$ collapses ϕ_2 to ϕ'_2. Then it is easy to see that $\mathcal{R}_{\phi_2}$ collapses $\phi_1 \wedge \phi_2$ to $\phi'_1 \wedge \phi'_2$ and $\mathcal{R}_{\phi_1}$ collapses $\neg\phi_1$ to $\neg\phi'_1$. □

We have shown above that all formulas in $\mathcal{L}_{\mathcal{S}}[<,+,\equiv]$ can be collapsed to active domain formulas. Now using a *Ramsey* type argument we obtain that addition is useless, giving us a formula in $\mathcal{L}_{\mathcal{S}}[<]$. This corresponds to the second step in our three step proof strategy.

Let R be any set of relations on $\mathbb{N}$ and let $\phi(x_1, \dots, x_t)$ be an active domain formula in $\mathcal{L}_{\mathcal{S}}[R]$. Let $X \subseteq \mathbb{N}$ be an infinite set. Then there exists a formula ϕ' in $\mathcal{L}_{\mathcal{S}}[<]$ such that ϕ is X-weakly equivalent to ϕ'.

Weak equivalence for first order logic has been considered by Libkin [Lib04], where the notion of weak equivalence is known as *Ramsey property*. We show that this result can be extended to our logic.

Theorem 5.7. *Let R be a set of relations on $\mathbb{N}$. Let $X \subseteq \mathbb{N}$ be an infinite set. Then every active domain sentence in $\mathcal{L}_{\mathcal{S}}[R]$ is X-weakly equivalent to an active domain formula in $\mathcal{L}_{\mathcal{S}}[<]$.*

Proof. Let $\phi \in \mathcal{L}_{\mathcal{S}}[R]$ be an active domain sentence. We now prove by induction on the structure of the formula. Let $P(x_1, \dots, x_k)$ be a term in ϕ. Consider the infinite complete hypergraph, whose vertices are labelled by numbers from X and whose edges are all k tuple of vertices. Let $P(a_1, \dots, a_k)$ be true, for $a_1, \dots, a_k \in X$ and let the *order type* of $a_1, \dots, a_k$ be o. Then we color the edge $(a_1, \dots, a_k)$ by the quantifier free formula on $x_1, \dots, x_k$ which describes the order type o. For example, if the order type is $a_2 = a_3 < a_1 < \dots < a_k$, then the formula will be $x_2 = x_3 < x_1 < \dots < x_k$. Observe that an edge can have multiple colors but the total number of different colorings possible is dependent only on k (a constant). Ramsey theory, now gives us that there exists an infinite set $Y \subseteq X$, such that the induced subgraph on the vertices in Y will have a monochromatic color, ie. all the edges will be colored using the same color or in other words, there exists an order type which is true for all the edges in this subgraph. Let us assume that the edges in Y are colored $x_1 < x_2 < \dots < x_k$. Then for all $a_1, \dots, a_t \in Y$

$$a_1, \dots, a_t \models P(x_1, \dots, x_k) \Leftrightarrow a_1, \dots, a_t \models x_1 < x_2 < \dots < x_k$$

This shows that $P(x_1, \dots, x_k)$ satisfies the Ramsey property and thus all atomic formulas satisfy the Ramsey property. We now show that Ramsey property is preserved while taking Boolean combination of formulas. Consider the formula $\phi_1(x_1, \dots, x_k) \wedge \phi_2(x_1, \dots, x_k)$. We know that by induction hypothesis there exists a formula ψ_1 and an infinite set X such that for all $a_1, \dots, a_t \in X$, $w \models \phi_1(a_1, \dots, a_t) \Leftrightarrow w \models \psi(a_1, \dots, a_t)$. We can now find an infinite set $Y \subseteq X$ and a formula ψ_2 such that the Ramsey property holds for the formula ϕ_2. Therefore for all $a_1, \dots, a_t \in Y$

$$w, a_1, \dots, a_k \vDash \phi_1 \wedge \phi_2 \Leftrightarrow w, a_1, \dots, a_k \vDash \psi_1 \wedge \psi_2$$

Similarly we can show that the Ramsey property holds for disjunctions and negations. We need to now show that active domain quantification also preserves Ramsey property. So let X be an infinite subset of $\mathbb{N}$ and let

$$\phi(\vec{x}) = Q_M^m z \ \neg\lambda(z) \ \langle \phi_1(z, \vec{x}), \dots, \phi_K(z, \vec{x}) \rangle$$

be a formula in $\mathcal{L}_{\mathcal{S}}[R]$. By induction hypothesis we know that there exists an infinite set $Y_1 \subseteq X$ and an active domain formula $\psi_1 \in \mathcal{L}[<]$ such that for all $\vec{a} \in Y_1^t$ the Ramsey property is satisfied. That is $w \models \phi_1(\vec{a}) \Leftrightarrow w \models \psi_1(\vec{a})$. Now for ϕ_2, using the infinite set Y_1 we can find an infinite set $Y_2 \subseteq Y_1$ and a formula ψ_2 satisfying the Ramsey property. Continuing like this will give us a set Y_K and formulas $\psi_1, \dots, \psi_K$ such that $\forall j \leq K$ and for all $w \in \Sigma^* \lambda^\omega$ with $\text{nnp}(w) \subseteq Y_K$, we have that $\forall b \in Y_K, \vec{a} \in Y_K^t,\ w \vDash \phi_j(b, \vec{a}) \Leftrightarrow w \vDash \psi_j(b, \vec{a})$. Hence we also have that $\forall j \leq K$

$$\{b \in Y_K \mid w \vDash \phi_j(b, \vec{a})\} = \{b \in Y_K \mid w \vDash \psi_j(b, \vec{a})\}$$

Therefore for the formula $\psi = Q_M^m z \ \neg\lambda(z) \ \langle \psi_1, \dots, \psi_K \rangle$, we have $\forall w$ where $\text{nnp}(w) \subseteq Y_K$ and $a_1, \dots, a_t \in Y_K$ that

$$w \vDash \phi(a_1, \dots, a_t) \Leftrightarrow w \vDash \psi(a_1, \dots, a_t)$$

Observe that ψ is an active domain formula in $\mathcal{L}_{\mathcal{S}}[<]$. □

We continue with the third step of our three step proof strategy.

LEMMA 5.8. *Every active domain sentence in $\mathcal{L}_{\mathcal{S}}[<]$ define a language with a neutral letter.*

Proof. Let $\phi \in \mathcal{L}_{\mathcal{S}}[<]$ be an active domain formula over letter $\lambda \in \Sigma$. Let $w \in \Sigma^\omega$. Let $w' \in \Sigma^\omega$ got by inserting letter λ in w at some positions. Let $n_1 < n_2 < \dots$ belong to $\text{nnp}(w)$ and $m_1 < m_2 < \dots$ be in $\text{nnp}(w')$. Let $\rho : \text{nnp}(w) \to \text{nnp}(w')$ be the bijective map $\rho(n_i) = m_i$. We show that for any subformula ψ of ϕ and any $\vec{t} \in \text{nnp}(w)^s$, we have that $w, \vec{t} \vDash \psi \Leftrightarrow w', \rho(\vec{t}) \vDash \psi$. Since the variables quantifier only over the active domain, the claim holds for the atomic formula $x > y$, because $n_i > n_j$ iff $\rho(n_i) > \rho(n_j)$ for any i, j. Similarly the claim also hold for all other atomic formulas $x < y, x = y$ and $a(x)$ for an $a \in \Sigma$. The claim remains to hold under conjunctions, negations and active domain quantifications. Hence $w \models \phi \Leftrightarrow w' \models \phi$. This proves that λ is a neutral letter for $L(\phi)$. □

Now we can prove our main theorem. This step shows that if an active domain formula $\phi' \in \mathcal{L}_{\mathcal{S}}[<]$ is X-weakly equivalent to a formula $\phi \in \mathcal{L}_{\mathcal{S}}[<,+]$ then ϕ' is infact equivalent to ϕ.

Proof. [Proof of Theorem 3.1] Let $\phi \in \mathcal{L}_{\mathcal{S}}[<,+]$, such that $L(\phi)$ is a language with the neutral letter, λ. By Theorem 5.6 there exists an active domain sentence $\phi' \in \mathcal{L}_{\mathcal{S}}[<,+,\equiv]$ and a set $\mathcal{R} \subseteq_\infty \mathbb{N}$ such that $\mathcal{R}$ collapses ϕ to ϕ'. Theorem 5.7 now gives an active domain formula $\psi \in \mathcal{L}_{\mathcal{S}}[<]$ and an infinite set $Y \subseteq_\infty \mathcal{R}$ such that ψ is Y-weakly equivalent to ϕ'. We now show that $L(\phi) = L(\psi)$. Let $w \in \Sigma^*\lambda^\omega$. Consider the word $w' \in \Sigma^*\lambda^\omega$ got by inserting the neutral letter λ in w in such a way that $\text{nnp}(w') \subseteq Y$. Since $L(\phi)$ is a language with a neutral letter we have that $w \models \phi \Leftrightarrow w' \vDash \phi$. From Theorem 5.6 and Theorem 5.7 we get $w' \vDash \phi \Leftrightarrow w' \vDash \phi' \Leftrightarrow w' \vDash \psi$. Finally as shown in Lemma 5.8, ψ defines a language with a neutral letter and hence $w' \models \psi \Leftrightarrow w \models \psi$. □

6. Proof of Lemma 5.4. In this section we replace a group quantifier by an active domain formula. Here we make use of the fact that we can a priory restrict our domain as shown in the previous section.

Recall that $\phi = Q_G^m z\langle\phi_1, \dots, \phi_K\rangle$ and $G = \{m_1, \dots, m_K, 1\}$. Let $X \subseteq \mathbb{N}$ be an arbitray infinite set. We show that there exists an infinite set $R_\phi \subseteq X$ and an active domain formula $\phi' \in \mathcal{L}_{\mathcal{S}}[<,+,\equiv]$ such that R_ϕ collapse ϕ to ϕ'. By the assumption of the Lemma, we know that for all $i \leq K$, there exists active domain formulas ϕ'_i such that ϕ_i is weakly equivalent to ϕ'_i. Therefore there exists an infinite set $R_{\phi_1} \subseteq X$ such that for all words w where $nnp(w) \subseteq R_{\phi_1}$ we have that $w \models \phi_1 \Leftrightarrow w \models \phi'_1$. Similarly we can find infinite sets $R_{\phi_K} \subseteq R_{\phi_{K-1}} \subseteq \dots \subseteq R_{\phi_1} \subseteq X$ such that for all $i \leq K$, and for all words w where $nnp(w) \subseteq R_{\phi_K}$, we have that $w \models \phi_i \Leftrightarrow w \models \phi'_i$. So without loss of generality we assume ϕ_is are active domain formulas. In this section, we will find an active domain formula ϕ' and an infinite set $R_\phi \subseteq R_{\phi_K}$, such that ϕ is weakly equivalent to ϕ'.

Before we go in the details we will give a rough overview of the proof idea. The group quantifier evaluates the product $\prod_j u(j)$, where $u(j)$ is a group element that depends on the set of i such that $w, j \models \phi_i$. So we start and analyze the sets $J_i = \{j \mid w, j \models \phi_i\}$. Since the formulas ϕ_is are active domain formulas, we will see that there are certain positions in the word called "*boundary points*" which are crucial. We see that in between two boundary points, the set J_i is periodic. In the construction of the active domain formula for ϕ we show how to iterate over these boundary points in a strictly increasing order. An active domain quantifier can only iterate over active domain positions, hence we will need nested active domain quantifiers, and a way how to "encode" the boundary points by tuples of active domain positions in a unique and order preserving way. Additionally we need to deal with the periodic positions inside the intervals, without being able to compute the length of such an interval, or even check if the length is zero. Here will make use of the inverse elements that always exist in groups.

We start by analyzing the intervals which occur. Since we consider a fixed set $\mathcal{S}$ for the rest of the paper, we will write $\mathcal{L}[<,+]$ for the logic $\mathcal{L}_{\mathcal{S}}[<,+,0,\{\equiv_q\colon q>1\}]$.

6.1. Intervals and Linear Functions. We first show that every formula ψ with at least one free variable has a normal form. This step is a standard procedure in Presburger's *quantifier elimination* technique [Pre29],[End72].

LEMMA 6.1. *Let $\psi(z) \in \mathcal{L}[<,+]$. Then there exists a formula $\hat{\psi}(z) \in \mathcal{L}[<,+]$ such that ψ is equivalent to $\hat{\psi}$, where all atomic formulas in $\hat{\psi}$ with z are of the form $z>\rho, z=\rho, z<\rho, z\equiv_n \rho$, where ρ is a linear function on variables other than z.*

Proof. Terms in our logic are expressions of the form

$$\alpha_0+\alpha_1x_1+\cdots+\alpha_sx_s \qquad \text{, where } \ \alpha_i\in\mathbb{N}$$

and atomic formulas are of the form

$$\sigma=\gamma, \sigma<\gamma, \sigma>\gamma, \sigma\equiv_m\gamma, c(\sigma)$$

whee σ,γ are linear functions, $c\in\Sigma$ and $m>1$.
Now using any $M\in\mathcal{S}$, where $m_1\in M$ is not the identity element, we can rewrite $c(\sigma)$ by an equivalent formula

$$Q_M^{m_1}x\ \neg\lambda(x)\langle(x=\sigma)\wedge c(x), false,\ldots,false\rangle$$

Now consider the atomic formulas containing the free variable z in $\psi(z)$. By multiplying with appropriate numbers, we can re-write these atomic formulas as $nz=\rho, nz<\rho, nz>\rho, nz\equiv_l\rho$ for one particular n, which is the least common multiple (lcm) of all the coefficients in ψ. Here ρ does not contain z and also it might contain subtraction. That is $nz=\rho$ might stand for $nz+\rho_1=\rho_2$. Now we replace nz by z and conjunct the formula with $z\equiv_n 0$. □

For any formula $\psi(z)$, the notation $\hat{\psi}(z)$ denotes the normal form as in Lemma 6.1. Let $x_1,\ldots,x_s$ be the bounded variables occurring in $\hat{\phi}_i(z)$ and $y_1,\ldots,y_r$ be the free variables other than z in $\hat{\phi}_i(z)$. Hence the terms ρ that appear in the formula $\hat{\phi}_i(z)$ can be identified as functions, : $\mathbb{N}^{s+r}\to\mathbb{N}$.

We collect all functions $\rho(\vec{x},\vec{y})$ that occur in the formulas $\hat{\phi}_i(z)$ for an $i\le K$:

$$R=\{\rho\mid \text{where } \rho \text{ is a linear term occurring in } \hat{\phi}_i(z), i\le K\}$$

We define the set T of offsets as a set of terms which are functions using the variables $y_1,\ldots,y_r$ as parameters:

$$T=\{\rho(0,\ldots,0,y_1,\ldots,y_r)\mid \rho\in R\}\cup\{0\}$$

Consider the set of absolute values of all the coefficients appearing in one of the functions in R. Let $\alpha'\in\mathbb{N}$ be the maximum value among these. That is $\alpha'=max\{|\gamma|\mid f\in R, \gamma \text{ is a coefficient in } f\}$. Let $\Delta=s\cdot\alpha'$. Now we can define our set of extended functions. For a $t\in T$ we define a set of terms which are functions using the variables $x_1,\ldots,x_s,y_1,\ldots,y_r$ as parameters:

$$F_t=\Big\{\sum_i^{s'}\alpha_ix_i+t\mid s'\le s, -\Delta\le\alpha_i\le\Delta, \alpha_i\in\mathbb{N}\Big\}.$$

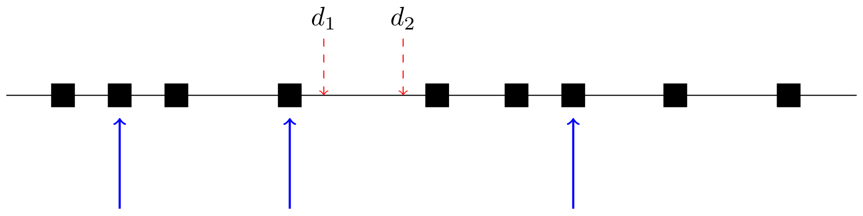

FIG. 2. *The up arrows point to the* active domain *and the boxes are the* boundary points*,* $B^{w,\vec{a}}$*. We have marked two points* d_1 *and* d_2 *in the same* interval*. All points in an interval has the neutral letter.*

We denote by $F = \cup_{t \in T} F_t$.

For a fixed word $w \in \Sigma^* \lambda^\omega$ and a fixed assignment of the free variables $\vec{y}$ to $\vec{a}$ we let

$$B^{w,\vec{a}} = \{f(\vec{d}, \vec{a}) \mid t \in T, f \in F_t, \vec{d} \in \mathrm{nnp}(w)^{s'}, d_1 > d_2 > \cdots > d_{s'}\}$$

be the set of *boundary points*. Note that the assignments to the functions are of strictly decreasing order. Let

$$b_1 < b_2 < \ldots < b_l$$

be the boundary points in $B^{w,\vec{a}}$. Then the following sets are called *intervals*:

$$(0, b_1), (b_1, b_2), \ldots, (b_{l-1}, b_l), (b_l, \infty)$$

Here $(a, b) = \{x \in \mathbb{N} \mid a < x < b\}$ and (b_l, ∞) is called the *infinite interval* . We also split the set of points in $B^{w,\vec{a}}$ depending on the offset

$$B_t^{w,\vec{a}} = \{f(\vec{d}, \vec{a}) \mid f \in F_t, \vec{d} \in \mathrm{nnp}(w)^{s'}, d_1 > d_2 > \cdots > d_{s'}\}.$$

We fix a word $w \in \Sigma^* \lambda^\omega$ and an $\vec{a} \in \mathbb{N}^r$. Therefore we drop the superscripts in $B^{w,\vec{a}}$ ($B_t^{w,\vec{a}}$) and call them B (B_t). Figure 2 illustrates some of the definitions.

The following lemma shows that all points the linear terms in $\hat{\phi}_i$s point to are in $B^{w,\vec{a}}$.

LEMMA 6.2. $\{\rho(d_1, \ldots, d_s, \vec{a}) \mid \rho \in R, d_i \in nnp(w)\} \cup \mathrm{nnp}(w) \subseteq B^{w,\vec{a}}$

Proof. Let $S = \{\rho(d_1, \ldots, d_s, \vec{a}) \mid \rho \in R, d_i \in nnp(w)\} \cup \mathrm{nnp}(w)$. Since $\rho(x_1) = x_1$ is in F_t, for some $t \in T$, we have $nnp(w) \subseteq B^{w,\vec{a}}$. Let $b \in S$. Then there is a function $\rho = \sum_i^{s'} \alpha_i x_i + t(\vec{y})$ in F_t and values $p_1, \ldots, p_{s'} \in nnp(w)$ such that $b = \rho(p_1, \ldots, p_{s'}, \vec{a})$. Let $p'_1 > p'_2 > \cdots > p'_l$ be the ordered set of all p_is in the above assignment. We let $\rho'(x_1, \ldots, x_l) = \sum_i \beta_i x_i + t$, where $\beta_i = \sum_{j : p_j = p'_i} \alpha_j$. Therefore $b = \rho'(p'_1, \ldots, p'_l)$. Since $|\beta_i| \leq \Delta \cdot s$ we have $\rho' \in F_t$ and hence $b \in B_t^{w,\vec{a}}$. ◻

Let us try to understand the definitions and the lemma seen above. Note that all the quantifiers in $\hat{\phi}_i$s are active domain quantifiers. Thus the bounded variables in $\hat{\phi}_i$s will only be evaluated at the active domain points in w. Now consider a linear term $\rho(x_1, \ldots, x_s, y_1, \ldots, y_r)$ present in one of the $\hat{\phi}_i$s. Observe that once we assign the free variables to $\vec{a}$, then the bounded variables $x_1, \ldots, x_s$ will run over all the active domain points in w. We therefore look at all the points ρ points to, if its bounded variables $x_1, \ldots, x_s$ are assigned values from $nnp(w)$. The importance of these points will be clear once we see Lemma 6.3. Lemma 6.2 shows that all such points are infact inside the set $B^{w,\vec{a}}$. That is $B^{w,\vec{a}}$ over-approximates the set of points we want. Lemma 6.3 also shows that the over-approxmiation also preserves the properties we are looking for. Later we will see that this over-approximation is more suitable to build active domain formulas which are weakly equivalent to ϕ.

Let q be the lcm of all q' where $\equiv_{q'}$ occurs in one of the ϕ_i. We need the following lemma, that inside an interval with only neutral letters, the congruence relations decide the truth of an active domain formula.

Lemma 6.3. *For the fixed word w and $a_1, \ldots, a_r \in \mathbb{N}$ and let $c, d \in \mathbb{N}$ belong to the same interval in $B^{w,\vec{a}}$ such that $c \equiv_q d$. Then for all $i \le K$: $w, c \vDash \phi_i(z, \vec{a}) \Leftrightarrow w, d \vDash \phi_i(z, \vec{a})$.*

Proof. Proof is by induction on the structure of the formula $\hat{\phi}_i$. We will now show that $\forall b_i \in \text{nnp}(w)$ and all subformulas $\psi(z, \vec{x}, \vec{y})$ of $\hat{\phi}_i$ that $w, c, \vec{b}, \vec{a} \vDash \psi \Leftrightarrow w, d, \vec{b}, \vec{a} \vDash \psi$. The atomic formulas of $\hat{\phi}_i(z, \vec{a})$ are of the following form: $z < \rho(\vec{x}, \vec{a}), z = \rho(\vec{x}, \vec{a}), z > \rho(\vec{x}, \vec{a}), z \equiv_{q'} \rho(\vec{x}, \vec{a}), a(z)$ and formulas which does not depend on z. It is clear that the truth of formulas which does not depend on z, $a(z)$ and $z \equiv_{q'} \rho$ does not change whether we assign c or d to z. For example, $w, c \models a(z) \Leftrightarrow w, d \models a(z)$, since inside an interval we see only the neutral letter. Let $\vec{b} \in nnp(w)^s$. By Lemma 6.2 we know that $\rho(\vec{b}, \vec{a})$ is in $B^{w,\vec{a}}$ and since c, d lies in the same interval it follows that $c < \rho(\vec{b}, \vec{a}) \Leftrightarrow d < (\vec{b}, \vec{a})$. Similarly we can show that the truth of $z > \rho, z = \rho$ does not change on z being assigned c or d. Thus we have that the claim holds for atomic formulas. The claim clearly holds for conjunction and negation of formulas. Now let the claim hold for subformulas $\psi_1, \ldots, \psi_K$. Therefore $\forall i \le K$ we have that $\{\vec{b} \in \text{nnp}(w)^s \mid w, c, \vec{b}, \vec{a} \vDash \psi_i\} = \{\vec{b} \in \text{nnp}(w)^s \mid w, d, \vec{b}, \vec{a} \vDash \psi_i\}$. Therefore we have that

$$\begin{aligned} &w, c, b_2, \ldots, b_s, \vec{a} \vDash Q_M^m x\ \neg\lambda(x)\langle\psi_1, \ldots, \psi_K\rangle \\ \Leftrightarrow\ &w, d, b_2, \ldots, b_s, \vec{a} \vDash Q_M^m x\ \neg\lambda(x)\langle\psi_1, \ldots, \psi_K\rangle \end{aligned}$$

And hence it is closed under active domain quantification. ◻

The above lemma says that the two points d_1 and d_2 in Figure 2 satisfy the same set of formulas ϕ_i if $d_1 \equiv_q d_2$. In other Lemma 6.3 says that inside an interval, only the congruence relations can change the satisfiability of the ϕ_is. Thus, it is enough to know the truth values of ϕ_i at a distance of $\ge q$ from the boundary points, since the truth values inside an interval are going to repeat after every q positions.

The following Lemma deals with the infinite interval.

Lemma 6.4. *Let b belong to the infinite interval and $\vec{a} \in \mathbb{N}^r$. If $w, \vec{a} \vDash \phi$ then $w, b, \vec{a} \nvDash \phi_i$ for any $i \le K$.*

Proof. Let $i \le K$ and b be in the infinite interval and $w, b, \vec{a} \vDash \phi_i$. From Lemma 6.3 we know that all points $c \equiv_q b$ and such that c is also in the infinite interval will be a witnesses for ϕ_i. This means the set of witnesses is infinite and hence $w, \vec{a} \nvDash \phi$. ◻

Let us first understand the importance of Lemma 6.3 and 6.4. Recall from the Preliminaries (Section 2) that we denoted by $u(i)$ the group element at position i. That is

$$u(i) = m_j \text{ iff } w, \vec{a} \models \phi_j \wedge \bigwedge_{l<j} \neg\phi_l$$

For a $b \in B$, we define the function $\text{IL}(b)$ to be the length of the interval to the left of b. That is if (b', b) form an interval then $\text{IL}(b) = b - b' - 1$. Similarly we define $\text{IR}(b)$

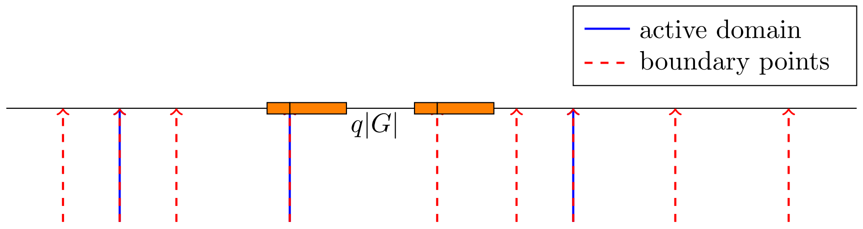

Fig. 3. *The only points of interest in a word are a fixed area around the boundary points (shaded area shown here). The length of the unshaded area is congruent to $W = q|G|$.*

to be the length of the interval to the right of b. Let $W = q|G|$. We now define two group values.

$$Post(b) = \begin{cases} u(b+1)u(b+2)\dots u(b+\mathrm{IR}(b)) & \text{if } \mathrm{IR}(b) < W \\ u(b+1)u(b+2)\dots u(b+r) & \\ & \text{if } \mathrm{IR}(b) \geq W \text{ and } r < W, b+r \equiv_W 0 \end{cases}$$

$$Pre(b) = \begin{cases} 1_G & \text{if } \mathrm{IL}(b) < W \\ u(b-r+1)\dots u(b-1) & \\ & \text{if } \mathrm{IL}(b) \geq W \text{ and } r < W, b-r \equiv_W 0 \end{cases}$$

We define

$$N_0(b) = Pre(b).u(b).Post(b)$$

The Figure 3 shows the important points around the boundary points. The following Lemma shows that a finite interval around the boundary points are the "only" points of interest to us.

Lemma 6.5. *Let us assume that for all $b \in \mathbb{N}$ in the infinite interval, $u(b) = 1_G$. Then*

$$\prod_{i=1}^{\infty} u(i) = \prod_{i \in B} N_0(i)$$

Proof. Let $b_0 < b_1 < \cdots < b_x$ be the positions in B. Then

$$\prod_{b=b_0}^{b_x} N_0(b) = Pre(b_0)u(b_0)\left(\prod_{i=0}^{x-1} Post(b_i)Pre(b_{i+1})u(b_{i+1})\right)Post(b_x) \tag{6.1}$$

Now consider an interval (b_i, b_{i+1}). We show that $Post(b_i)Pre(b_{i+1}) = \prod_{j=b_i+1}^{b_{i+1}-1} u(j)$. There are two cases to consider

- Case $b_{i+1} - b_i < W$: Then

$$Post(b_i)Pre(b_{i+1}) = \big(u(b+1)u(b+2)\dots u(b+\mathrm{IR}(b))\big)(1_G) = \prod_{j=b_i+1}^{b_{i+1}-1} u(j)$$

- Case $b_{i+1} - b_i \geq W$: Let $s, t \in \mathbb{N}$, be such that $s, t < W$ and $b_i + s \equiv_W b_{i+1} - t \equiv_W 0$. Lemma 6.3 shows that inside an interval all positions congruent modulo q satisfy the same formulas and hence the product of group elements of any $W = q|G|$ consecutive positions evaluate to the identity element. Therefore $u(b_i + s + 1)u(b_i + s + 2)\dots u(b_{i+1} - t) = 1_G$. So

$$Post(b_i)Pre(b_{i+1}) = \big(u(b_i{+}1)u(b_i{+}2)\dots u(b_i{+}s)\big)\big(u(b_{i+1}{-}t{+}1)\dots u(b_{i+1}{-}1)\big) = \prod_{j=b_i+1}^{b_{i+1}-1} u(j)$$

Now substituting the value of $Post(b_i)Pre(b_{i+1})$ for all $i < x$ on equation 6.1 will give us the claim. □

We view $N_0(b)$ as a group value at a point b. Observe that the cardinality of B is finite, even though it might depend on the length of the word w. The following lemma helps us understand when does w be a model of ϕ. Recall that $\phi = Q_G^m z\langle\phi_1, \dots, \phi_K\rangle$.

Lemma 6.6.

$$w, \vec{a} \models \phi \Leftrightarrow \prod_{i \in B} N_0(i) = m \textit{ and for all } b \textit{ in the infinite interval } u(b) = 1_G$$

Proof. $(\Rightarrow)$: Since $w \models \phi(\vec{a})$ we have by Lemma 6.4 that for all b in the infinite interval $u(b) = 1_G$. Then the claim follows from Lemma 6.5.
$(\Leftarrow)$: Let the right side be true. Then by Lemma 6.5 we have that $\prod_i^\infty u(i) = m$. The claim now follows from the definition of the group quantifier. □

So it remains to show that there exists an active domain formula which can multiply the group values $N_0(i)$ in the correct order and also that there exists an active domain formula to check whether $u(b) = 1$, for all b in the infinite interval. The major part of the work is in showing the former. For this we need to go through the points in B in an increasing order. The rest of the proof demonstrates

1. How we can treat each B_t differently.
2. There is an active domain formula which goes through the points in B_t in an increasing order

6.2. Treating each B_t differently. Recall the definition of $N_0(b)$ from the previous section.

Our aim is to give an active domain formula such that the formula evaluates to true iff the group element $\prod_{i=0} u(i)$ is equal to m. The rest of this subsection will be devoted to computing this product in a way which helps in building an active domain formula.

Let $b < b'$ be boundary points in B. Below we compute $\prod_{i=b}^{b'-1} N_0(i)$ in a different way:

$$\prod_{i=b}^{b'-1} N_0(i) = \prod_{i \geq b} N_0(i) \left(\prod_{i \geq b'} N_0(i)\right)^{-1}.$$

Observe that we can compute the product of the interval using two terms that both need to know only one boundary of the interval. It becomes simpler if we note that the two products do not really need to multiply all the elements $u(i)$, for $i \geq b'$ but simply agree on a common set of elements to multiply.

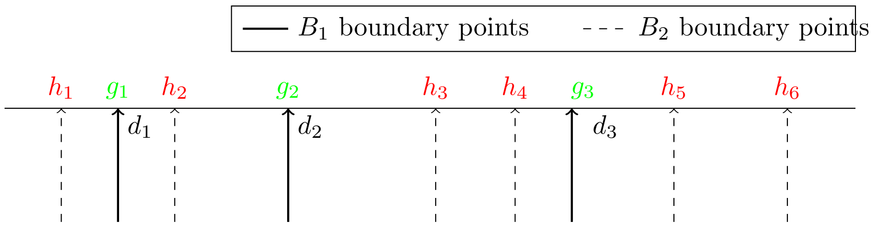

FIG. 4. *Boundary points and group values there.*

The following example will help us understand the method better. Consider Figure 4. The arrows point to boundary points. The **bold** arrows point to the boundary points B_1 and the dashed arrows point to the boundary points B_2. Consider the point marked d_1 and our aim is to compute the product of the group elements to the right of it. We compute the product in the following way.

- n_{d_1} = Product of group elements in $B_2 > d_1 = h_2.h_3.h_4.h_5.h_6$.
- n_{d_2} = Product of group elements in $B_2 > d_2 = h_3.h_4.h_5.h_6$.
- n_{d_3} = Product of group elements in $B_2 > d_3 = h_5.h_6$.
- m_{d_1} = (Group element at d_1)$\times n_{d_1} = g_1.h_2.h_3.h_4.h_5.h_6$.
- $m_{d_2} = n_{d_2}^{-1}\times$(Group element at d_2) $\times n_{d_2} = (h_3.h_4.h_5.h_6)^{-1}g_2(h_3.h_4.h_5.h_6)$.
- $m_{d_3} = n_{d_3}^{-1}\times$ (Group element at d_3) $\times n_{d_3} = (h_5.h_6)^{-1}g_3(h_5.h_6)$
- $m_{d_1}.m_{d_2}.m_{d_3}$ gives product of group elements in interval $[d_1,\infty) = g_1.(h_1).g_2.(h_3.h_4).g_3.(h_5.h_6)$.

In the above example we inductively assumed that, starting from any point d_1, we know the product $\prod_{i\in B_2, i>d_1} u(i)$. We show that if we can go through the boundary points in B_1 in an increasing order, then we can compute the product of the group elements at both B_1 and B_2.

We now formalize the above idea. Firstly we define functions which computes the product of the group values for the set B_{t_1}.

$$N_1(b) = \prod_{\substack{b'\in B_{t_1}\\ b'\geq b}} N_0(b')$$

$$\hat{N}_1(b) = \prod_{\substack{b'\in B_{t_1}\\ b'> b}} N_0(b')$$

Note that $N_1(b)$ computes the product of group values at positions greater than or equal to b, whereas $\hat{N}_1(b)$ computes the product of group values strictly greater than b. Inductively we define, for all k, such that $1\leq k\leq |T|$.

$$N_k(b) = N_{k-1}(b)\prod_{\substack{b'\in B_{t_k}\\ b'\geq b}} \left(N_{k-1}(b')\right)^{-1} N_0(b')\hat{N}_{k-1}(b')$$

$$\hat{N}_k(b) = \hat{N}_{k-1}(b)\prod_{\substack{b\in B_{t_k}\\ b'> b}} \left(N_{k-1}(b')\right)^{-1} N_0(b')\hat{N}_{k-1}(b')$$

The following lemma shows that $N_k(b)$ computes the product of group values at positions in $\cup_{j\leq k} B_{t_j}$, which are greater than or equal to b. On the other hand $\hat{N}_k(b)$

computes the product of group values at positions in $\cup_{j\leq k} B_{t_j}$ which are strictly greater than b.

LEMMA 6.7. *Let k be such that $1 \leq k \leq |T|$. Let $b \in \mathbb{N}$. Then*

$$N_k(b) = \prod_{\substack{d\geq b\\ d\in\cup_{j\leq k}B_{t_j}}} N_0(d)$$

$$\hat{N}_k(b) = \prod_{\substack{d> b\\ d\in\cup_{j\leq k}B_{t_j}}} N_0(d)$$

Proof. We prove both the equations by induction over k. The base case, that is when $k = 1$ is true by the definition of $N_1(b)$. So let us assume that the claim is true for $k-1$.

Then, we have for a $b \leq b'$.

$$N_{k-1}(b).\left(N_{k-1}(b')\right)^{-1} = \left(\prod_{\substack{d\geq b\\ d\in\cup_{j<k}B_{t_j}}} N_0(d)\right).\left(\prod_{\substack{d\geq b'\\ d\in\cup_{j<k}B_{t_j}}} N_0(d)\right)^{-1}$$

$$= \prod_{\substack{d=b\\ d\in\cup_{j<k}B_{t_j}}}^{b'-1} N_0(d) \tag{6.2}$$

For a $b < b'$ we also have that.

$$\hat{N}_{k-1}(b).\left(N_{k-1}(b')\right)^{-1} = \left(\prod_{\substack{d> b\\ d\in\cup_{j<k}B_{t_j}}} N_0(d)\right).\left(\prod_{\substack{d\geq b'\\ d\in\cup_{j<k}B_{t_j}}} N_0(d)\right)^{-1}$$

$$= \prod_{\substack{d>b\\ d\in\cup_{j<k}B_{t_j}}}^{b'-1} N_0(d) \tag{6.3}$$

We need to now prove the second equality. Let $b \leq b_0 < b_1 < \cdots < b_{x-1} < b_x$ be all positions in B_{t_k}. Writing out the product we get

$$N_k(b) = N_{k-1}(b)\left(N_{k-1}(b_0)\right)^{-1}\left(\prod_{i=0}^{x-1} N_0(b_i)\ \hat{N}_{k-1}(b_i)\left(N_{k-1}(b_{i+1})\right)^{-1}\right) N_0(b_x)\hat{N}_{k-1}(b_x)$$

Substituting the equations 6.2 and 6.3 in the above equation gives us the following formula.

$$N_k(b) = \left(\prod_{\substack{d=b\\ d\in\cup_{j\leq k}B_{t_j}}}^{b_0-1} N_0(d)\right)\left(\prod_{i=0}^{x-1} N_0(b_i)\left(\prod_{\substack{d>b_i\\ d\in\cup_{j\leq k}B_{t_j}}}^{b_{i+1}-1} N_0(d)\right)\right)\left(N_0(b_x)\hat{N}_{k-1}(b_x)\right)$$

$$= \prod_{\substack{d\geq b\\ d\in\cup_{j\leq k}B_{t_j}}} N_0(d)$$

Similarly we get that

$$\begin{aligned}
\hat{N}_k(b) &= \hat{N}_{k-1}(b)\,(N_{k-1}(b_0))^{-1}\left(\prod_{i=0}^{x-1} N_0(b_i)\ \hat{N}_{k-1}(b_i)\,(N_{k-1}(b_{i+1}))^{-1}\right) N_0(b_x)\hat{N}_{k-1}(b_x)\\
&= \left(\prod_{\substack{d>b\\ d\in\cup_{j\le k}B_{t_j}}}^{b_0-1} N_0(d)\right)\left(\prod_{i=0}^{x-1} N_0(b_i)\left(\prod_{\substack{d>b_i\\ d\in\cup_{j\le k}B_{t_j}}}^{b_{i+1}-1} N_0(d)\right)\right)\left(N_0(b_x)\hat{N}_{k-1}(b_x)\right)\\
&= \prod_{\substack{d>b\\ d\in\cup_{j\le k}B_{t_j}}} N_0(d)
\end{aligned}$$

Thus the claim is true for all k. ☐ Observe that in the above lemma $N_k(b)$ compute the product of group elements at positions greater than or equal to b in the set $\cup_{j\le k}B_{t_j}$, whereas $\hat{N}_k(b)$ compute the product of group elements at positions strictly greater than b in the set $\cup_{j\le k}B_{t_j}$.

The following Lemma shows that $N_{|T|}(1)$ gives the product of the group elements.

Lemma 6.8. *We have that $N_{|T|}(1) = \prod_{i\in B} N_0(i)$.*

Proof. This follows from Lemma 6.7. ☐

We now give active domain formulas γ^m, $m\in G$, such that γ^m is true iff $N_{|T|}(0) = m$. For this we make use of the inductive definition of N_k and show that there exists active domain formulas γ^m such that $w \models \gamma^m(b) \Leftrightarrow N_k(b) = m$. Similarly we give active domain formulas $\hat{\gamma}^m$ such that $w \models \hat{\gamma}^m(b) \Leftrightarrow \hat{N}_k(b) = m$. Observe that $N_k(b)$ is got by computing the product of $\left(\hat{N}_{k-1}(b')\right)^{-1} u(b')N_{k-1}(b')$, over b', where b' strictly increases. This requires us to traverse the elements in $B_{t_{k-1}}$ in an increasing order. The following section builds a Sorting tree to sort the elements of $B_{t_{k-1}}$ in an increasing order.

6.3. Sorting Tree. Let $t\in T$. The aim of this section is to create a data structure, which can traverse the elements in B_t in an ascending order. Note that the active domain formulas can only "access" the active domain of the word. But what we need is to access the boundary points. We know that if we assign the variables in a linear term to active domain points, then we can get hold of the boundary point. Thus we can built active domain quantifiers which can iterate through the active domain points of the word, compute the boundary point and generate a group value associated at that particular point. So the first "idea" will be to use active domain quantifiers for all variables appearing in the linear terms. But can we get the active domain points in an ordered way?

The following example will give an intution of the problem and the solution.

Example 5. *Consider two linear equations $2y_1 + y_2$ and $y_1 + 5y_2$, with the y_is being bounded variables. Thus they get assigned from the active domain of the word. Let the active domain of a word w be $D = \{5, 10, 15\}$. Then the boundary points are*

$$15, 30, 20, 55, 25, 80, 25, 35, 30, 60, 35, 85, 35, 40, 40, 65, 45, 90$$

if we assign the two variables (y_1, y_2) from D in the following order:

$$(5,5), (5,10), (5,15), (10,5), (10,10), (10,15), (15,5), (15,10), (15,15)$$

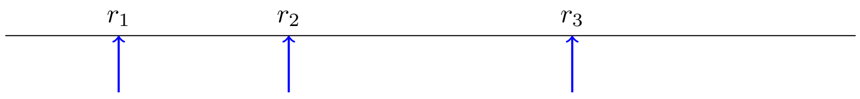

FIG. 5. *The points marked are the active domain points*

Observe that if we follow the particular ordering, then the boundary points, BP is not generated in the order we would want it to be (since it is not in ascending order). We also see that there is repetition of occurrences of boundary points. Thus if we generate the group values in this ordering and compute its product, we would not get the required product. Since we might be looking at non-commutative groups, we need to ensure that BP is generated in an ascending order.

Here is the solution we propose. We rename the variables and generate the following linear terms:

$$\rho_1 = 2x_1 + x_2, \rho_2 = x_1 + 2x_2, \rho_3 = x_1 + 5x_2, \rho_4 = 5x_1 + x_2$$

The first and third got by replacing y_1 by x_1 and y_2 by x_2. The second and fourth are got by replacing y_1 by x_2 and y_2 by x_1. Now we make use of the fact that we have the leeway to choose $\mathcal{D}$, the active domain. We will assume that the active domain points we choose are very far from each other. Thus, for any assignment to the x_is such that $x_1 >> x_2$ we have

$$5x_1 + x_2 > 2x_1 + x_2 > x_1 + 5x_2 > x_1 + 2x_2$$

Thus we have an ordering for any fixed assignment of the x_is. Now we fix an ordering among different assignments of x_is. Consider the figure 5, where the active domain points are marked r_1, r_2, r_3. Let us look at one of the linear terms, say $\rho_1(x_1, x_2)$. Then we have

$$\rho_1(r_3, r_3) > \rho_1(r_3, r_2) > \rho_1(r_3, r_1) > \rho_1(r_2, r_2) > \rho_1(r_2, r_1) > \rho_1(r_1, r_1)$$

since $r_3 >> r_2 >> r_1$. Note that $x_1 \geq x_2$ always. That is we do not consider the points generated by $x_1 < x_2$. But observe that since $\rho_1(x_1, x_2) = \rho_2(x_2, x_1)$ we will be looking at all the boundary points, even if we restrict our variables to this ordering. We generalize the idea in the above example. In general we need to worry about not two, but a fixed number of linear terms. Also the linear terms might have negated variables.

To take care of this general situation, we define a tree called *sorting tree*, $\mathcal{T}_t$ which corresponds to the set B_t. The tree satisfies the following property: If the leaves of the tree are enumerated from left to right, then we get the set B_t in ascending order, provided the active domain for the word is choosen judiciously. We will first give the construction of the tree and then the active domain R_ϕ.

Sorting Tree: A node in $\mathcal{T}_t$ is labeled by a tuple (f, A), where $f(x_1, \ldots, x_l)$ is a function in F_t, A an assignment for the variables in f such that $A(x_1) > A(x_2) > \cdots > A(x_l)$ and $\forall i \leq l : A(x_i) \in nnp(w)$.

We show how to inductively built the tree. The root is labeled by the tuple $(t, \{\})$, where t is the function which depends only on $\vec{y}$ (and hence constant on $\vec{x}$) and $\{\}$ is the empty assignment. The root is not marked a leaf node. Consider the internal node $(f(x_1, \ldots, x_l), A)$. It will have three kinds of children ordered from left to right as follows.

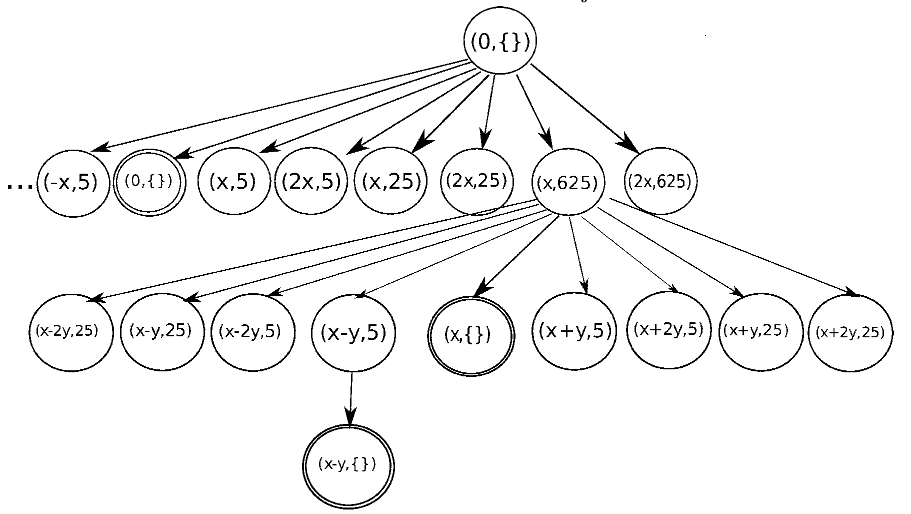

FIG. 6. *Sorting Tree: The double circles represent leaves of the tree. The nodes of the tree are labelled (f, A), where A is an assignment for the function f and $t = 0$. For better presentation we only show the assignment to the newly introduced variable in a node. For example, the tuple $(x - 2y, 25)$ assigns $x = 625$ and $y = 25$. The assignment to x is given in the node's parent.*

1. Left children: These are labeled by tuples of the form (f'_α, A'_j) where

$$f'_\alpha(x_1, \dots, x_{l+1}) = f(x_1, \dots, x_l) + \alpha x_{l+1} \text{ and } -\Delta \le \alpha < 0, -\alpha \in \mathbb{N}$$

$$A'_j = A \cup [x_{l+1} \mapsto j], \text{ where } j < A(x_l) \text{ and } j \in \text{nnp}(w)$$

The left children are now ordered as follows. The tuple $(f'_{\alpha_1}, A'_{j_1})$ is on the left of $(f'_{\alpha_2}, A'_{j_2})$ if $j_1 > j_2$ or if $j_1 = j_2$ and $\alpha_1 < \alpha_2$.
2. Middle child: It is labeled by the tuple (f'', A) where $f''(x_1, \dots, x_l) = f(x_1, \dots, x_l)$. It is marked a **leaf** node.
3. Right children: These are labeled by tuples of the form (f'_α, A'_j) where

$$f'_\alpha(x_1, \dots, x_{l+1}) = f(x_1, \dots, x_l) + \alpha x_{l+1} \text{ and } 0 < \alpha \le \Delta, \alpha \in \mathbb{N}$$

$$A'_j = A \cup [x_{l+1} \mapsto j], \text{ where } j < A(x_l) \text{ and } j \in \text{nnp}(w)$$

The children are now ordered as follows. The tuple $(f'_{\alpha_1}, A'_{j_1})$ is on the left of $(f'_{\alpha_2}, A'_{j_2})$ if $j_1 < j_2$ or $j_1 = j_2$ and $\alpha_1 < \alpha_2$.

Observe that if there is no j such that $j < A(x_l)$ and $j \in \text{nnp}(w)$, then (f, A) will only have the single child (f'', A). The tree is built until all functions with s variables appear in leaves and hence the depth of the tree is $s + 2$.
Active domain: The infinite set $R_\phi \subseteq R_{\phi_K}$ satisfies the following property: Any two points $a < b$ in R_ϕ is such that $b \ge a(4s\Delta)$. The idea is to ensure that the active domain points are "far" enough to satisfy the nice properties we will soon see. Note that there always exists an infinite set with the above property. We pick any set which satisfies this property and call it R_ϕ. Such an active domain will ensure that the values of the leaves of the tree increases from left to right.

Let us look at the following example. Figure 6 shows part of a tree, where $\Delta = 2$, $t = 0$ and $\text{nnp}(w) = \{5, 25, 625\}$. Note that the values of the leaves of tree is in ascending order.

Let $\mathcal{R} = 4s\Delta$. Let us also assume that $nnp(w) \subseteq R_\phi$. Given a node (f, A), we say the *value* of the node is the function f evaluated under the assignment of A (denoted by $f(A)$).

LEMMA 6.9. *Let N be an internal node labeled by a function $f(x_1, \ldots, x_l)$ with $l < s$ and an assignment A. If $A(x_l) = n$, then the children of this node have values in the range $[f(A) - \frac{\Delta}{\mathcal{R}}n, f(A) + \frac{\Delta}{\mathcal{R}}n]$. Moreover the values of the children increases from left to right.*

Proof. The range is given by the construction. Let us look at the case when both α_1 and α_2 are negative. The other cases are similar to this case or are trivial. There are two cases to consider now. If $j_1 > j_2$, then $\mathcal{R}.j_2 \leq j_1$. Therefore $|\alpha_2|.j_2 \leq j_1$ and since both the α_is are negative we get $\alpha_2.j_2 > \alpha_1.j_1$, which shows that the value of the children increases from left to right. In the other case, we have that $\alpha_1 < \alpha_2$ and $j_1 = j_2$. Then it is obvious that $\alpha_1.j_1 < \alpha_2.j_2$ and therefore the claim is true. □

Next we show that for any two neighboring nodes in the tree, the values in the leaves of the subtree rooted at the left node is less than the values in the leaves of the subtree rooted at the right node. Let $V_{(f,A)}$ denote the set of values in the leaves of the subtree rooted at (f, A).

LEMMA 6.10. *Let (f, A) and (f', A') be neighboring nodes of the same parent such that (f, A) is to the left of (f', A'). Then $u < v$ for every $u \in V_{(f,A)}$ and $v \in V_{(f',A')}$.*

Proof. Let $f = \sum_{i=1}^{l-1} \alpha_i x_i + \alpha_l x_l + t$ and $f' = \sum_{i=1}^{l-1} \alpha_i x_i + \alpha'_l x_l + t$. We show that the rightmost element, u in $V_{(f,A)}$ is less than the left most element, v in $V_{(f',A')}$. From Lemma 6.9 and applying induction on the depth of the tree, one can show that $u \leq f(A) + (s - l)\frac{\Delta}{\mathcal{R}}n$ and $v \geq f'(A') - (s - l)\frac{\Delta}{\mathcal{R}}n'$. Here n, n' are the minimum assignments in A and A' respectively. Let us assume that both coefficients $\alpha'_l, \alpha_l > 0$. A similar analysis can be given for other combinations of α'_l and α_l. Now since (f, A) is the left neighbor of (f', A') we have $n < n'$. Then $v - u \geq \alpha'_l n' - (s - l)\Delta\frac{n'}{\mathcal{R}} - \alpha_l n - (s - l)\Delta\frac{n'}{\mathcal{R}} \geq n' - 3s\Delta\frac{n'}{\mathcal{R}} > 0$. The claim follows, since $\mathcal{R} = 4s\Delta$. □ The next lemma says that the values of the leaves of the tree increases as we traverse from left to right.

LEMMA 6.11. *Let (f, A) and (f', A') be two distinct nodes such that $f(A) < f'(A')$. Then (f, A) appear to the left of (f', A').*

Proof. This follows from Lemma 6.10. □

Lemma 6.11 shows that if we travel the tree from left to right, then we get the elements in B_t in an ascending order. The following lemma now shows how to build a formula, which can use the sorting tree to traverse through the elements in B_t.

LEMMA 6.12 (Tree Lemma). *Let $nnp(w) \subseteq R_\phi$. Fix $t \in T$. Assume that for every $b \in B_t$ we have an element $g_b \in G$. For all $f \in F_t$, $m \in G$, $\vec{d} \in nnp(w)^t$, $\vec{a}$, let γ_f^m be active domain formulas such that $w, \vec{d}, \vec{a} \models \gamma_f^m(\vec{x}, \vec{y})$ iff $g_{f(\vec{d},\vec{a})} = m$. Then there are active domain formulas $\Gamma^{m'}$ such that $w, \vec{a} \models \Gamma^{m'}(\vec{y})$ iff $\prod_{b \in B_t} g_b = m'$.*

Proof. We will use the sorting tree, $\mathcal{T}_t$ corresponding to B_t for the construction of our formula. Recall that the nodes are labeled by tuples (f, A), where f is a function and A is the assignment of the parameters of f. Let $V_{(f,A)} \subseteq B_t$ be the set of values at the leaves of the subtree rooted at the node labeled by (f, A), and $g_{(f,A)} = \prod_{b \in V_{(f,A)}} g_b$. We will do induction on the depth D of the tree. Let $\tau_f^{m,D}(\vec{x})$ be a formula such that $w, \vec{d} \models \tau_f^{m,D}(\vec{x})$ iff $\prod_{b \in V_{(f,\vec{d})}} g_b = m$ where $(f, \vec{d})$ is the label of a node that has a subtree of depth at most D. Hence we multiply all group elements g_b for which b is in $V_{(f,\vec{d})}$.

Base Case (leaves): We define $\tau_f^{m,0}(\vec{x}) = \gamma_f^m(\vec{x})$.

Induction Step: Let us assume that the claim is true for all nodes with a subtree of depth at most D. Let the node labeled by (f, A) have a subtree of depth $D+1$. We will need to specify the formula $\tau_f^{m,D+1}(\vec{x})$, where $\vec{x}$ agrees with the assignment A. For every child (f', A') of (f, A) the depth of the corresponding subtree is less than or equal to D. Hence we know we have already formulas by induction.

Recall what the children of (f, A) are: They are of form (f'_α, A'_j) and (f'', A_j). Moreover all nodes (f'_α, A'_j), where α is negative, come to the left of (f'', A_j) and all nodes (f'_α, A'_j), where α is positive, come to its right.

We start by grouping some of the children and computing their product. We let $T^-(A'_j)$ be the product of all subtrees labeled by (f'_α, A'_j) for $\alpha = -\Delta, -\Delta+1, \ldots, -1$. This is a finite product so we can compute this by a Boolean combination of the formulas $\tau_{f'_\alpha}^{m,D}(\vec{x}, x_{l+1})$.

$$\pi_f^{-,m,D}(\vec{x}, x_{l+1}) ::= \bigvee_{m_{-\Delta}\ldots m_{-1}=m} \left(\bigwedge_{\alpha=-\Delta}^{-1} \tau_{f'_\alpha}^{m_\alpha,D}(\vec{x}, x_{l+1}) \right)$$

Now we want to compute the product $\left(\prod_{j\in \mathrm{nnp}(w)} (T^-(A'_j))^{-1}\right)^{-1}$ which is the product of the $T^-(A'_j)$ where $j \in \mathrm{nnp}(w)$ is decreasing. But this can be computed using an active domain group quantifier, $\tau_f^{-,m,D}(\vec{x})$ as follows:

$$\tau_f^{-,m,D}(\vec{x}) = Q_G^{m^{-1}} x_{l+1} \left(\neg\lambda(x_{l+1}) \wedge (x_l > x_{l+1})\right)$$

$$\langle \pi_f^{-,m_1^{-1},D}(\vec{x}, x_{l+1}), \ldots, \pi_f^{-,m_K^{-1},D}(\vec{x}, x_{l+1})\rangle$$

Recall that the elements of group G are ordered $m_1, \ldots, m_K$. For the single node (f'', A) we already have the formulas $\tau_{f''}^{m,D}(\vec{x})$ by induction (here we have $\vec{x}$ since the assignment A is the same for (f, A) and (f'', A)).

Similarly we define formulas $\pi_f^{+,m,D}(\vec{x}, x_{l+1})$ for the positive coefficients, and compute their product $\prod_{j\in \mathrm{nnp}(w)} T^+(A'_j)$ in an increasing order.

$$\pi_f^{+,m,D}(\vec{x}, x_{l+1}) ::= \bigvee_{m_1\ldots m_\Delta=m} \left(\bigwedge_{\alpha=1}^{\Delta} \tau_{f'_\alpha}^{m_\alpha,D}(\vec{x}, x_{l+1}) \right)$$

$$\tau_f^{+,m,D}(\vec{x}) ::= Q_G^m x_{l+1}\left(\neg\lambda(x_{l+1}) \wedge (x_l > x_{l+1})\right)$$

$$\langle \pi_f^{+,m_1,D}(\vec{x}, x_{l+1}), \ldots, \pi_f^{+,m_K,D}(\vec{x}, x_{l+1})\rangle$$

We have now computed the product of the group elements for the three different groups of children. So by a Boolean combination over these formulas we get $\tau_f^{m,D+1}(\vec{x})$:

$$\bigvee_{m'm''m'''=m} \left(\tau_f^{-,m',D}(\vec{x}) \wedge \tau_{f''}^{m'',D}(\vec{x}) \wedge \tau_f^{+,m''',D}(\vec{x})\right)$$

So finally we get $\Gamma^{m'}$ which is same as the formula $\tau^{m',s+2}_{(0,\{\})}$, which is valid at the root of the tree. □

We can also relativize the formulas $\Gamma^{m'}$, with respect to any position.

Lemma 6.13. *Let the hypothesis of Lemma 6.12 hold. Then for all $m \in G$, there are active domain formulas $\Gamma^m(z,\vec{y})$ such that*

$$w, d, \vec{a} \models \Gamma^m(z,\vec{y}) \text{ iff } \prod_{\substack{b \in B_t \\ b \geq d}} g_b = m$$

Similarly, for all $m \in G$, there are active domain formuals $\Gamma^m(z,\vec{y})$ such that

$$w, d, \vec{a} \models \Gamma^m(z,\vec{y}) \text{ iff } \prod_{\substack{b \in B_t \\ b > d}} g_b = m$$

Proof. We will show how to do the first part of the Lemma. We can conjunct all formulas γ_f^m, for $m \neq 1_G$, with the condition $z \geq f(\vec{x},\vec{y})$. Whereas the formula γ_f^1, can be disjuncted with the formula $z \geq f(\vec{x},\vec{y})$. Now applying the tree Lemma 6.12 will give us the formula we were looking for. □

6.4. Constructing the active domain formula. This subsection uses all the previous subsections to build the active domain formula we are looking for.

First for all $m \in G, f \in F$, we give formulas γ_f^m such that it is true when N_0 evaluates to m.

Lemma 6.14. *For all $m \in G, f \in F$, there are formulas γ_f^m such that for all $d_1, \ldots, d_s \in nnp(w)$, we have that*

$$w, \vec{d}, \vec{a} \models \gamma_f^m(\vec{x},\vec{y}) \Leftrightarrow N_0(f(\vec{d},\vec{a})) = m$$

Proof. For an $i \leq K$, we denote by $\tilde{\phi}_{m_i}(z,\vec{y})$ the formula $\bigwedge_{j<i} \neg\phi_j(z,\vec{y}) \wedge \phi_i(z,\vec{y})$. For a $l \in \mathbb{N}$, the following formula checks if there is a point b' in B such that $b+l = b'$. Since in each B there is at most one such element, we can use the group quantifier to simulate the existential quantifier.

$$\delta_f^l(\vec{x},\vec{y}) ::= \bigvee_{f' \in F \setminus f} Q_G^{m_1} \vec{x}' \langle f'(\vec{x}',\vec{y}) = f(\vec{x},\vec{y}) + l, false, \ldots, false \rangle$$

So we have that $\mathrm{IR}(b) = l$ iff $\delta_f^{l+1} \wedge \bigwedge_{l'<l} \neg\delta_f^l$ is true. We define $\pi_f^{m,+l}$ to be true if the product of the first l group elements to the right is m.

$$\pi_f^{m,+l}(\vec{x},\vec{y}) ::= \bigvee_{g_0 \cdots g_l = m} \left(\bigwedge_{i=0}^{l} \tilde{\phi}_{g_i}(f(\vec{x},\vec{y}) + i, \vec{y}) \right)$$

We now give a formula $\psi_{f,m}^{Post}(\vec{x},\vec{y})$ such that $Post(f(\vec{d},\vec{a})) = m$ iff $w \models \psi_{f,m}^{Post}(\vec{d},\vec{a})$. Now we have two cases to consider.

Case $\mathrm{IR}(b) < W$**:** For each of the case $b < b'$ such that $l = b' - b < W$, the formula $\pi_f^{m,l}$ compute the product of the group elements. We define ψ_f^m to be true if the interval is less than W and the product of the group elements is equal to m.

$$\psi_f^m(\vec{x},\vec{y}) ::= \bigwedge_{l=1}^{W-1} \left(\left(\delta_f^l(\vec{x},\vec{y}) \wedge \bigwedge_{l'=1}^{l-1} \neg\delta_f^{l'}(\vec{x},\vec{y})\right) \rightarrow \pi_f^{m,+l}(\vec{x},\vec{y}) \right)$$

Case IR$(b) \geq W$**:** When $b' - b \geq W$ we have to compute the product for the first r group elements, where $b + r \equiv_W 0$ and $r \leq W$. Therefore we define $\hat{\psi}_f^m$ which computes the product of the group elements equal to m in this case.

$$\hat{\psi}_f^m(\vec{x}, \vec{y}) ::= \bigwedge_{l=1}^{W} \big(f(\vec{x}, \vec{y}) + r \equiv_W 0\big) \to \pi_f^{m,+r}(\vec{x}, \vec{y})$$

A Boolean combination over δ_f^l, $\hat{\psi}_f^m$ and ψ_f^m can give us the formula $\psi_{f,m}^{Post}$.

$$\psi_{f,m}^{Post}(\vec{x}, \vec{y}) ::= \left(\left(\bigwedge_{l=1}^{W-1} \neg\delta_f^l\right) \implies \hat{\psi}_f^m\right) \vee \psi_f^m$$

Similarly we give formulas $\psi_{f,m}^{Pre}$ for all $m \in G$ and $f \in F$, such that $Pre(f(\vec{d}, \vec{a})) = m$ iff $w \models \psi_{f,m}^{Pre}(\vec{d}, \vec{a})$.

For all $m \in G$ and $f \in F$, the formulas, $\gamma_f^m(\vec{x}, \vec{y})$ are as follows:

$$\gamma_f^m(\vec{x}, \vec{y}) ::= \bigvee_{g_1 g_2 g_3 = m} \psi_{f,g_1}^{Pre} \wedge \tilde{\phi}_{g_2} \wedge \psi_{f,g_3}^{Post}$$

□

We know that for every $b \in B$ there is a function $f \in F$, $d_1, \ldots, d_{s'} \in \text{nnp}(w)$, such that $b = f(\vec{d}, \vec{a})$, where $\vec{a}$ is the fixed assignment to the variables $\vec{y}$. We will use this encoding of a position and define a formula $\nu_{k,f}^m$ such that

$$w, \vec{d}, \vec{a} \models \nu_{k,f}^m(\vec{x}, \vec{y}) \Leftrightarrow N_k(f(\vec{d}, \vec{a})) = m$$

Similarly we define formulas $\hat{\nu}_{k,f}^m$ such that $w, \vec{d}, \vec{a} \models \hat{\nu}_{k,f}^m(\vec{x}, \vec{y})$ iff $\hat{N}_k(f(\vec{d}, \vec{a})) = m$. We show this by induction over $k \leq |T|$. Starting with the base case $k = 1$.

Lemma 6.15. *Let $d_1, \ldots, d_{s'} \in nnp(w)$. For each $m \in G$, there is an active domain formula $\nu_{1,f}^m(\vec{x}, \vec{y})$ in $\mathcal{L}[<, +]$, such that if $w \models \nu_{1,f}^m(\vec{d}, \vec{a})$ then $N_1(f(\vec{d}, \vec{a})) = m$.*
Similarly there is an active domain formula $\hat{\nu}_{1,f}^m(\vec{x}, \vec{y})$ in $\mathcal{L}[<, +]$ such that if $w, \vec{d} \models \hat{\nu}_{1,f}^m(\vec{x}, \vec{a})$ then $\hat{N}_1(f(\vec{d}, \vec{a})) = m$.

Proof. Consider the set B_{t_1} of boundary points formed using functions F_{t_1}. Look at all the formulas γ_f^m, where $f \in F_{t_1}$, given by Lemma 6.14. Now we are in a position to apply Lemma 6.13, which gives us the required formulas. □

The induction step follows.

Lemma 6.16. *Let $d_1, \ldots, d_{s'} \in nnp(w)$. For each $m \in G$, there is an active domain formula $\nu_{k,f}^m$ in $\mathcal{L}[<, +]$, such that $w, \vec{d}, \vec{a} \models \nu_{k,f}^m(\vec{x}, \vec{y})$ then $N_k(f(\vec{d}, \vec{a})) = m$.*
Similarly there is an active domain formula $\hat{\nu}_{k,f}^m(\vec{x}, \vec{y})$ in $\mathcal{L}[<, +]$, such that $w, \vec{d}, \vec{a} \models \hat{\nu}_{k,f}^m(\vec{x}, \vec{y})$ then $\hat{N}_k(f(\vec{d}, \vec{a})) = m$.

Proof. For all $m \in G$ and $f' \in F_{t_k}$ we give formulas $\gamma_{f'}^m$ such that the following holds:

$$w, \vec{d'}, \vec{a} \models \gamma_{f'}^m(\vec{x}, \vec{y}) \Leftrightarrow \left(\hat{N}_{k-1}(b')\right)^{-1} N_0(b')\hat{N}_{k-1}(b') = m$$

Here $d'_1, \ldots, d'_s \in nnp(w)$ and $f'(\vec{d'}, \vec{a}) = b'$. By induction hypothesis there exists formulas $\nu_{k-1,f'}^m$ and $\hat{\nu}_{k-1,f'}^m$ which corresponds to $N_{k-1}(f'(\vec{x}, \vec{y}))$ and $\hat{N}_{k-1}(f'(\vec{x}, \vec{y}))$

respectively. Taking a Boolean combination over these formulas we get the required formula $\gamma^m_{f'}$. We now apply our Tree Lemma 6.13 which gives us formulas Γ^m, for all $m \in G$, such that

$$w, \vec{d}, \vec{a} \models \Gamma^m(f(\vec{x}, \vec{y}), \vec{y}) \Leftrightarrow w \models \prod_{\substack{b' \in B_{t_k} \\ b' > b}} \left(\hat{N}_{k-1}(b')\right)^{-1} u(b') N_{k-1}(b') = m$$

Taking Boolean combination over Γ^m and $\nu^m_{k-1,f}$ will give us the formula $\nu^m_{k,f}$. Similarly we can build active domain formulas $\hat{\nu}^m_{k,f}(\vec{x}, \vec{y})$, for all $m \in G$. □

The formulas we were looking for are the active domain formulas $\nu^m_{|T|,f}$, where f is the function which outputs the constant 1. The following Lemma now follows.

Lemma 6.17. *Let $nnp(w) \subseteq R_\phi$. Then*

$$\prod_{b \in B} N_0(b) = m \Leftrightarrow w, \vec{a} \models \nu^m_{|T|,1}(\vec{y})$$

Proof. From Lemma 6.16 it follows that $w, \vec{a} \models \nu^m_{|T|,1} \Leftrightarrow N_k(1) = m$. The claim now follows from Lemma 6.8. □

Now we are in a position to finish the proof of our main Lemma.

Proof. [Proof of Lemma 5.4] By Lemma 6.6 we know that it suffices to compute the product of $N_0(b)$, provided the infinite interval evaluate to identity. From Lemma 6.17 we know that there are active domain formulas $\nu^m_{|T|,1} \in \mathcal{L}[<, +]$ such that $N_{|T|}(1) = m$ iff $w, \vec{a} \models \nu^m_{|T|,1}$, provided the active domain of the word w comes from the set R_ϕ.

We need to do one last thing. Check that the infinite interval evaluates to 1_G. Replace all formulas $z > \rho$, $z < \rho$, $c(z)$ for a $c \neq \lambda$ and $\lambda(z)$ by $true$, $false$, $false$, $true$ respectively in the formulas $\hat{\phi}_i$ and call these formulas $\hat{\psi}_i$. There exists a witness in the infinite interval for the formula $\hat{\phi}_i$ iff $\hat{\psi}_i$ evaluates to true. By Theorem 6.4 there should not be any witness in the infinite interval. Therefore the formula $\hat{\psi} = \bigvee_i \psi_i$ evaluates to true iff the infinite interval does not evaluate to 1_G.

Combining both the formulas, we get that $\phi'(\vec{y}) = \neg\hat{\psi}(\vec{y}) \wedge \nu^m_{|T|,1}(\vec{y})$ and therefore for all w, such that $nnp(w) \subseteq R_\phi$ and for all $\vec{a} \in \mathbb{N}^r$ we have that

$$w \models \phi(\vec{a}) \Leftrightarrow w \models \phi'(\vec{a})$$

This completes the proof. □

7. Conclusion. We have shown that in the presence of a neutral letter the addition relation collapse to linear ordering no matter what monoid quantifier is been used. All languages definable using monoid quantifiers and an order predicate, on the other hand, are regular [BIS90]. Now using semigroup theoretic methods we can separate these classes [Str94]. This enabled us to show separation between various logics which uses addition and order predicates.

Unfortunately if both addition and multiplication are present, then the collapse does not happen. It is also interesting to note that non-solvable groups do not show any surprising property if only addition is present, but as we know from Barrington's theorem non-solvable groups behave quite differently when both addition and multiplication are present.

Figure 7 compare the expressiveness of different logics in the presence of addition.

The ultimate objective is to show non-expressibility results for arbitrary predicates or at least when both addition and multiplication are present. One possible

$$\begin{tikzcd}
 & {\mathrm{GRP}[<,+]} \arrow[dl] & {S_{10}[<,+]} \arrow[l] & {S_5[<,+]} \arrow[l] & \\
{\mathrm{FOGRP}[<,+]} & & & & \\
 & & {\mathrm{MOD}[<,+]} \arrow[uul] \arrow[dl] & {C_6[<,+]} \arrow[l] \arrow[uu] & {C_2[<,+]} \arrow[l] \\
 & {\mathrm{FOMOD}[<,+]} \arrow[uul] & & & \\
 & & {\mathrm{FO}[<,+]} \arrow[ul] & &
\end{tikzcd}$$

FIG. 7. *Relation between logics with regular quantifiers and addition relation. The arrows show strict inclusion. No arrow show that the classes are incomparable.* C_6 *and* C_2 *denote cyclic groups with* 6 *and* 2 *elements respectively and* S_{10} *and* S_5 *denote symmetric groups with* 10 *and* 5 *elements respectively. The diagram would be similar if the only predicate was* $<$. *On the other hand, if multiplication is present, the logics behave differently and many questions remain open.*

direction to take this work forward will be to identify the kinds of predicates where our techniques could be applied.

Another way to look at separating the "natural uniform" versions of the complexity classes will be to ask whether one can come up with other suitable restrictions on the set of languages. We saw how restricting the language class to neutral letter languages, help us come up with lower bound results. Can one come up with better restrictions on the set of languages? Inside this restricted set of languages can one show addition and multiplication collapse to order relation? This seems to be the idea Straubing considers in [Str05]. Straubing [Str94] proposes word problems over Regular language as a suitable restriction, while McKenzie, Thomas, Vollmer [MTV10] consider context free languages as a restriction.

Another interesting question which our result fails to answer is whether word problems over non-solvable groups can be defined in $\mathrm{MAJ}[<,+]$ [KLR07]? Can our techniques be of use when working with infinite groups?

REFERENCES

[ABO84] Miklos Ajtai and Michael Ben-Or. A theorem on probabilistic constant depth Computations. In ACM, editor, *Proceedings of the sixteenth annual ACM Symposium on Theory of Computing, Washington, DC, April 30–May 2, 1984*, pages 471–474, pub-ACM:adr, 1984. ACM Press.

[Bar89] David A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38(1):150–164, February 1989.

[BIL+05] David A. Mix Barrington, Neil Immerman, Clemens Lautemann, Nicole Schweikardt, and Denis Thérien. First-order expressibility of languages with neutral letters or: The Crane Beach conjecture. *J. Comput. Syst. Sci.*, 70(2):101–127, 2005.

[BIS90] David A. Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41(3):274–306, December 1990.

[BL00a] Michael Benedikt and Leonid Libkin. Expressive power: The finite case. In *Constraint Databases*, pages 55–87, 2000.

[BL00b] Michael Benedikt and Leonid Libkin. Relational queries over interpreted structures. *J. ACM*, 47(4):644–680, 2000.

[BL06] Christoph Behle and Klaus-Jörn Lange. FO[<]-uniformity. In *IEEE Conference on Computational Complexity*, pages 183–189, 2006.

[BS91] David A. Mix Barrington and Howard Straubing. Superlinear lower bounds for bounded-width branching programs. In *Structure in Complexity Theory Conference*, pages 305–313, 1991.

[CKK$^+$07] Arkadev Chattopadhyay, Andreas Krebs, Michal Koucký, Mario Szegedy, Pascal Tesson, and Denis Thérien. Languages with bounded multiparty communication complexity. In *STACS*, pages 500–511, 2007.

[End72] Herbert B. Enderton. *A Mathematical Introduction to Logic.* Academic Press, New York, 1972.

[FSS84] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

[Imm87a] Neil Immerman. Expressibility as a complexity measure: Results and directions. In Stephen R. Mahaney, editor, *Proceedings of the 2nd Annual Conference on Structure in Complexity Theory, CSCT'87 (Cornell University, Ithaca, NY, June 16-19, 1987)*, pages 194–202, Washington, D.C., 1987. IEEE Computer Society, Computer Society Press of the IEEE.

[Imm87b] Neil Immerman. Languages that capture complexity classes. *SIAM Journal on Computing*, 16(4):760–778, August 1987.

[Imm99] Neil Immerman. *Descriptive complexity.* Graduate texts in computer science. Springer, 1999.

[Juk12] Stasys Jukna. *Boolean Function Complexity*, volume 27 of *Algorithms and Combinatorics.* Springer-Verlag, New York, 2012.

[KLR07] Andreas Krebs, Klaus-Jörn Lange, and Stephanie Reifferscheid. Characterizing TC^0 in terms of infinite groups. *Theory Comput. Syst.*, 40(4):303–325, 2007.

[KPT05] Michal Koucký, Pavel Pudlák, and Denis Thérien. Bounded-depth circuits: separating wires from gates. In *STOC*, pages 257–265, 2005.

[Lib04] Leonid Libkin. *Elements of Finite Model Theory.* Springer-Verlag, Berlin, 2004.

[Lin66] Per Lindström. First order predicate logic with generalized quantifiers. *Theoria*, 32:186–195, 1966.

[LTT06] Clemens Lautemann, Pascal Tesson, and Denis Thérien. An algebraic point of view on the crane beach property. In *CSL*, pages 426–440, 2006.

[MTV10] Pierre McKenzie, Michael Thomas, and Heribert Vollmer. Extensional uniformity for boolean circuits. *SIAM Journal on Computing*, 39(7):3186–3206, 2010.

[Pre29] Mojżesz Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. *Comptes Rendus, I. Congrès de Mathématiciens des pays slaves, Warsaw*, pages 192–201, 1929.

[Raz89] A. A. Razborov. On the method of approximations. In ACM, editor, *Proceedings of the twenty-first annual ACM Symposium on Theory of Computing, Seattle, Washington, May 15–17, 1989*, pages 167–176, pub-ACM:adr, 1989. ACM Press.

[RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, August 1997.

[RS07] Amitabha Roy and Howard Straubing. Definability of languages by generalized first-order formulas over $(N, +)$. *SIAM J. Comput*, 37(2):502–521, 2007.

[Smo87] R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In ACM, editor, *Proceedings of the nineteenth annual ACM Symposium on Theory of Computing, New York City, May 25–27, 1987*, pages 77–82, pub-ACM:adr, 1987. ACM Press.

[Str94] Howard Straubing. *Finite automata, formal logic, and circuit complexity.* Birkhauser Verlag, Basel, Switzerland, 1994.

[Str05] Howard Straubing. Inexpressibility results for regular languages in nonregular settings. In *Developments in Language Theory*, pages 69–77, 2005.

[STT95] Howard Straubing, Denis Thérien, and Wolfgang Thomas. Regular languages defined with generalized quanifiers. *Inf. Comput*, 118(2):289–301, May 1995.

[Vol99] Heribert Vollmer. *Introduction to circuit complexity.* Springer-Verlag, Berlin-Heidelberg-New York-Barcelona-Hong Kong-London-Milan-Paris-Singapur-Tokyo, 1999.