EXPRESSIBILITY OF QUANTIFIERS OVER SMALL STRUCTURES

GAVIN BELL, JEREMY LEE

FACULTY MENTOR: PATRICK LUTZ

AUGUST 14, 2026

Abstract. In this paper, we present an improvement to Walkoe's Linear Prefix Theorem [Wal70]. We build on the work of [KW73] and [Ros05] to show that the theorem holds over models whose sizes grow linearly in the length of the quantifier strings we analyze. Further, we show that certain quantifier strings of arbitrary length can be distinguished by sentences in prenex normal form over finite models.

1. Introduction

In mathematical logic, a common problem is to study what properties can be expressed by certain types of sentences. For sentences in first-order logic, one such expressibility result is the Linear Prefix Theorem, proven by Walkoe [Wal70] in 1970.

Theorem 1.1 (Linear Prefix Theorem, [Wal70])**.** *Let p and q be different quantifier prefixes of the same length n, there exists a first-order sentence beginning in p that is not equivalent to any first-order sentence beginning in q.*

For example, this result shows that there is some sentence beginning with $\forall\exists\forall$ that is not equivalent to any sentence beginning with $\forall\forall\exists$. Intuitively, this theorem proves that every sequence of quantifiers can express some property that a different sequence of the same length cannot express.

Walkoe's original proof of this theorem utilized uncountably large models, obtained from infinite Ramsey theory. In 1973, Keisler and Walkoe [KW73] gave a proof using only finite models through Ramsey theory, though the sizes of these models were superexponential in n, the length of the quantifier prefixes.

In 1996, Grädel and McColm [GM96] conjectured that the Linear Prefix Theorem held over structures with a single binary relation (i.e. graphs). In 2005, Rosen [Ros05] proved this conjecture and strengthened Walkoe's original result in several ways. One strengthening showed that the Linear Prefix Theorem holds as long as p does not embed into q, using countably infinite structures obtained through Fraïssé limits. However, it remains an open question whether this theorem can be proved over finite graphs, though further progress was made by He [He14] in 2014.

The methods used to prove the Linear Prefix Theorem in the past inherently produce structures that are either infinite or very large compared to n, the length of the quantifier prefixes in question. This motivates exploring exploring whether this result can be proven to hold over smaller models. In this paper, we prove the Linear Prefix Theorem using models of size $\leq 3n$. We also demonstrate a special case of Rosen's result for quantifiers prefixes that do not embed into each other using finite structures. For both of these results, our techniques used a modified version of Ehrenfeucht–Fraïssé games, similar to those used by both Rosen [Ros05] and He [He14].

2. Preliminaries

For the purposes of this paper, we assume a basic familiarity with first-order logic and model theory.

Let $p = P_1P_2 \dots P_n$ be a quantifier string of length n.

Notation 2.1. *If φ is a quantifier-free first order sentence, we define*

$$p\varphi := P_1x_1P_2x_2 \dots P_nx_n\varphi.$$

The following definitions will be recurrent throughout the rest of this paper.

Definition 2.2. A first-order sentence φ is said to be in **prenex normal form** φ is of the form $p\psi$, where p is a quantifier string and ψ is quantifier-free.

That is to say, a sentence in prenex normal form is one in which all quantifiers occur at the very start of the sentence. The statement of the Linear Prefix Theorem that we prove will require distinguishing between sentences in prenex normal form further:

Definition 2.3. A first order sentence φ in prenex normal form is called a **p-sentence** if it begins with a quantifier string matching p, followed by a quantifier-free formula.

With the above definitions, we can rephrase the Linear Prefix Theorem as follows.

Theorem 2.4 (Linear Prefix Theorem)**.** *Let p and q be distinct quantifier strings of length n. Then, there exists a p-sentence which is not equivalent to any q-sentence.*

For our proof of the Linear Prefix Theorem, we use a slightly modified version of Ehrenfeucht–Fraïssé games, which allows us to identify which kinds of sentences cause the first-order differences between structures detected by the game. A similar sort of game is employed in [Ros05]. We give the definition of an Ehrenfeucht–Fraïssé game below. Throughout the rest of this paper, we will refer to Ehrenfeucht–Fraïssé games as EF games.

Definition 2.5. An n-round **EF game** consists of two players: Spoiler and Duplicator, along with two first-order structures A and B in the same language. At each round, the two players play in the following order:

(1) Spoiler chooses one of A or B along with an element from their chosen structure.

(2) Duplicator chooses the structure which Spoiler did not choose, along with an element from that structure

After our n rounds, we will have n elements chosen from each structure. We will denote by $\overline{x} = \{x_1, \ldots, x_n\}$ the elements chosen from A and $\overline{y} = \{y_1, \ldots, y_n\}$ the elements chosen from B.

We say that these elements **determine a partial isomorphism** if

- $x_i = x_j$ if and only if $y_i = y_j$
- If R is an m-ary relation for $m > 0$, then $R(x_{i_1}, \ldots, x_{i_m})$ holds in A if and only if $R(y_{i_1}, \ldots, y_{i_m})$ holds in B for $1 \leq i_1, \ldots i_m, \leq n$.

Duplicator wins if the tuples $\bar{x}$ and $\bar{y}$ determine a partial isomorphism, and **Spoiler wins** otherwise.

We say that **Duplicator has a winning strategy** in an EF-game if Duplicator can play in a way that guarantees that they win. **Spoiler has a winning strategy** if they can guarantee that Duplicator loses, meaning Spoiler will win.

We modify this definition of the game as follows:

Definition 2.6. Let p be a quantifier string of length n and A and B be first-order structures. The n-round **EF game with quantifier string** p (also notated **EF(p) game**) on A and B is the n-round EF game on A and B with the following restrictions.

Each round in the EF(p) game corresponds to a quantifier in p, read from left to right.

- On an $\exists$ round, Spoiler will play in A and Duplicator will play in B.
- On a $\forall$ round, Spoiler will play in B and Duplicator will play in A.

To see how these games will help us, we introduce a few more definitions.

Definition 2.7 ([Ros05])**.** The quantifier structure of a first-order formula φ, $\mathrm{qs}(\varphi)$ will be defined inductively as follows:

- If φ is an atomic formula, $\mathrm{qs}(\varphi) = \emptyset$.
- If $\varphi = \theta_1 \wedge \theta_2$ or $\varphi = \theta_1 \vee \theta_2$, then $\mathrm{qs}(\varphi) = \mathrm{qs}(\theta_1) \cup \mathrm{qs}(\theta_2)$.
- If $\varphi = \neg\theta$, then $\mathrm{qs}(\varphi) = \mathrm{qs}(\theta)$.
- If $\varphi = \exists x\theta$, then $\mathrm{qs}(\varphi) = \exists \circ \mathrm{qs}(\theta) \cup \mathrm{qs}(\theta) \cup \{\exists\}$, and if $\varphi = \forall x\theta$, then $\mathrm{qs}(\varphi) = \forall \circ \mathrm{qs}(\theta) \cup \mathrm{qs}(\theta) \cup \{\forall\}$

Definition 2.8. For quantifier strings p and q, we will say $p \preceq q$ if p appears as a quantifier substring of q. That is, if we index the quantifiers showing up in q as

$$q = Q_1 Q_2 \ldots Q_n,$$

then $p \preceq q$ if and only if there exists some $1 \leq i_1 < i_2 < \cdots < i_m \leq n$ such that

$$p = Q_{i_1} Q_{i_2} \dots Q_{i_m}.$$

Definition 2.9. Let p be quantifier prefix. Define $\mathrm{FO}\{p\}$ to be set of all φ such that every element of $\mathrm{qs}(\varphi)$ embeds into p.

Definition 2.10. Let C be a collection of first-order sentences. We say $A \Rightarrow_C B$ if for all $\varphi \in C$, $A \models \varphi \implies B \models \varphi$.

We now state a theorem originally proved by Grädel and McColm [GM96] that connects our $\mathrm{EF}(p)$ games to the expressibility of these quantifier strings.

Theorem 2.11 ([GM96])**.** *Let p be a quantifier string and let A and B be structures. Duplicator has a winning strategy in the EF(p) game on A and B if and only if $A \Rightarrow_{FO\{p\}} B$.*

Remark 2.12. In fact, Grädel and McColm showed a stronger result for infinitary logic, though this weakened form is sufficient for our purposes.

The upshot of this theorem is that a winning strategy for Duplicator with quantifier string q will guarantee that any q-sentence will not distinguish the two structures we are analyzing. Being able to restrict the types of sentences that can distinguish between our structures will be central to our proof of the Linear Prefix Theorem. We will construct structures and a p-sentence holds in one but not the other. Then, we find a winning strategy for Duplicator in the n-round $\mathrm{EF}(q)$ game on our structures, and by applying Theorem 2.11l, we will get the desired result.

Further, note that Spoiler never benefits from selecting an already-selected element, as Duplicator will simply respond with the corresponding element in the other structure. Thus, we will assume that Spoiler always plays on a new element in our EF games.

If p is a quantifier string, let p^* denote the quantifier string obtained by switching each $\exists$ symbol in p with a $\forall$ symbol and switching each $\forall$ symbol in p with an $\exists$ symbol. We call p^* the **dual** of p.

Lemma 2.13. *Let p and q be quantifier strings and say that φ is a p-sentence which is not equivalent to any q sentence. Then, $\neg\varphi$ is a p^*-sentence which is not equivalent to any q^*-sentence.*

Proof. It is clear that, since φ is a p-sentence, $\neg\varphi$ is a p^*-sentence. Suppose for the sake of contradiction that $\neg\varphi$ were equivalent to some q^*-sentence ψ. Then, taking negations, φ is equivalent to $\neg\psi$, a q-sentence. This is not possible by assumption, so we cannot have that $\neg\varphi$ is equivalent to ψ. □

Throughout our proof, we will use this lemma to extend distinctions between certain strings of quantifiers to distinctions involving the dual strings.

3. Quantifier Strings of The Same Length

The following is our main theorem:

Theorem 3.1. *Let p and q be distinct quantifier strings of length n. Then, there exists a p-sentence which is not equivalent to any sentence in $FO\{q\}$ over structures of size $\leq 3n$.*

In all that follows, we shall be working in the language $\mathcal{L} = \{R\}$ consisting of a single n-ary relation symbol. We will also be taking

$$pR = P_1 x_1 \dots P_n x_n, R(x_1, \dots, x_n)$$

to be the p-sentence that we want to show is not equivalent to any q-sentence

Fix some p. Our proof will be broken up into two cases.

- Case 1 will be if q has a different number of $\exists$ symbols than p. Note that as p and q are the same the same length, this is the same condition as q having a different number of $\forall$ symbols than p.
- Case 2 will be if q has the same number of $\exists$ symbols (and $\forall$ symbols), meaning p and q are permutations of one another.

For each case, we will construct two $\mathcal{L}$-structures A and B such that $|A|, |B| \leq 3n$. Then, we will demonstrate that $A \models pR$ and $B \not\models pR$. Using our modified EF-games, we will then show that for any q-sentence ψ, we have that $A \models \psi \implies B \models \psi$. It follows that p cannot be equivalent to any q-sentence over A and B.

3.1. Case 1: p and q have a different number of $\exists$'s.

Theorem 3.2. *Let p be a string of quantifiers of length n which contains k $\exists$ symbols. If q is another length n string of quantifiers which contains less than k $\exists$ symbols, then pR is not equivalent to any sentence in $FO\{q\}$ over models of size $n+1$.*

Proof. We will construct structures A and B, both with the same universe, as described above.

Take the universe of both A and B to be the set $\{1, 2, \dots, n+1\}$. In A, we will say that $R(x_1, x_2, \dots, x_n)$ holds if and only if $\{1, 2, \dots, k\} \subseteq \{x_1, x_2, \dots, x_n\}$. That is, all numbers from 1 to k appear at least once in the tuple $(x_1, \dots, x_n)$. In B, we will say that $R(x_1, x_2, \dots, x_n)$ never holds.

We note that $A \models pR$, since we can choose the numbers $1, 2, \dots, k$ for the variables quantified by an $\exists$, guaranteeing that the relation holds. It is clear that $B \not\models p$, since we would need to be able to pick some collection of elements to be quantified by $\exists$ for which the relation holds, but this can never happen.

Now, let q be a quantifier string with less than k $\exists$ symbols. We will show that Duplicator has a winning strategy in the EF-game played on A and B with quantifier string q.

Duplicator's strategy will be as follows:

- On an $\exists$ round, Duplicator will play any arbitrary element in B. Note that Spoiler can play one of $1, 2, \ldots, k$ in A.
- On a $\forall$ round, Duplicator will play any element in A other than $1, 2, \ldots, k$, as long as it is possible. If all of $k+1, \ldots, n+1$ have already been played, Duplicator will play in $1, 2, \ldots, k$ arbitrarily.

This strategy is winning for Duplicator, since with less than k $\exists$ rounds, Spoiler will not be able to play all of $1, 2, \ldots, k$ in A on their own. Additionally, Duplicator will not play any of $1, 2, \ldots, k$ in A until all of $k+1, \ldots, n+1$ have been played. Since there are n rounds total in this game, this means that there will be some unplayed number between 1 and k, and, therefore, the relation will not hold in A. Since the relation never holds in B, this is a win for Duplicator.

This gives us that for any sentence $\psi \in \mathrm{FO}\{q\}$, whenever $A \models \psi$, $B \models \psi$. Hence, pR is not equivalent to any sentence in $\mathrm{FO}\{q\}$ by the argument above.

Note that the structures used have size $n+1$, proving that the result holds for models of this size! □

Now, suppose that q has more existential quantifiers than p. Then, q^* has less existential quantifiers then p^*, so this theorem gives us that $\varphi = p^*R$ is not equivalent to any q^*-sentence.

By Lemma 2.13, $\neg\varphi$ is a p-sentence which is not equivalent to any q-sentence. Hence, this same argument accounts for the case where p has more $\exists$ symbols than q.

3.2. Case 2: p and q have the same number of $\exists$'s.

Theorem 3.3. *Let p be a string of quantifiers of length n with a $\forall$ in the mth position. If q is a string of quantifiers of length n with the same number of $\exists$ and $\forall$ symbols in total as p and that agrees with p before the mth position but has an $\exists$ at the mth position, we have that pR is not equivalent to any sentence in FO$\{q\}$ over models of size in the range $[2n, 3n]$.*

To prove this, we start by fixing some prefix p.

Again, we will construct structures A and B such that

- $A \models pR$,
- $B \not\models pR$,
- For any q-sentence ψ, with q as above, if $A \models \psi$, then B models ψ.

And again, this will show that pR is not equivalent to any q-sentence.

This case is more complicated than our previous case, so we will break our proof up into sections, first constructing our structures, and then showing which sentences they satisfy.

3.2.1. ***Constructing our models.*** We will construct two structures, A and B with the same underlying set. For each quantifier p_i appearing in the prefix p, define the set C_i to be $\{0_i, 1_i\}$ if p_i is an $\exists$ and $\{0_i, 1_i, 2_i\}$ if p_i is a $\forall$. We define the underlying set of the structures A and B to be

$$\bigsqcup_{i=1}^{n} C_i.$$

Note that the size of this underlying set is bounded below by $2n$ and above by $3n$; it is, in particular, linear in the number of quantifiers appearing in p.

Let $S_\forall$ and $S_\exists$ denote the set of indices of the universal and existential quantifiers in the string p respectively.

Next, we make some definitions that will help us define the interpretation of R in both A and B (which we will denote as R^A and R^B respectively). Essentially, we deem tuples "good" in a way that makes it so that every choice made in an $\exists$ position depends on all the choices made in $\forall$ positions beforehand.

Definition 3.4. Let $\bar{x}$ be a tuple of elements of A which has at most one element from each C_i. Define $\rho_i(\bar{x})$ as the unique element of $\bar{x}$ which is in C_i, when such an element exists.

Definition 3.5. A tuple $(x_1, \ldots, x_n)$ of length n is **good** if it contains one element from each C_i, and for each $i \in S_\exists$,

$$\rho_i(\bar{x}) = \sum_{j<i, j\in S_\forall} \rho_j(\bar{x}) \mod 2$$

where we take the empty sum to also be 0.

We would also like to define some notion of a collection of elements being able to extend to a good tuple, meaning that there are additional elements that can be chosen to form a good tuple overall.

Given $\bar{x} = (x_1, \ldots, x_i)$ and $\bar{y} = (y_1, \ldots, y_j)$, let

$$(\bar{x}, \bar{y}) = (x_1, \ldots, x_i, y_1, \ldots, y_j).$$

Let R^A consist of the n-length tuples $\bar{x}$ satisfying either of the following:

(1) For some $i \in S_\forall$, $x_i \notin C_i$.

(2) $\bar{x}$ is good, and $x_i \in C_i$ for all i.

Let $R^B = R^A \setminus \{(0_1, 0_2, \ldots, 0_n)\}$.

The only difference between A and B is that, $R(0_1, \ldots, 0_n)$ holds in A (since $(0_1, \ldots, 0_n)$ is a good tuple) but in B if $x_i = 0_i$ for $i \in S_\forall$, then for $i \in S_\exists$, no choice of x_i will be

able to make $R(x_1, \ldots, x_n)$ hold. This being the sole difference between the structures is an important part of the rest of our proof.

3.2.2. ***Our sentence separates these models!*** To show $A \models pR$, i.e.

$$A \models P_1 x_1, \ldots P_n x_n, R(x_1, \ldots x_n),$$

we can consider arbitrary choices for x_i where $i \in S_\forall$. Then, for $i \in S_\exists$, letting

$$x_i = \sum_{j<i, j \in S_\forall} x_j \mod 2$$

and choosing $x_i \in C_i$, we get a tuple $\bar{x}$ such that $R(\bar{x})$ holds, showing that $A \models pR$.

To show $B \not\models pR$, consider choosing $x_i = 0_i$ for $i \in S_\forall$. Note that as condition (1) isn't satisfied (as for all $i \in S_\forall$, $x_i \in C_i$), we would need condition (2) to be satisfied, i.e. $\bar{x}$ has to be good. But for $i \in S_\exists$, the choices for positions in $S_\forall$ now also forces

$$x_i = \sum_{j<i, j \in S_\forall} x_j = 0_i$$

meaning the only way for for our tuple to be good would be with the tuple $(0_1, \ldots, 0_n)$. However, as this tuple has been removed from R^B, there are no choices that can be made for $R(\bar{x})$ in B to hold, meaning that $B \not\models pR$.

3.2.3. ***Duplicator's strategy!*** Suppose p has a $\forall$ symbol in the mth position. Let q be a quantifier of length n that:

- Has the same number of $\forall$ symbols and $\exists$ symbol as p
- Agrees with p before the mth position ($P_i = Q_i$ for $i < m$)
- And has an $\exists$ symbol in the mth position.

We can play an EF-game determined by q on our structures A and B, which we claim Duplicator will win! By Theorem 2.11, this will show that for any sentence $\psi \in \mathrm{FO}\{q\}$, $A \models \psi \implies B \models \psi$.

As part of Duplicator's strategy, we will say that Duplicator will always play in the same C_i as Spoiler. We will assume that Spoiler always plays in a new C_i. If this does not occur, then Duplicator can simply repeat as well and guarantee a win. This is because if not all the C_i in are played in, then the whether or not the relation holds for the elements that were played will only depend on which C_i were played in, so Duplicator will be able to preserve a partial isomorphism as long as they play in the same positions as Spoiler.

We now describe Duplicator's strategy on the kth round of the game. Let $\bar{a}_{k-1} = (a_1, \ldots, a_{k-1})$ and $\bar{b}_{k-1} = (b_1, \ldots, b_{k-1})$ be the elements of A and B that have been played so far. Let T_{k-1} denote the set of i such that C_i has been played in so far.

Duplicator's will play as follows:

- $\forall$ round: Say Spoiler plays b_k, such that $(b_1, \ldots, b_k)$ is consistent. Then, Duplicator will respond with some a_k such that $(a_1, \ldots, a_k)$ is consistent. Similarly, if $(b_1, \ldots, b_k)$ is inconsistent, then Duplicator will play a_k such that $(a_1, \ldots, a_k)$ is inconsistent. We demonstrate in the following section that this is possible in Proposition 3.11.
- On a $\exists$ round, if Spoiler plays on C_i in A where $i \in S_\forall$, and where $(a_1, \ldots, a_k)$ is consistent, then Duplicator will play $b_k = 2_i$ if either $(b_1, \ldots, b_{k-1}, 0_i)$ or $(b_1, \ldots, b_{k-1}, 2_i)$ would be consistent in B (as, if one choice is consistent the other would be as well). Otherwise Duplicator can play $b_k = 1_i$, leading to $(b_1, \ldots, b_k)$ being consistent by Proposition 3.11. If $(a_1, \ldots, a_k)$ is inconsistent, then by Proposition 3.11, Duplicator can also play b_k such that $(b_1, \ldots, b_k)$ is inconsistent.

  Otherwise, suppose Spoiler plays on an C_i in A where $i \in S_\exists$. Let T be the set of all j such that Spoiler has played in C_j prior to this round. If i is in a T-determined block, then Duplicator will match Spoiler's move up to consistency again, which is doable by Proposition 3.11. If neither of the above conditions holds, then Duplicator can play 1.

We will soon show that Duplicator can both follow this strategy and that it will guarantee them a win!

3.2.4. ***Some Definitions and a Lemma.*** Before that, we will define an equivalence relation on $S_\exists \cup \{0\}$ which will help us describe the structure of $\forall$'s and $\exists$'s at any point in our EF-game.

Definition 3.6. Let $T \subseteq \{1, \ldots, n\}$ be any subset. For $i, j \in S_\exists \cup \{0\}$, we will say $i \sim_T j$ if and only if for all k such that $\min(i, j) < k < \max(i, j)$, if $\ell \in S_\forall$, then $k \in T$.

One can quickly verify that $\sim_T$ is indeed an equivalence relation on $S_\exists \cup \{0\}$ for all $T \subseteq \{0, 1, \ldots, n\}$.

Definition 3.7. Fix $T \subseteq \{1, \ldots, n\}$.

- Each $\sim_T$-equivalence class is called a **T-block**.
- A T-block is called **T-determined** provided that it contains either 0 or some element of T.
- An index $i \in S_\forall$ is called a **T-hinge** provided that $i \notin T$ and there exist $k, \ell \in S_\exists \cup \{0\}$ such that $k < i < \ell$, k and ℓ are in different T-determined T-blocks, and for all $k < j < \ell$ for which $i \neq j$ and $j \in S_\forall$, we have that $j \in T$.

In other words, a T-hinge is the only $\forall$ index separating two T-determined T-blocks.

The idea behind these definitions is that if Spoiler plays in a T-determined block, Duplicator will have their choice "determined" in some sense in that at most one of 0 or 1 will be consistent with previous choices, and so if Spoiler plays a consistent move, Duplicator will only have one choice that will also be consistent.

Notation 3.8. *Given $\bar{x}$ as above, let $T_{\bar{x}}$ denote the collection of indices i such that such some element of $\bar{x}$ is in C_i. We will use $\sim_{\bar{x}}$, $\bar{x}$-block, $\bar{x}$-determined, and $\bar{x}$-hinge as synonyms for $\sim_{T_{\bar{x}}}$, $T_{\bar{x}}$-block, $T_{\bar{x}}$-determined, and $T_{\bar{x}}$-hinge.*

Definition 3.9. For a subset $T \subseteq \{1, \ldots, n\}$ and a collection of elements $x_i \in C_i$ for $i \in T$, we call the collection $\{x_i \mid i \in T\}$ **consistent** provided that there exists a collection of elements $y_j \in C_j$ for $j \in \{1, \ldots, n\} \setminus T$ such that the n-tuple $(\bar{x}, \bar{y})$ is a good tuple.

We say $\bar{X}$ is an **extension** of $\bar{x}$ to a good tuple if $\bar{X}$ is a permutation of $(\bar{x}, \bar{y})$ and is a good tuple.

We now introduce a lemma that allows us to characterize consistent tuples in a way that will be useful for us to work with.

Lemma 3.10. *A tuple $\bar{x}$ is consistent if and only if all of the following hold:*

(1) No two elements of $\bar{x}$ are contained in the same C_i.

(2) If $i, j \in T_{\bar{x}} \cap S_\exists$ are in the same $\bar{x}$-block and $i < j$ then

$$\rho_j(\bar{x}) = \rho_i(\bar{x}) + \sum_{i<k<j, k \in S_\forall} \rho_k(\bar{x}) \mod 2.$$

(3) If $i \in T_{\bar{x}} \cap S_\exists$ and $0 \sim_{\bar{x}} i$ then

$$\rho_i(\bar{x}) = \sum_{k<i, k \in S_\forall} \rho_k(\bar{x}) \mod 2.$$

Proof. We show both directions of the if and only if!

($\Rightarrow$) Suppose a tuple $\bar{x}$ is consistent, meaning there is some n-tuple $\bar{X}$ that is an extension of $\bar{x}$ to a good tuple. We now show that all 3 conditions hold.

- As no two elements of $\bar{X}$ are contained in the same C_i, condition (1) will be satisfied in $\bar{x}$ as its elements are a subset of the elements of $\bar{X}$.
- Choose $i, j \in T_{\bar{x}} \cap S_\exists$ that are in the same $\bar{x}$-block with $i < j$. Note that $\rho_i(\bar{x}) = \rho_i(\bar{X})$ as they contain the same unique element from C_i. The same will hold for j, as well as any $k \in S_\forall$ such that $i < k < j$, as i and j are in the same $\bar{x}$ block. As $\bar{X}$ is good, we have that

  $$\rho_i(\bar{X}) = \sum_{k<i, k \in S_\forall} \rho_k(\bar{X}) \mod 2$$

  as well as

  $$\rho_j(\bar{X}) = \sum_{k<j, k \in S_\forall} \rho_k(\bar{X}) \mod 2,$$

so as $j > i$, we have

$$\rho_j(\bar{X}) = \rho_i(\bar{X}) + \sum_{i<k<j, k\in S_\forall} \rho_k(\bar{X}) \mod 2.$$

As all of these indices will also be in $T_{\bar{x}}$, we get that

$$\rho_j(\bar{x}) = \rho_i(\bar{x}) + \sum_{i<k<j, k\in S_\forall} \rho_k(\bar{x}) \mod 2,$$

showing that $\bar{x}$ satisfies condition (2).

- Finally, if $i \in T_{\bar{x}} \cap S_\exists$ and $0 \sim_{\bar{x}} i$, then by the same reasoning as above, $\rho_i(\bar{x}) = \rho_i(\bar{X})$ and $\rho_k(\bar{x}) = \rho_k(\bar{X})$ for $k < i$. Thus, $\bar{X}$ being good gives us that

$$\rho_i(\bar{X}) = \sum_{k<i, k\in S_\forall} \rho_k(\bar{X}) \mod 2,$$

so

$$\rho_i(\bar{x}) = \sum_{k<i, k\in S_\forall} \rho_k(\bar{x}) \mod 2,$$

showing that $\bar{x}$ satisfies condition (3).

Thus, $\bar{x}$ being consistent means that all of the above conditions will be satisfied!

($\Leftarrow$) Suppose that all of the above conditions hold for $\bar{x}$, we want to show that $\bar{x}$ can be extended to a good tuple.

Let $\bar{x}^0 = \bar{x}$. We can recursively construct $\bar{x}^i$ as follows.

- If $i \in T_{\bar{x}}$, $\bar{x}^i = \bar{x}^{i-1}$
- Otherwise, let $\bar{x}^i = \bar{x}^{i-1} \cup \{x_i\}$, where x_i is chosen as an element of C_i as follows:
  - If $i \in S_\exists$ and is in a $\bar{x}^{i-1}$-determined block with $i \sim_{\bar{x}^{i-1}} j$ where $j \in T_{\bar{x}^{i-1}}$ and $j < i$, then

$$x_i = \rho_j(\bar{x}^{i-1}) + \sum_{j<k<i, k\in S_\forall} \rho_k(\bar{x}^{i-1}) \mod 2.$$

    Note that as addition and subtraction are the same mod 2 this will be the same if $i < j$ or $i > j$, so if $i < j$ we can sum over k such that $i < k < j$ instead.

  - If $i \in S_\forall$ and is a $\bar{x}^{i-1}$-hinge, choose k and ℓ as in the definition of a T-hinge. By construction, for all $j < i$, $j \in T_{\bar{x}^{i-1}}$, so we can choose $k = 0$. As ℓ will be chosen from a $\bar{x}^{i-1}$-determined block, we can choose l such that $\ell \in T_{\bar{x}^{i-1}}$ Then, let

$$x_i = \rho_\ell(\bar{x}^{i-1}) - \sum_{j<\ell, j\neq i, j\in S_\forall} \rho_j(\bar{x}^{i-1}) \mod 2.$$

– If neither of the above hold, take $x^i = 0$.

Using properties of determined blocks, it can be shown that this construction will be well defined.

By the end this construction, we claim that $\bar{x}^n$ will be a good tuple, which we now demonstrate.

As $\bar{x}$ only contained elements from different C_i and our construction only added elements from new C_i, it is clear that $\bar{x}^n$ is indeed an n-tuple. Thus, for each i, we have $\rho_i(\bar{x}^n) \in C_i$, so we can consider the n-tuple

$$\bar{X} = (\rho_1(\bar{x}^n), \rho_2(\bar{x}^n), \ldots, \rho_n(\bar{x}^n)).$$

We will now demonstrate that $\bar{X}$ is a good tuple. Note that by construction, $X_i \in C_i$ for all i, so we just need to show that the other conditions are satisfied.

Fix $i \in S_\exists$. We want to show that

$$X_i = \sum_{j<i, j\in S_\forall} X_j \mod 2.$$

Note that for i for which $0 \sim_{\bar{x}} i$, we have this by condition (3).

Suppose there is some $i \in S_\exists$ such that

$$X_i \neq \sum_{j<i, j\in S_\forall} X_j \mod 2$$

and that $0 \not\sim_{\bar{x}} i$, and also suppose this i is minimal, so that the condition holds for all $\tilde{i} < i$ such that $\tilde{i} \in S_\exists$.

Now let $\tilde{i} \in S_\exists \cup \{0\}$ be such that $\tilde{i} < i$ and for all j such that $\tilde{i} < j < i$, we have that $j \in S_\forall$.

We break into 3 cases:

- Suppose that for all such j such that $\tilde{i} < j < i$, $j \in T_{\bar{x}}$. Then, we have that $\tilde{i} \sim_{\bar{x}} i$, so by either conditions (2) or (3) (if both $\tilde{i}, i \in T_{\bar{x}}$) or the step of our construction where either $X_{\tilde{i}}$ or X_i was added, we get that

$$X_i = X_{\tilde{i}} + \sum_{\tilde{i}<j<i, j\in S_\forall} X_j \mod 2,$$

and as we chose X_i to be minimal, we get

$$X_i = \sum_{j<i, j\in S_\forall} X_j \mod 2,$$

contradicting our assumption that X_i did not satisfy this equation!

- Now suppose that at least one j such that $\tilde{i} < j < i$ and $j \notin T_{\bar{x}}$, and that i had been in a $\bar{x}^{j-1}$-determined block. Then, the greatest such j will have been an $\bar{x}^{j-1}$ hinge, as the indices prior to it will constitute one determined block and the block containing i will come immediately after.

  Suppose $i \in T_{\bar{x}}$, then we can take $l = i$ in our construction of $\bar{x}^j$, so that

  $$X_j = X_i - \sum_{k<i, k\neq j, k\in S_\forall} X_k \mod 2$$

  meaning that

  $$X_i = \sum_{k<i, k\in S_\forall} X_k \mod 2$$

  again giving a contradiction!

  If $i \notin T_{\bar{x}}$, then on the ith round, we will have that $i \sim_{\bar{x}_{i-1}} 0$, so

  $$X_i = \sum_{k<i, k\in S_\forall} X_k \mod 2,$$

  also leading to a contradiction!

- The last case is if there is some j such that $\tilde{i} < j < i$ with $j \notin T_{\bar{x}}$, and that i had not been in a $\bar{x}^{j-1}$-determined block. Again taking j to be the greatest such j, after $\bar{x}^j$ has been constructed, we will have that i will be in a $\bar{x}^j$-determined block containing all earlier $\exists$'s as well as 0, as for all $k < i$, $k \in T_{\bar{x}^j}$. Thus, when constructing $\bar{x}^i$, $i \sim_{\bar{x}^{i-1}} 0$, thus giving us that

  $$X_i = \sum_{k<i, k\in S_\forall} X_k \mod 2,$$

  again contradicting our assumption!

Thus, no matter what, we must have that for all $i \in S_\exists$,

$$X_i = \sum_{j<i, j\in S_\forall} X_j \mod 2,$$

so $\bar{X}$ is a good tuple, as desired!

□

These definitions and lemma will be important for the next few sections!

3.2.5. ***Proving Duplicator's can play this strategy!*** We first show the following proposition:

Proposition 3.11. *Let $(a_1, \ldots, a_{k-1})$ be the played elements of A, and let $(b_1, \ldots, b_{k-1})$ be the played elements of B. Say that Spoiler plays in A and Duplicator plays in B. If Spoiler plays a_k such that $(a_1, \ldots, a_k)$ is consistent, then Duplicator can choose some b_k such that $(b_1, \ldots, b_k)$ is consistent. If $(a_1, \ldots, a_k)$ is inconsistent, then Duplicator can choose b_k such*

that $(b_1, \dots, b_k)$ is inconsistent. The same holds for if Spoiler plays in B and Duplicator plays in A.

Proof. Suppose that $(a_1, \dots, a_{k-1})$ and $(b_1, \dots, b_{k-1})$ are both consistent in their respective structure. If Spoiler plays an element a_k from C_t in A such that the played elements of A are still consistent, then Duplicator can play the element b_k from C_t that would be in a good extension of the played elements of B, as a good tuple will contain elements from each C_t. By symmetry, the same holds if Spoiler plays in B and responds in A.

If the played elements were inconsistent in both A and B, then any move by Spoiler or Duplicator will still lead to inconsistent tuples.

Now suppose Spoiler plays a move $a_k \in C_t$ in A such that the played elements $\bar{a}_{k-1} = (a_1, \dots, a_{k-1})$ are consistent but the elements $\bar{a}_k = (a_1, \dots, a_k)$ are not. Say that the played elements of B are $\bar{b}_{k-1} = (b_1, \dots, b_{k-1})$ and are consistent, then we want to show that Duplicator can choose some $b_k \in C_t$ such that $\bar{b}_k = (b_1, \dots, b_k)$ is also inconsistent.

We can go through the ways that Spoiler could have made $(a_1, \dots, a_k)$ inconsistent, by Lemma 3.10:

- For condition (1) if two elements in $(a_1, \dots, a_k)$ lie in the same C_t, then it must be the case that there is some $1 \leq i < k$ such that $a_i, a_k \in C_t$. As such, Duplicator can choose b_k in in the same C_t as b_i to break consistency.
- For either conditions (2) and (3) to have failed, Spoiler will need to have had a_k come from C_t such that t is an $\bar{a}^{k-1}$-hinge or if t is in a $\bar{a}^{k-1}$-determined block. This is because as long as $\bar{a}^{k-1}$-hinges aren't played, then $\bar{a}^{k-1}$-blocks will stay the same. Additionally, a play in a block that is not determined will neither violate condition (2) as there will only be one played element in said block nor violate condition (3), as the block will not contain 0.

  Thus, Spoiler must have played in C_t where t is an $\bar{a}^{k-1}$-hinge or if t is in a $\bar{a}^{k-1}$-determined block. As Duplicator has matched which C_i has been played in, we have that either t is an $\bar{b}^{k-1}$-hinge or if t is in a $\bar{b}^{k-1}$-determined block

  If t is a $\bar{b}^{k-1}$-hinge, taking d, l from the definition of being a hinge (substituting d for the k in the hinge definition as we are already using k here), then Duplicator can choose

$$b_k = 1 + \rho_l(\bar{b}^{k-1}) - \rho_d(\bar{b}^{k-1}) - \sum_{d<j<l, j\neq t, j\in S_\forall} \rho_j(\bar{b}^{k-1}) \mod 2.$$

  and if t is in a $\bar{b}^{k-1}$-determined block with $t \sim_{\bar{b}^{k-1}} j$ where $j \in T_{\bar{b}^{k-1}}$ and $j < t$, Duplicator can pick,

$$b_k = 1 + \rho_j(\bar{b}^{k-1}) + \sum_{j<i<t, i\in S_\forall} \rho_i(\bar{b}^{k-1}) \mod 2$$

and if $t < j$ we can sum over i for $t < i < j$.

This will result in $\bar{b}^k$ being inconsistent, as we have chosen b_k to be an element that is misaligned with the already played element of B.

Thus, if Spoiler violated either condition (2) or (3), Duplicator can do the same!

So overall, if Spoiler plays a_k such that $(a_1, \ldots, a_k)$ is consistent, Duplicator can respond with b_k such that $(b_1, \ldots, b_k)$ is consistent, and if Spoiler makes $(a_1, \ldots, a_k)$ inconsistent, Duplicator can make $(b_1, \ldots, b_k)$ inconsistent. And using a similar argument, the analogous statement will hold if Spoiler plays in B and Duplicator plays in A, proving the proposition.

□

The previous proposition demonstrates that Duplicator can play as the strategy describes. We now show the Duplicator will win by using this strategy.

3.2.6. ***Proving Duplicator's strategy wins!***

Proposition 3.12. *The above strategy for the EF game determined by q on A and B will be winning for Duplicator.*

Proof. First, we show how the whether or not the tuples of elements that have been played are consistent determine which player wins.

Lemma 3.13. *Let $\bar{a} = (a_1, \ldots, a_n)$ and $\bar{b} = (b_1, \ldots, b_n)$ be the played elements of A and B respectively by the end of the game. If $\bar{a}$ and $\bar{b}$ are both good tuples and $\bar{b}$ is not a permutation of $(0_1, \ldots, 0_n)$, then Duplicator won the game.*

If $\bar{a}$ and $\bar{b}$ are both not good tuples, then Duplicator also won the game.

Proof. As R is the only relation in $\mathcal{L}$, we have that Duplicator will win as long as a tuple consisting of elements from $\bar{a} = (a_1, \ldots, a_n)$ is in R^A if only if the corresponding tuple of elements from $\bar{b} = (b_1, \ldots, b_n)$ is in R^B.

Recall that Duplicator always plays in the same C_i as Spoiler, and Spoiler will play in each C_i. Thus, if a tuple consisting of elements from $\bar{a}$ contains an element from the "wrong" C_i in one position, then the corresponding tuple with elements from $\bar{b}$ will have an element from the same C_i in the same position. As whether or not R holds in A and B only depends on the exact elements if, for each i, the ith element comes from C_i, this is the only the case we need to consider.

Thus, let $\bar{\alpha} = (\alpha_1, \ldots, \alpha_n)$ and $\bar{\beta} = (\beta_1, \ldots, \beta_n)$ be defined such that $\alpha_i = \rho_i(\bar{a})$ and $\beta_i = \rho_i(\bar{b})$.

Then, we want to show that

$$R^A(\bar{\alpha}) \iff R^B(\bar{\beta}).$$

$R^A(\bar{\alpha})$ will hold when $\bar{a}$ is good, and $R^B(\bar{\beta})$ will hold when both $\bar{b}$ is good and $\bar{\beta} \neq (0_1, \ldots, 0_n)$, meaning $\bar{b}$ is not a permutation of $(0_1, \ldots, 0_n)$. Thus, Duplicator will win whenever $\bar{a}$ is good, $\bar{b}$ is good, and $\bar{b}$ is not a permutation of $(0_1, \ldots, 0_n)$, as this would imply that $R^A(\bar{\alpha})$ and $R^B(\bar{\beta})$ hold. Duplicator also wins if neither $\bar{a}$ nor $\bar{b}$ is good, as in that case neither $R^A(\bar{\alpha})$ nor $R^B(\bar{\beta})$ will hold.

□

Corollary 3.14. *Say that after the ith round, $\bar{a}_i = (a_1, \ldots, a_i)$ and $\bar{b}_i = (a_1, \ldots, a_i)$ are the played elements of A and B respectively. If $\bar{a}_i$ and $\bar{b}_i$ are both inconsistent, then Duplicator will win the game.*

Proof. If $\bar{a}_i$ and $\bar{b}_i$ are inconsistent, then both $\bar{a} = (a_1, \ldots, a_n)$ and $\bar{b} = (b_1, \ldots, b_n)$ will not be good at the end of the game, so by Lemma 3.13, Duplicator will win. □

Next, we establish a condition for which sets of elements Spoiler can play on either an $\exists$ round or a $\forall$ round. Note that an $\exists$ round corresponds to an $\exists$ symbol that is in q, whereas $i \in S_\exists$ when the ith quantifier of p is an $\exists$.

Lemma 3.15. *On an $\exists$ round, if Spoiler selects an element from C_i for $i \in S_\forall$, Duplicator will have a winning strategy. Similarly, on a $\forall$ round, if Spoiler selects an element from C_i for $i \in S_\exists$, Duplicator will have a winning strategy.*

Proof. Suppose that, on an $\exists$ round, Spoiler plays in C_i in A where $i \in S_\forall$, and that this occurs on the kth round.

By Proposition 3.11, Duplicator can match Spoiler's move up to consistency.

If Spoiler plays a_k such that $(a_1, \ldots, a_k)$ is inconsistent, Duplicator can play b_k such that $(b_1, \ldots, b_k)$ is inconsistent. By Corollary 3.14, Duplicator wins if this happens.

If Spoiler plays a_k such that $(a_1, \ldots, a_k)$ is consistent, then Duplicator can play some $b_k \in C_i = \{0_i, 1_i, 2_i\}$ that will make $(b_1, \ldots, b_k)$ consistent. As $0 = 2 \mod 2$, if $b_k = 0_i$ is consistent, then $b_k = 2_i$ will also be consistent.

Thus, Duplicator can play whichever of $b_k = 1_i$ and $b_k = 2_i$ is consistent. And as Duplicator can match Spoiler's future moves up to consistency, the tuples $\bar{a} = (a_1, \ldots, a_n)$ and $\bar{b} = (b_1, \ldots, b_n)$ will either both be good with $\bar{b}$ not a permutation of $(0_1, \ldots, 0_n)$ (as $b_k \neq 0_i$), or both tuples will not be good. In either case, by Lemma 3.13, Duplicator will win.

Thus, Duplicator will win if, on an $\exists$ round, Spoiler chooses to play in C_i for $i \in S_\forall$.

Recall that q and p have the same number of $\exists$'s and $\forall$'s and Spoiler will play on a different C_i each round. This implies that on a $\forall$ round, if Spoiler plays on C_i for $i \in S_\exists$,

then Duplicator will win, as this would force Spoiler to later play on C_i for $i \in S_\forall$ on an $\exists$ round.

Thus, Duplicator will win if Spoiler plays an element from C_i for $i \in S_\forall$ on an $\exists$ round or if Spoiler plays on C_i for $i \in S_\exists$ on a $\forall$ round.

□

We now show that if Spoiler plays "out of order" before the mth round, Duplicator will be able to win.

Definition 3.16. Define a **$\forall$-segment** as a maximal consecutive string of $\forall$'s and an **$\exists$-segment** as a maximal consecutive string of $\exists$'s.

As p and q agree up to the mth quantifier, a segment that exists in one of them entirely before m will match with the indices in the other.

We also demonstrate that, on an $\exists$ round, if Spoiler plays in an $\exists$ block that is not determined, then Duplicator will win.

Lemma 3.17. *Let T_{k-1} be the set of i such that C_i has been played before the kth round. On an $\exists$ round, if Spoiler plays a_k on some C_i such that $i \in S_\exists$ and i is not in an T_{k-1}-determined block, then Duplicator wins.*

Proof. As i is not in a T_{k-1}-determined block, any a_k that Spoiler could have chosen will create a consistent tuple $(a_1, \ldots, a_k)$. By the same reasoning, Duplicator can make any choice of $b_k \in C_i$ and $(b_1, \ldots, b_k)$. In particular, they can choose $b_k = 1_i$. This now guarantees that $(b_1, \ldots, b_n)$ will not be a permutation of $(0_1, \ldots, 0_n)$, so as long as $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$ are both good or both not good, then Duplicator will win. □

Lemma 3.18. *Consider the kth round of the game, where $k < m$. If the kth round i an $\exists$ round, then Duplicator will win if Spoiler plays in C_i where the ith quantifier of p is not in the same $\exists$-segment as the kth quantifier of p.*

If the kth round is a $\forall$ round and there is some $\tilde{k}$ such that $k < \tilde{k} < m$ and the $\tilde{k}$th quantifier of p is an $\exists$, then Duplicator will win if Spoiler plays in C_i where the ith quantifier of p is not in the same $\forall$-segment as the kth quantifier of p.

Proof. We prove this inductively, assuming that the statement holds for all $j < k$.

As a result of Lemma 3.15, we can assume that Spoiler only plays in C_i for $i \in S_\exists$ in $\exists$ rounds and C_i for $i \in S_\forall$ in $\forall$ rounds.

Let T_{k-1} be the set of i such that C_i has been played before the kth round.

Suppose that for an $\exists$ in the kth position of p, and Spoiler played all prior quantifiers in the appropriate blocks. Thus, $i \in T_{k-1}$ has been played in for all $i < k$ corresponding to

quantifier segments prior to one containing the kth quantifier. For $i \in S_\forall$ such that $i > k$, we also have that $i \notin T_{k-1}$.

Consider $\tilde{k} \in S_\exists$ such that $\tilde{k} > k$ and k and $\tilde{k}$ are in different $\exists$-segments. As $\tilde{k}$ is in a different $\exists$ segment, there must be some j such that $j \in S_\forall$ and $k < j < \tilde{k}$. As $j \notin T_{k-1}$, it must be that $\tilde{k} \not\sim_{T_{k-1}} 0$. By assumption, if $i > j$, then $i \notin T_{k-1}$. As a result, $\tilde{k}$ cannot be in a T_{k-1} determined block, so by Lemma 3.17, if Spoiler played an element of $C_{\tilde{k}}$, Duplicator will win.

Thus, Duplicator will win if Spoiler plays in an $\exists$-segment that follows the $\exists$-segment containing k. By our inductive hypothesis, $\exists$-segments prior to k will have already been filled, and by Lemma 3.15, Duplicator will win if Spoiler plays on C_i where $i \in S_\forall$. Thus, if Spoiler doesn't play in the $\exists$-segment containing k, then Duplicator will win!

Now suppose that there is a $\forall$ in the kth position of p, that Spoiler played all prior quantifiers in the appropriate blocks, and that there is another $\exists$-segment following the kth position prior to the mth position. Again, $i \in T_{k-1}$ has been played in for all $i < k$ corresponding to quantifier segments prior to one containing the kth quantifier. For $i \in S_\exists$ such that $i > k$, we also have that $i \notin T_{k-1}$.

Let $j < m$ be the position of the earliest $\exists$ symbol following the kth position. Consider $\tilde{k} \in S_\forall$ such that $\tilde{k} > k$ and k and $\tilde{k}$ are in different $\forall$ segments, meaning $\tilde{k} > j$.

Suppose Spoiler plays on some element of $C_{\tilde{k}}$ on the kth round. Then, on the jth round, not all of the $i \in S_\forall$ such that $i < j$ will have been played, so $i \not\sim_{T_j} 0$. Furthermore, by assumption, Spoiler will have only played on $\exists$-segments preceding k, as any rounds in between the kth and jth round will have been a $\forall$ round, which Spoiler must play on $\forall$-segments by Lemma 3.15. Thus, there will be no determined $\exists$-blocks that have yet to be played on by the jth round, meaning that Spoiler can only play on an undetermined $\exists$-block, which by Lemma 3.17 will give Duplicator the win!

Combining the above cases, we get that every $\exists$ prior to the mth position must be played in the same $\exists$-segment, and every $\forall$-segment followed by an $\exists$-segment prior to position the mth position. must also be played in the same $\forall$-segment, as desired!

□

Finally, we demonstrate that, because q has an $\exists$ in the mth position while p has a $\forall$, Duplicator will always be able to win!

Suppose that the final $\forall$-segment before m consists of d many $\forall$'s. In p, there will be another $\forall$ in the mth position, so there will be at least $k + 1$ $\forall$'s between the earliest $\exists$ following the kth position (note that there will be such an $\exists$ as we have the same number of $\exists$ in each quantifier string). However, in q, there is an $\exists$ in the kth position, meaning that in the m $\forall$ rounds before the mth round, Spoiler can only ever play on at most d of the $\forall$'s in p. As a result, at least 1 of the $\forall$'s prior to the earliest $\exists$ following the mth position

will not be played on. Thus, on the mth round, the earliest $\exists$ following the kth position in p will not yet be determined (as all prior $\exists$ were played earlier and are separated from this new $\exists$ by at least one missing $\forall$), meaning that this earliest $\exists$ will not be determined. The same will hold for any later $\exists$ in p, meaning that Spoiler can only play on not determined $\exists$-blocks on round m which is an $\exists$ round. And by Lemma 3.17, this will allow Duplicator to win!

So even if Spoiler plays in the appropriate quantifier segments prior to the mth position, Duplicator will be able to win using this strategy, as desired!

□

Applying Theorem 2.11, we now have that for any sentence $\psi \in \mathrm{FO}\{q\}$, we have that $A \models \psi \implies B \models \psi$. However, as $A \models pR$ and $B \not\models pR$, pR cannot be equivalent to any sentence in $\mathrm{FO}\{q\}$.

Note that the structures used in this case have size contained within $[2n, 3n]$, showing the result of Theorem 3.3, as desired! □

3.3. **What this means!** Combining the results of Sections 3.1 and 3.2, we get that if p and q are different quantifier prefixes of length n with a different number of $\exists$ symbols, then pR is not equivalent to any sentence in $\mathrm{FO}\{q\}$ over models of size $n+1$, and if p and q have the same number of $\exists$'s then pR is not equivalent to any sentence in $\mathrm{FO}\{q\}$ over models of size in $[2n, 3n]$.

Together, these results give us that pR is not equivalent to any other sentence in $\mathrm{FO}\{q\}$ where q has the same length using models of size $\leq 3n$. This is the desired result of Theorem 3.1, and concludes our proof! □

4. Quantifier Strings of Arbitrary Length

In the previous section, we demonstrated a proof of Walkoe's Linear Prefix Theorem using finite models whose sizes grow linearly in the shared length of the quantifier strings. It is natural to ask whether a similar result holds for quantifier strings who may not have a shared length. It turns out that, in certain cases, we can guarantee that certain quantifier strings can be distinguished from ones which they do not embed into over large enough finite structures.

Recall Definition 2.8. Note that if $p \preceq q$, then any p-sentence is automatically equivalent to the q-sentence were we add in the additional quantifiers and variables that appear in q, but do not change the quantifier-free part of the sentence. Hence, the questions we ask about distinguishing quantifier strings, p and q, of arbitrary length will assume that $p \not\preceq q$.

Theorem 4.1. *Let p be a length m quantifier string consisting of an alternating sequence of $\forall$ and $\exists$. If q is a length n quantifier string such that $p \not\preceq q$, then there exists a p-sentence which is not equivalent to any sentence in FO$\{q\}$ over finite models.*

4.1. **Constructing the models!** We will assume that p begins with a $\forall$ symbol. Once we have this result for statements beginning with a $\forall$, the corresponding statement for statements beginning with an $\exists$ symbol follows from Lemma 2.13. Again, our structures will be equipped with a single m-ary relation symbol.

We will construct structures A and B as in the proof of Theorem 3.1.

For $1 \leq k \leq m$, A will contain the k-tuple $(c_1, c_2, \ldots, c_k)$ where $c_i \in \{0, \ldots, n\}$ for $1 \leq i \leq k$, $c_1 \neq 0$, and $c_i = 0 \implies c_{i+1} \neq 0$ for $1 < i < k$.

For B, we use the same construction as A, but without the restriction that $c_1 \neq 0$.

Let S_k^A and S_k^B denote the set of k-tuples in A and B respectively.

That is, if an element ends in 0, then any element that immediately follows it will not end in 0, but if an element does not end in 0, then it will have some successor that does end in 0. The difference between A and B is that B has an element that ends in 0 in the first position, whereas A does not. Throughout, it will be helpful to have a dictionary by which to describe these positional relationships between elements.

Definition 4.2. (Hereditary relationships)

- An **ancestor** of a k-tuple $(c_1, \ldots, c_k)$ is some l-tuple with $l < k$ of the form $(c_1, \ldots, c_l)$.
- The **parent** of a k-tuple is the $k-1$ tuple of the above form.
- A **child** of a k-tuple is a $k+1$-tuple whose parent is the original k-tuple.
- A **descendant** of a k-tuple $(c_1, \ldots, c_k)$ is a ℓ-tuple for some $k < \ell \leq m$ which is of the form $(c_1, \ldots, c_k, c_{k+1}, \ldots, c_\ell)$.

Note that odd indices of $p = P_1 P_2 \ldots, P_m$ will be $\forall$'s and even indices be $\exists$'s. In both A and B, we will have our m-ary relation, R, hold on the tuple $(x_1, x_2, \ldots, x_m)$ whenever any of the following hold:

- $x_1 \notin S_1$
- For $i > 1$ odd, x_{i-1} is not the parent of x_i, and for $j < i-1$, $x_j \in S_j$ and x_j is the parent of x_{j+1}.
- If p ends in $\forall$, x_{i+1} is a child of x_i for $1 \leq i < m$, and x_m's last coordinate is nonzero.
- If p ends in $\exists$, x_{i+1} is a child of x_i for $1 \leq i < m$, and x_m's last coordinate is zero.

4.2. **Our sentence separates these models!** Again, note that x_i will be quantified by a $\forall$ if i is odd and by an $\exists$ if i is even. We will show that the witnesses to the variables quantified by $\exists$ in A can always be chosen so that the relation holds.

If $x_1 \notin S_1$, then the relation automatically holds, so we will assume that $x_1 \in S_1$. Regardless of the choice of x_1, we will select the element $(x_1, 0)$ to be the witness to the first existential quantifier. We claim that, in general, our witness at each existential step can be chosen to be the child of the previous tuple which has a zero in its final coordinate.

If a $\forall$ variable, x_i, is not a child of x_{i-1}, then the relation will immediately hold. Hence, we will assume that all witnesses corresponding to $\forall$ symbols in p are children of the ones corresponding to the previous $\exists$ positions.

Say that the i-th position in p is a $\forall$ and that the witness to the statement that we have constructed is $(x_1, \ldots, x_i)$. Assume that we have been able to select the child whose last coordinate is zero at each $\exists$ step. Then, the last coordinate of x_i cannot be zero, since we assumed that at the existential steps, we chose the child of the previous $\forall$ position which has a zero in the final coordinate. Thus, x_i does have a child whose last coordinate is zero, and we will select this child as x_{i+1} as long as $i < m$.

By induction, then, this choice is always possible. We are left with our m-tuple $(x_1, \ldots, x_m)$. This is a valid string of parents and children by assumption. If p ends in $\forall$, then the last coordinate, x_n, can never be zero since the last coordinate of x_{m-1} is zero. Hence, $R(x_1, \ldots, x_m)$ holds in A. If p ends in $\exists$, then the last coordinate of x_n is zero, so $R(x_1, \ldots, x_m)$ holds in A.

Thus $A \models pR$.

We can argue similarly to the above that at each $\forall$ stage, we can choose the child of the previous tuple which has a zero in the final coordinate. We claim that, regardless of the choices made at east $\exists$ position, $R(x_1, \ldots, x_n)$ will never hold in B for the resulting tuple $(x_1, \ldots, x_n)$.

Since at each odd position, we chose elements whose final coordinates are zero, we must have that the final coordinate of the elements in the even positions are nonzero.

If p ends in $\forall$, then this means that the last coordinate of x_m is zero and that we have a valid chain of parents and children. Hence, the relation will not hold in B.

Similarly, if p ends in $\exists$, then the last coordinate of x_m is nonzero and we have a valid chain of parents and children. Hence, the relation will not hold in B.

Thus $B \not\models pR$.

4.3. **Duplicator's strategy.** Note that any quantifier string which $p = \forall\exists\ldots\forall$ has length m and does not embed into will be of the form $q = \exists^{k_1}\forall^{k_2}\ldots\exists^{k_m}$. If instead $p = \forall\exists\ldots\exists$, then the strings which p does not embed into are of the form $q = \exists^{k_1}\forall^{k_2}\ldots\forall^{k_m}$. Fix q with this property and let n be the total number of quantifiers in q. We claim that Duplicator wins the EF-game on A and B played with the quantifier string q.

We will assume that Spoiler plays a tuple only if all its ancestors have been played. To do this, if Spoiler would like to play a tuple that hasn't had all of its ancestors played, we

simply add extra turns in which Spoiler plays each of its ancestors. Having more moves can only ever help Spoiler, since it gives them more chances to break a partial isomorphism. We will always have Duplicator respond to an element which lies in S_i in one structure with an element which lies in S_i in the other structure.

Duplicator's strategy will be as follows.

Say Spoiler plays some element $a = (a_1, \ldots, a_j)$ in A on an $\exists$ round.

- If $a = (a_1)$, choose some unplayed $b = (b_1)$. Since $a_1 \neq 0$, we will choose $b_1 \neq 0$.
- Otherwise, $(a_1, \ldots, a_{j-1})$ will have been played already and will correspond to some element $(b_1, \ldots, b_{j-1})$. Duplicator will respond with some $b = (b_1, \ldots, b_j)$. Again, if $a_j \neq 0$, choose an unplayed $b_j \neq 0$, and if $a_j = 0$, then choose $b_j = 0$.

  If $b_j = 0$ is not possible, then choose some child of $(b_1, \ldots, b_{j-1})$ which was unplayed previously.

Now, say Spoiler plays some $(b_1, \ldots, b_j)$ in B on a $\forall^*$ round.

- If $b = (0)$, then Duplicator will choose some unplayed $a = (a_1)$ in A. If $b = (b_1)$ where $b_1 \neq 0$, Duplicator can respond with some (a_1) with $a_1 \neq 0$.
- Otherwise, $(b_1, \ldots, b_{j-1})$ will have already been played and will correspond to some $(a_1, \ldots, a_{j-1})$ in A. If $b_j = 0$, then Duplicator will respond with $(a_1, \ldots, a_{j-1}, 0)$ if they can. If they cannot, then Duplicator will respond with some unplayed child of $(a_1, \ldots, a_{j-1})$. If $b_j \neq 0$, then Duplicator will respond with $(a_1, \ldots, a_{j-1}, a_j)$ for nonzero a_j.

We will call a move in which Spoiler plays an element with a zero in the final coordinate and Duplicator cannot respond with an element which has a zero in the final coordinate a **sad move**. In our proof that this strategy will result in a Duplicator win, we will claim that we can only have one sad move per $\forall$ or $\exists$-block in q.

4.4. **Strategy works!** Note first that whenever Spoiler plays an element with a nonzero final coordinate, Duplicator can always respond with an element with a nonzero final coordinate. This is because the game has n rounds, and there are n choices of elements whose last coordinate is nonzero in each position. Hence, at each round, there will be at least one element of this form remaining in the position Duplicator wants to play in.

Now, we will prove that Spoiler can make at most 1 sad move per $\exists$ or $\forall$ block. Since we know that Spoiler cannot make any sad moves in the first block, this will mean that the case described above, in which Spoiler makes m sad moves, will never happen.

Note that in the first k_2 $\forall$ rounds, Spoiler can make a sad move by playing (0) in B. Duplicator is not able to respond with a nonzero element from S_1 in A since they do not exist by construction. We claim that if Spoiler is to make a sad move in S_i for $i > 1$, it must have been preceded by a sad move in S_j for all $1 \leq j < i$. We will argue by induction.

Lemma 4.3. *If Spoiler plays a sad move in S_j, then this sad move must have been preceded by a sad move in S_i for $1 \leq i < j$.*

Proof. Suppose that Spoiler is able to make a sad move in S_2. That is, Spoiler has played some $(x_1, 0)$ for which Duplicator is forced to respond with (y_1, y_2) for $y_2 \neq 0$. Since Duplicator was forced to play this tuple, it must be the case that Duplicator could not have chosen $y_2 = 0$. This means that we must have $y_1 = 0$. Since we are assuming that the parents of each move played in our EF game were played beforehand, (0) must have been played in B at an earlier point in the game. However, Duplicator would not have made this move, as Duplicator will match a nonzero element in S_1 in A from spoiler with a nonzero element in S_1 in B. Hence, on an earlier $\exists$ round, Spoiler played (0) in B, and duplicator was forced to respond with (x_1) in A for some nonzero x_1. This is a sad move. Hence, a sad move for spoiler in S_2 must have been preceded by a sad move in S_1.

Now, suppose that every sad move from Spoiler in S_{j-1} was preceded by a sad move in S_i for all $1 \leq i < j-1$. Say Spoiler plays a sad move, $(x_1, \ldots, x_{j-1}, 0)$ in S_j. Duplicator will then match Spoiler with $(y_1, \ldots, y_{j-1}, y_j)$ for some y_j nonzero. Since Duplicator is unable to play zero in the coordinate where y_j exists, y_{j-1} must then be zero. Note that, previously in the game, $(x_1, \ldots, x_{j-1})$ was played in S_{j-1} and was matched with $(y_1, \ldots, 0)$. Since Spoiler was able to play $(x_1, \ldots, x_{j-1}, 0)$, x_{j-1} cannot be zero itself. If Spoiler were the player to choose $(x_1, \ldots, x_{j-1})$, then Duplicator would be able to respond with $(y_1, \ldots, c_{j-1})$ for some nonzero c_{j-1}. Since this did not happen, we deduce that Spoiler played $(y_1, \ldots, 0)$ and Duplicator responded with $(x_1, \ldots, x_{i-1})$, a tuple with a nonzero final coordinate. This is, therefore, a sad move in S_{j-1}. Hence, the sad move in S_j must have been preceded by a sad move in S_{j-1}. By induction, it was preceded by a sad move in S_i for all $1 \leq i < j$. □

Note that if Spoiler is able to make m sad moves, then they will win. This is because the tuples which are produced will be valid chains of parents and children, but one of whose final element's final coordinate is zero while the other's is not. Further, Spoiler cannot make any sad moves during the first k_1 $\exists$ rounds, since, in B, Duplicator can copy Spoiler's moves in A exactly.

We now argue by induction that Spoiler can only possibly make exactly one sad move in the rounds corresponding to each $\forall$ block and $\exists$ block after the first $\exists$ block in q, and that each consecutive sad move is a child of the tuple associated to the previous.

In the first $\forall$ block, Spoiler can play (0) in B, forcing (x_1) in A from Duplicator for some nonzero x_1. This is a sad move. If Spoiler plays $(0, y_2)$ for y_2 nonzero, Duplicator will match with (x_1, x_2) for nonzero x_2 such that the tuple (x_1, x_2) has not yet been played. Duplicator is able to respond to this nonzero element, so Spoiler cannot play a child of (0) as a sad move in this $\forall$ block. By Lemma 4.3, this means that, in this $\forall$ block, any descendant of (0) will not be a sad move.

Any nonzero element (y_1) played by spoiler in B will not be a sad move, since Duplicator can respond with some nonzero (x_1) in A. Again, by Lemma 4.3, any ancestor of (y_1) will

not be a sad move. Hence, we see that the only sad move that Spoiler can make in this round is (0) in B.

Now, suppose that Spoiler has only been able to play one sad move in each of the $\ell - 1$ $\forall$ and $\exists$ blocks following the first $\exists$ block in q, and that each of these sad moves has been a child of the previous. Suppose that the ℓ-th block in q is a $\forall$ block. Say that the previous sad move was $(y_1, \ldots, 0)$, played in A by Spoiler. Then, Duplicator will have matched this move with $(x_1, \ldots, x_{\ell-1})$ in B for some nonzero $x_{\ell-1}$. Now, Spoiler is playing in B and can play the move $(x_1, \ldots, x_{\ell-1}, 0)$. Duplicator cannot respond with an descendant of $(y_1, \ldots, 0)$ whose last coordinate is zero, so this is a sad move which has all the previous sad moves as ancestors.

If Spoiler plays any move with a nonzero final coordinate, then it will not be a sad move, and so none of its descendants will be either by Lemma 4.3. If Spoiler plays any move other than the sad move discussed above with zero as a final coordinate, this move will necessarily have at least one non-sad ancestor because all sad moves that previously exist are the ancestors of $(x_1, \ldots, x_{\ell-1})$. Hence, this move will not be sad either by Lemma 4.3.

We conclude that Spoiler can only make one sad move per $\forall$ block and that this sad move is a child of the tuple associated to the previous. A similar argument will show the case where the ℓ-th block is an $\exists$ block.

By the above argument, Spoiler can only make $m - 1$ sad moves throughout the game.

Suppose that, by the end of the game, there is some tuple $(x_1, \ldots, x_m) \subseteq A^m$ that is paired with some $(y_1, \ldots, y_m) \subseteq B^m$ such that $A \models R(x_1, \ldots, x_m)$ but $B \not\models R(y_1, \ldots, y_m)$. As Duplicator's choices preserve familial relationships, the only way for this to occur is if x_m ends in 0 but y_m does not.

However, this can only have resulted from a sad move in the final position, which is the mth position. By Lemma 4.3, Spoiler would have needed m total sad moves to achieve this, as a sad move in the ith position requires a sad move in the $i-1$th position for $i > 1$. But as demonstrated above, Spoiler will only be able to play at most $m - 1$ sad moves, contradicting our assumption that there could be such a pair of tuples with a sad move pairing x_m and y_m.

Thus, for any tuple $(x_1, \ldots, x_m) \subseteq A^m$ that is paired with some $(y_1, \ldots, y_m) \subseteq B^m$, we have that $A \models R(x_1, \ldots, x_m)$ if and only if $B \not\models R(y_1, \ldots, y_m)$, giving Duplicator the win. So by Theorem 2.11, for any sentence $\psi \in \mathrm{FO}\{q\}$, we again have that $A \models \psi \implies B \models \psi$. As $A \models pR$ and $B \not\models pR$, pR cannot be equivalent to any sentence in $\mathrm{FO}\{q\}$, and since A and B are both finite, we have demonstrated this result using finite models, as desired!

□

References

[GM96] Erich Grädel and Gregory L. McColm. Hierarchies in transitive closure logic, stratified Datalog and infinitary logic. *Ann. Pure Appl. Logic*, 77(2):169–199, 1996.

[He14] Yuguo He. On the strictness of the quantifier structure hierarchy in first-order logic. *Log. Methods Comput. Sci.*, 10(4):4:3, 38, 2014.

[KW73] H. Jerome Keisler and Wilbur Walkoe, Jr. The diversity of quantifier prefixes. *J. Symbolic Logic*, 38:79–85, 1973.

[Ros05] Eric Rosen. On the first-order prefix hierarchy. *Notre Dame J. Formal Logic*, 46(2):147–164, 2005.

[Wal70] Wilbur John Walkoe, Jr. Finite partially-ordered quantification. *J. Symbolic Logic*, 35:535–555, 1970.

Email address: gbel@umich.edu

Email address: jeremlee@hmc.edu