The Crane Beach Conjecture

DAVID A. MIX BARRINGTON *
Computer Science Department
University of Massachusetts
barring@cs.umass.edu

NEIL IMMERMAN †
Computer Science Department
University of Massachusetts
immerman@cs.umass.edu

CLEMENS LAUTEMANN
Institut für Informatik
Johannes Gutenberg-Universität Mainz
cl@informatik.uni-mainz.de

NICOLE SCHWEIKARDT
Institut für Informatik
Johannes Gutenberg-Universität Mainz
nisch@informatik.uni-mainz.de

DENIS THÉRIEN ‡
School of Computer Science
McGill University
denis@cs.mcgill.ca

Abstract

A language L over an alphabet A is said to have a neutral letter *if there is a letter $e \in A$ such that inserting or deleting e's from any word in A^* does not change its membership (or non–membership) in L.*

The presence of a neutral letter affects the definability of a language in first–order logic. It was conjectured that it renders all numerical predicates apart from the order predicate useless, i.e., that if a language L with a neutral letter is not definable in first–order logic with linear order, then it is not definable in first–order logic with any set $\mathcal{N}$ of numerical predicates.

We investigate this conjecture in detail, showing that it fails already for $\mathcal{N} = \{+, *\}$, or, possibly stronger, for any set $\mathcal{N}$ that allows counting up to the m times iterated logarithm, $\lg^{(m)}$, for any constant m.

On the positive side, we prove the conjecture for the case of all monadic numerical predicates, for $\mathcal{N} = \{+\}$, for the fragment $BC(\Sigma_1)$ of first–order logic, and for binary alphabets.

1 Introduction

Logicians have long been interested in the relative expressive power of different logical formalisms. In the last twenty years, these investigations have also been motivated by a close connection to computational complexity theory — most computational complexity classes have been given characterisations as finite model classes of appropriate logics, cf. [Imm98]. In these investigations it became apparent that in order to describe computation over a finite structure, a formula has to be able to refer to some linear order of the elements of this structure. Given such an order, the universe of the structure, i.e., the set of its elements, can be identified with an initial segment of the natural numbers. In a logic with the capability to express induction we can then define predicates for arithmetical operations such as addition or multiplication on the universe, and use them in order to describe operations on time or memory locations. In weak logics, however, e.g., first–order logic, defining an order relation does not automatically make arithmetic available. In fact, even over strings, the expressive power of first–order logic varies considerably, depending on the set of numerical predicates that can be used.

As an example, if the order is the only numerical relation then the only *regular* languages that can be defined in first–order logic are the star–free languages. If, however, for every $p \in \mathbb{N}$ we have available the predicate mod_p (which holds for a number m iff $m \equiv 0 \pmod p$) then we can express regular languages that are not star–free,

*Supported by NSF grant CCR-9988260.
†Supported by NSF grant CCR-9877078.
‡Supported by NSERC and FCAR.

such as $(000 + 001)^*$. In fact, with these predicates we can express *all* the first–order definable regular languages, cf. [Str94]. Thus, even very powerful relations (arithmetical relations, or even undecidable ones) are of no further help in defining regular languages. On the other hand, with addition, we can express languages that are not regular, such as $\{0^n 1^n \,/\, n\in\mathbb{N}\}$.

First-order logic with varying numerical predicates can also be thought of as specifying circuit complexity classes with varying *uniformity conditions* [BIS90]. The language defined by a first-order formula is naturally computed by a family of boolean circuits with constant depth, polynomial size, and unbounded fan-in (called "AC^0 circuits"). The power of such a family depends in part on the sophistication of the connections among the nodes. A formula with only simple numerical predicates leads to a circuit family where these connections are easily computable. These are called "uniform circuits", and how uniform they are is quantified by the computational complexity of a language describing the connections. A formula with arbitrary numerical predicates leads to a circuit family with arbitrary connections — the set of languages so describable is called "non-uniform AC^0".

There are languages, such as the PARITY language, for which we can prove no AC^0 circuit exists [Ajt83, FSS84]. A major open problem in complexity theory is to develop methods for showing languages to be outside of uniform circuit complexity classes even if they are in the corresponding non-uniform class. This is an additional motivation for the study of the expressive power of first-order logic with various numerical predicates, as this provides a parametrization of various versions of "uniform AC^0".

In an attempt to obtain a better understanding of this expressive power, Thérien considered the concept of a *neutral letter* for a language L, i.e., a letter e that can be inserted into or deleted from a string without affecting its membership in L. Since, in the presence of such a letter, membership in L cannot depend on specific (combinations of) letters being in specific (combinations of) positions, it seemed conceivable that neutral letters would render all numerical predicates, except for the order, useless. With this in mind, Thérien proposed what was later dubbed the *Crane Beach Conjecture*:

> If a language with a neutral letter can be defined in first–order logic using some set $\mathcal{N}$ of numerical predicates then it can be so defined using only the order relation.

One particular example of a language with a neutral letter is PARITY, consisting precisely of those 0–1–strings in which 1 occurs an even number of times. PARITY is not definable in first–order logic – no matter what numerical predicates are used (cf. [Ajt83, FSS84]). The Crane Beach conjecture would imply this result, since PARITY is a regular language known not to be star–free.

In this paper, we investigate the Crane Beach conjecture in detail. We first show that in general it is not true — in fact, it already fails for $\mathcal{N} = \{+, *\}$. However, we also show that the conjecture is true in a number of interesting special cases, including the case of addition, i.e., when $\mathcal{N} = \{+\}$.

This work is closely related to a line of research in data base theory which is concerned with so–called *collapse results* (cf. [BL00]). Here one considers a finite data base embedded in some infinite, ordered domain, and then looks at *locally generic* queries, i.e., queries which are invariant under monotone injections of the data base universe into the larger domain. In this setting, a language with a neutral letter is the special case of a locally generic (Boolean) query over monadic databases with background structure $\langle\mathbb{N}, \mathcal{N}\rangle$, and the conjecture then can be translated into a collapse for first–order logic.
We will come back to this in connection with Theorem 3.12.

Acknowledgements
We are indebted to Thomas Schwentick for bringing the data base theory connection to our attention. He also took an active part in many discussions on the subject of this paper. In particular, the first proof of Theorem 3.9 was partly due to him. The first author in particular would like to thank Eric Allender, Pierre McKenzie, and Howard Straubing for valuable discussions on this topic, many of which occurred at a Dagstuhl workshop in March 1997. Much important work on this topic also occurred at various McGill Invitational Workshops on Complexity Theory, particularly on excursions to Crane Beach, St. Philip, Barbados.

2 Preliminaries

2.1 First–Order Logic

A *signature* is a set σ containing finitely many relation, or predicate, symbols, each with a fixed arity. A σ–structure $\mathfrak{A} = \langle \mathcal{U}^{\mathfrak{A}}, \sigma^{\mathfrak{A}}\rangle$ consists of a set $\mathcal{U}^{\mathfrak{A}}$, called the *universe* of $\mathfrak{A}$ and a set $\sigma^{\mathfrak{A}}$ that contains an interpretation $R^{\mathfrak{A}} \subseteq (\mathcal{U}^{\mathfrak{A}})^k$ for each k–ary relation symbol $R \in \sigma$.
In this paper, we are concerned almost exclusively with first–order logic over finite strings. In this context, for an alphabet A we use the signature $\sigma_A := \{Q_a \,/\, a \in A\}$ and identify a string $w = w_1 \cdots w_n \in A^*$ with the structure $w = \langle\{1, \ldots, n\}, \sigma_A^w\rangle$, where $\sigma_A^w = \{Q_a^w \,/\, a{\in}A\}$ and $Q_a^w = \{i \leq n \,/\, w_i = a\}$, i.e, $i \in Q_a^w \iff w_i = a$, for all $a \in A$.
In addition to the predicates Q_a we also have *numerical predicates*. A k–ary numerical predicate P has, for every

$n \in \mathbb{N}$, a fixed interpretation $P_n \subseteq \{1, \ldots, n\}^k$. Our prime example of a numerical predicate is the linear order relation $\leq$. Where we see no danger of confusion (i.e., almost everywhere) we will not distinguish notationally between a predicate and its interpretation.

An *atomic σ–formula* is either of the form $x_1 = x_2$, or $P(x_1, \ldots, x_k)$, where $x_1, x_2, \ldots, x_k$ are variables and $P \in \sigma$ is a k–ary predicate symbol. First–order σ–formulas are built from atomic σ–formulas in the usual way, using Boolean connectives $\wedge$, $\vee$, $\neg$, etc. and universal ($\forall\, x$) and existential ($\exists\, x$) quantifiers.

For every alphabet A, and every set $\mathcal{N}$ of numerical predicates, we will denote the set of first–order $\sigma_A \cup \mathcal{N}$–formulas by $FO[\mathcal{N}]$. We define semantics of first–order formulas in the usual way. In particular, for a string $w \in A^*$ and a formula $\varphi \in FO[\mathcal{N}]$ without free variables (i.e., variables not bound by a quantifier), we will write $w \models \varphi$ if φ holds on the string w. If $x_1, \ldots, x_k$ are the free variables of φ, and if $p_1, \ldots, p_k \leq |w|$, $w \models \varphi(p_1, \ldots, p_k)$ indicates that φ holds on the string w with x_i interpreted as p_i, for every $i \leq k$.

Every formula $\varphi \in FO[\mathcal{N}]$ without free variables defines the set L_φ of those A–strings which satisfy φ. We say that a language $L \subseteq A^*$ is *definable in* $FO[\mathcal{N}]$, and write $L \in FO[\mathcal{N}]$, if $L = L_\varphi$, for some $\varphi \in FO[\mathcal{N}]$. We will use analogous notation for subsets of $FO[\mathcal{N}]$, in particular, we will consider the set $\Sigma_1[\mathcal{N}]$ of formulas which are of the form $\exists x_1 \cdots \exists x_r \psi$, for some quantifier–free $\psi \in FO[\mathcal{N}]$, and its Boolean closure, $BC(\Sigma_1[\mathcal{N}])$. (One can define a complete hierarchy of classes $\Sigma_i[\mathcal{N}]$ and $\Pi_i[\mathcal{N}]$ along with their Boolean closures, using the hierarchy of first-order formulas given by the number of quantifier alternations. But in this paper we will have need only for $BC(\Sigma_1[\mathcal{N}])$.

2.2 Ehrenfeucht–Fraïssé Games

One of our main technical tools will be (various versions of) the *Ehrenfeucht–Fraïssé game*. In our context, the Ehrenfeucht–Fraïssé game for a set of numerical predicates, $\mathcal{N}$, is played by two players, Spoiler and Duplicator, on two strings $u, v \in A^*$. There is a fixed number k of rounds, and in each round i

- first, Spoiler chooses one position, a_i in u, or a position b_i in v;
- then Duplicator chooses a position in the other string, i.e., a b_i in v, if Spoiler's move was in u, and an a_i in u, otherwise.

After k rounds, the game finishes with positions $a_1, \ldots, a_k$ chosen in u and $b_1, \ldots, b_k$ chosen in v. Duplicator has won if the mapping $a_i \mapsto b_i$, $i = 1, \ldots, k$, is a *partial* $\sigma_A \cup \mathcal{N}$*–isomorphism*, i.e., if

- for every $i, j \leq k$, $a_i = a_j \iff b_i = b_j$,
- for every $i \leq k$, a_i and b_i carry the same letter, i.e., $u_{a_i} = v_{b_i}$, and
- for every m–ary predicate $P \in \mathcal{N}$, and every $i_1, \ldots, i_m \leq k$, it holds that $P(a_{i_1}, \ldots, a_{i_m}) \iff P(b_{i_1}, \ldots, b_{i_m})$.

If Duplicator has a winning strategy in the k–round game for $\mathcal{N}$ on two strings u and v, we write $u \equiv_k^{\mathcal{N}} v$. The fundamental use of the game comes from the fact that it characterises first–order logic (c.f., e.g., [EFT94]). In our context, this can be formulated as follows:

2.1 Theorem (Ehrenfeucht, Fraïssé)
A language $L \subseteq A^*$ is definable in $FO[\mathcal{N}]$ iff there is a finite subset $\mathcal{N}'$ of $\mathcal{N}$ and a number k such that, for every $u \in L$, $v \notin L$, Spoiler has a winning strategy in the k–round game for $\mathcal{N}'$ on u and v. □

We will also use the following variant of the game:
In the single–round k–game for $\mathcal{N}$ on two strings u, v

- first, Spoiler chooses k positions $a_1, \ldots, a_k$ in u, or $b_1, \ldots, b_k$ in v;
- then Duplicator chooses k positions in the other string, i.e., positions $b_1, \ldots, b_k$ in v, if Spoiler's move was in u, $a_1, \ldots, a_k$ in u, otherwise.

Again, Duplicator wins iff the mapping $a_i \mapsto b_i$, $i = 1, \ldots, k$, is a partial isomorphism. Clearly, if Duplicator has a winning strategy for the single–round k–game on u and v, then she also has one for the single–round h–game, for all $h \leq k$.

This game characterises the expressive power of $BC(\Sigma_1[\mathcal{N}])$:

2.2 Theorem
A language $L \subseteq A^*$ is definable in $BC(\Sigma_1[\mathcal{N}])$ iff there is a finite subset $\mathcal{N}'$ of $\mathcal{N}$ and a number k such that, for every $u \in L$, $v \notin L$, Spoiler has a winning strategy in the single–round k–game for $\mathcal{N}'$ on u and v. □

3 The Crane Beach Conjecture

Intuitively, since numerical predicates can only talk about *positions* in strings, it seems that they can only help express properties that depend on certain (combinations of) letters appearing in certain (combinations of) positions. The Crane Beach Conjecture (named after the location of its first, flawed, proof) is an attempt to make that intuition precise.

3.1 Definition (Neutral letter)
Let $L \subseteq A^*$. A letter $e \in A$ is called *neutral* for L if for any $u, v \in A^*$ it holds that $uv \in L \iff uev \in L$. □

Thus membership in a language with a neutral letter cannot depend on the individual positions on which letters are: any letter can be moved away from any position by insertion or deletion of neutral letters. It seems therefore conceivable that for every such language, if it can be defined at all in first–order logic then it can be defined using the linear order as the only numerical relation.

3.2 Definition (Crane Beach Conjecture)
Let $\mathcal{N}$ be a set of numerical predicates. We say that *the Crane Beach conjecture is true for* $\mathcal{N}$, iff every language $L \in FO[\leq, \mathcal{N}]$ that has a neutral letter is also definable in $FO[\leq]$. □

It turns out that the conjecture is true for some sets of numerical predicates, but not for all. In fact, it fails for the set $\mathcal{N} = \{+, *\}$. This set of predicates is particularly important because $FO[+, *]$ corresponds to the most natural uniform version of the circuit complexity class AC^0 [BIS90].
Our counterexample to the Crane Beach conjecture makes use of the well-known but somewhat counterintuitive ability of $FO[+, *]$ formulas to *count* letters up to numbers polylogarithmic in the input size:

3.3 Definition (Definibility of Counting)
Let $f(n) \leq n$ be a nondecreasing function from $\mathbb{N}$ to $\mathbb{N}$. We say that a logical system can *count up to* $f(n)$ if there is a formula φ such that for every n and for every $w \in \{0, 1\}^n$,

$$w \models \varphi(c) \iff c \leq f(n) \wedge c = \#_1(w),$$

where $\#_1(w)$ is the number of ones in w.

We will need to consider two functions with similar notation. We write the base-two logarithm of n as $\lg n$, the k'th power of this logarithm as $(\lg n)^k$, and the k'th *iterated* logarithm as $\lg^{(k)} n$. For example, $\lg^{(2)} n$ is the same as $\lg(\lg n)$.

3.4 Proposition ([AB84, FKPS85, DGS86, WWY92])
The system $FO[+, *]$ can count up to $(\lg n)^k$ for any k. If $f(n) = (\lg n)^{\omega(1)}$, and $\mathcal{N}$ is any set of numerical predicates, then $FO[\leq, \mathcal{N}]$ cannot count up to $f(n)$.

3.5 Theorem
There is a language L with a neutral letter that is definable in $FO[+, *]$ but not in $FO[\leq]$.

Proof:
We define a language A on alphabet $\{0, 1, a\}$ as follows. For each positive integer k, A will contain a string consisting of the 2^k binary strings of length k, in order, separated by a's. The total length of the k'th string in A is thus $2^k(k + 1) - 1$. The first three strings in A are thus $0a1$, $00a01a10a11$, and

$$000a001a010a011a100a101a110a111.$$

Our desired language B has alphabet $\{0, 1, a, e\}$ and is simply the set of strings w over this alphabet such that the string obtained by deleting all the e's in w is in A. Clearly B has a neutral letter e, as inserting or deleting e's cannot affect membership in B. Clearly B is not regular, so it cannot be in $FO[\leq]$. It remains for us to prove:

3.6 Lemma
B is definable in $FO[+, *]$.

Proof:
We need to formulate a sentence of $FO[+, *]$ that will hold for a string exactly if it is in B, that is, exactly if its non-neutral letters form a string in A. Recall that a string w is in A exactly if for some number k, w consists of the 2^w binary strings of length k, in order, separated by a's.
Our sentence will assert the existence of a number k such that the input string, with e's removed, is the k'th string in the language A. Since the length of the k'th string in A is exponential in k, and a valid input string must be at least as long, any valid k must be at most $\lg n$. Therefore by Proposition 3.4, the system $FO[+, *]$ is able to count letters in any interval in the input string up to a limit of k.
We first assert that there are exactly k 0's and no 1's before the first a, exactly k 0's and 1's between each pair of a's, exactly k 1's (and no 0's) after the last a. It then remains to assert that each string of 0's and 1's between two a's is the successor of the previous one. To do this, we assert that for every position y containing a 0 or 1:

- If there is a position w left of y such that there is a 0 or 1 at y and exactly $k - 1$ 0's and 1's between w and y,
- Then w has the same letter as y *unless*
- x has the unique a between x and y, z has the next a to the right of x or is the rightmost position if there is no such a,
- w has 1, there are no 0's between w and x, y has 0, and there are no 1's between y and z, *or*
- w has 0, there are no 0's between w and x, y has 1, and there are no 0's between y and z.

This proves Lemma 3.6 and thus Theorem 3.5. □

Theorem 3.5 now follows immediately. □

The construction above crucially uses the fact that we can count up to $\lg n$ in $FO[+,*]$. We can strengthen the construction so that it provides a counterexample using only counting up to $\lg^{(m)} n$, the m times iterated logarithm of n. However, we do not yet know whether this strengthening is non-trivial — it may be that any set of numerical predicates that allows counting up to $\lg^{(m)} n$ also allows counting up to $\lg n$.

3.7 Proposition
If the system $FO[\leq,\mathcal{N}]$ can count up to $\lg^{(m)} n$ for some m, then there is a language L with a neutral letter that is definable in $FO[\leq,\mathcal{N}]$ but not in $FO[\leq]$.

Proof:
We must show that counting up to $\lg^{(m)} n$ suffices to provide a counterexample to the Crane Beach conjecture. We give the construction in some detail for $m = 2$, indicating how to generalize it to arbitrary values for m. Take the alphabet $\{a, b, 0, 1, e\}$ and for every k consider strings of the form $(b(0+1)^k(a(0+1)^k)^*)^*b$. Finally, add e as a neutral letter. a and b are used as markers, and we interpret the 0–1–substring between any two successive markers as the binary representation of some number between 0 and $2^k - 1$. If x is any position, we define $block(x)$ to be the interval between the two markers nearest x, and $num(x)$ to be the number represented by the 0–1 subsequence in $block(x)$. Using a formula that can count up to k and the construction from the proof of Theorem 3.5 we can write formulas expressing $num(x) = num(y)$ and $num(x) + 1 = num(y)$, respectively. We can now express easily that between every successive occurences of two b's each number from 0 to $2^k - 1$ is represented precisely once. In other words, this formula stipulates that the $\{a, 0, 1\}$–substring between two b's represent a permutation of the numbers $0, \ldots, 2^k - 1$. Finally, we write a formula that expresses that all permutations are represented. Altogether, our formula defines the set of those strings which consist of a sequence of permutations of the numbers $0, \ldots, 2^k-1$, for some k, containing every permutation at least once. In particular, every such string has length $\Omega(2^k!)$, whereas counting is only required up to $k = O(\lg\lg(2^k!))$.
To be more precise, the formula forces all permutations to be present as follows. It says that for every represented permutation π (starting, say, with a b at position p), and every pair of positions i, j within that permutation (i.e., $p < i < j < p'$, where p' is the smallest position $> p$ that carries a b), there is a permutation ρ (between b's at q and q', say) which is equal to π, except that $num(i)$ and $num(j)$ are swapped. In what follows we will use abbreviations $first(x)$ and $last(x)$ for formulas which express that x lies in the first, respectively last, block of some permutation; $next(x)$ will denote the first position in the block directly to the right of $block(x)$. Our formula for i and j now expresses the following for all r, s such that $p < r < p'$ and $q < s < q'$:

- $num(r) = num(s) \rightarrow num(next(r)) = num(next(s))$
  unless $last(r)$ or $\{num(r), num(next(r))\} \cap \{num(i), num(j)\} \neq \emptyset$
- $(num(r)=num(s) \wedge num(next(r))=num(i)) \rightarrow num(next(s))=num(j)$
- $(num(r)=num(s) \wedge num(next(r))=num(j)) \rightarrow num(next(s))=num(i)$
- $(num(s)=num(j) \wedge \neg last(s)) \rightarrow num(next(s))=num(next(i))$
- $(num(s) = num(i) \wedge \neg last(s)) \rightarrow num(next(s)) = num(next(j))$
- $(first(r) \wedge first(s) \wedge num(r) \neq num(i)) \rightarrow num(r) = num(s)$
- $(first(r) \wedge first(s) \wedge num(r) = num(i)) \rightarrow num(s) = num(j)$.

Thus we can construct the desired formula for $m = 2$.
We can then iterate this process, using an additional marker symbol c. The resulting formula stipulates that our string represent all permutations of all the permutations of the numbers $0, \ldots, 2^k - 1$. This will guarantee that string to be of length $\Omega(((2^k)!)!)$, etc. □

It is not difficult to code the languages above using only two non–neutral letters: just apply the homomorphism $\{a, b, 0, 1, e\}^* \rightarrow \{0, 1, e\}^*$ which maps e to e, a to 010, b to 0110, 0 to 01110, and 1 to 011110, for example. However, with only one non–neutral letter there is no way of defeating the conjecture.

3.8 Theorem
If $|A| = 2$ then for every set $\mathcal{N}$ of numerical predicates and every language $L \subseteq A^*$ with a neutral letter it holds that $L \in FO[\leq,\mathcal{N}] \Longrightarrow L \in FO[\leq]$.

Proof:
Let L be a language on $\{1, e\}$ with e as a neutral letter. Consider the set of numbers n such that 1^n is in L and 1^{n+1} is not. If this set is finite, it is easy to see that L is regular and definable in $FO[\leq]$. Otherwise, we will show that no family of unbounded fan-in circuits with constant depth and polynomial size can recognize L — it follows from [BIS90] that L is not definable in $FO[\leq,\mathcal{N}]$ for any $\mathcal{N}$.

For these particular values of n, any circuit deciding L on strings of length $2n$ would compute a symmetric function of the inputs saying yes on inputs with n 1's and no on inputs with $n+1$ ones. Following the construction of [FKPS85], a constant-depth poly-size combination of these circuits can be used to compute the parity function on inputs of this size. If the circuit deciding L had constant depth and polynomial size, then this new circuit would compute the parity function in AC^0 for infinitely many input sizes, violating [Ajt83, FSS84]. □

Since PARITY is a non–star-free regular language over $\{0,1\}^*$ and has a neutral letter, Theorem 3.8 implies the nonexpressibility of PARITY in first–order logic with arbitrary numerical predicates (i.e., AC^0). Note, however, that it directly uses the existing proofs of the nonexpressibility of PARITY to get this result.
On the other hand, the following special case of the Crane Beach conjecture can be proved directly:

3.9 Theorem
The Crane Beach conjecture holds for the set of all monadic relations.

Proof:
Let L be a language with a neutral letter that is not definable in $FO[\leq]$. This means that for any number of moves k there must be two strings $y \in L$ and $z \notin L$ such that the Duplicator wins the k-move game (using only $\leq$) on y and z. By adding neutral letters we can make y and z have the same length m.
Now let $\mathcal{N}$ be any monadic predicate. We will show that L is not definable in $FO[\leq, \mathcal{N}]$ as follows. We will use $\mathcal{N}$ to construct two strings $u \in L$ and $v \notin L$ from y and z by suitable padding with neutral letters. (The length of u and v will be a suitably large number n to be defined below.) Then we will show how the Duplicator can win the k-move game on u and v, with both $\leq$ and $\mathcal{N}$ as numerical predicates.
The predicate $\mathcal{N}$ may be regarded as a *coloring* of the input positions from 1 to n, with finitely many colors. If r and s are input positions, consider the colored string given by the interval from r to s, with each input position holding a neutral letter. For any two such strings, consider the k-move game with only $\leq$ as numerical predicate and the colors considered as the input. Let two strings be considered equivalent iff the Duplicator wins this game on them. Since the language defined by this game is regular, there are only a *finite number* of equivalence classes. We now define a colored undirected graph whose vertices are these n input positions and where the color of the edge from position r to position s represents the equivalence class of the colored string for that interval.
By the Erdos-Szekeres Theorem [ES35], as long as n is greater than m^d where d is the number of edge colors, there must be a *monochromatic path* in the graph of length at least m. We create u from y, and v from z, by placing the letters of the shorter strings in the locations given by the vertices of these path (the "special locations"), and making all other letters neutral. We must now explain how the Duplicator can win the game with $\leq$ and $\mathcal{N}$ on the strings u and v (the "Big Game").
The Duplicator will model the Big Game by a series of "small games", where she already has a winning strategy for each. One small game is played on the strings y and z using only $\leq$, and there is another small game (using $\leq$ and color only) for each interval between special locations. Whenever the Spoiler moves in the Big Game, the Duplicator translates this move into the y-z small game by moving to the position matching the next special position to the right. She also translates it into the small game for that interval. The Duplicator's reply in the Big Game is determined by her correct move in the y-z game, and her correct move in the special small game for that particular interval.
After k moves Delilah must win the original Small Game and all the interval Small Games, as she has made at most k moves in each. It is easy but tedious to look at the input predicates, order, equality, and position color in the Big Game and verify that Delilah has won that as well. □

We can use Theorem 3.9 to derive the following interesting generalization of the nonexpressibility of PARITY. But again, we do not get an *independent* proof of this fact because the existing proofs are used crucially to obtain the results in [BCST92].

3.10 Corollary
The Crane Beach conjecture holds for all regular languages. That is, for every set $\mathcal{N}$ of numerical predicates and every regular set L with a neutral letter it is true that that $L \in FO[\leq, \mathcal{N}] \Longrightarrow L \in FO[\leq]$.

Proof:
This follows from Theorem 3.9 and the fact, proven in [BCST92], that every regular language definable in $FO[\leq, \mathcal{N}]$ (using any set $\mathcal{N}$ of numerical predicates) is definable in $FO[\leq, \{mod_p \,/\, p \in \mathbb{N}\}]$, where $mod_p(i)$ is true iff $i \equiv 0 \mod p$. □

Although according to Theorem 3.9 the Crane Beach conjecture holds for the set of all unary relations, it is not true for all *binary* relations, since $FO[\leq, +, *] = FO[\leq, Bit]$, c.f., [Imm98]. In fact, it already fails for the set of all unary functions, or for the set of all linear orderings. This follows from the existence of a unary function $f : \mathbb{N} \to \mathbb{N}$ (see the proof of Theorem 3 in [Sch97]) and a set $\mathcal{O}$ of linear orderings (in fact, four order relations suffice, cf.[ScSc]) such that $FO[\leq, +, *] = FO[\leq, Bit] = FO[\leq, f] = FO[\leq, \mathcal{O}]$.

We can also consider special cases of the Crane Beach conjecture based on restrictions on the type of logical formulas allowed. For example, with arbitrary sets of numerical relations the conjecture does hold for Boolean combinations of Σ_1–formulas:

3.11 Theorem
Let $\mathcal{N}$ be a set of numerical predicates, and let L be a language with a neutral letter that is definable in the class $BC(\Sigma_1[\leq,\mathcal{N}])$. Then $L \in BC(\Sigma_1[\leq])$.

Proof:
We must show that for any set $\mathcal{N}$ of numerical predicates and any language L with a neutral letter, L is definable in $BC(\Sigma_1[\leq,\mathcal{N}])$ iff it is definable in $BC(\Sigma_1[\leq])$.
Using Theorem 2.2, we first show the proposition for the special case $\mathcal{N} = \{suc, \min, \max\}$, where suc is the successor relation $suc(n,m)$ iff $m = n{+}1$, $\langle w, n\rangle \models \min(n)$ iff $x{=}1$, and $\langle w, n\rangle \models \max(n)$ iff $n = |w|$.
Let e be the neutral letter, and assume that $L \notin BC(\Sigma_1[\leq])$. Then, for every k, there are strings $u \in L$, $v \notin L$ such that Duplicator wins the single–round k–game for $\leq$ on u, v. We can assume u and v to be of the same length m (if not, append $|v|{+}k$ e's to u and $|u|{+}k$ e's to v). We construct strings U from u and V from v such that $U \in L$, $V \notin L$, and Duplicator wins the single–round k–game for $\{\leq, suc, \min, \max\}$ on U, V. Then $L \notin BC(\Sigma_1[\leq, suc, \min, \max])$, which proves the assertion, by contraposition.
In order to construct U, insert $2k{-}1$ e's between each pair of adjacent positions in u, as well as at the beginning and the end of u. More precisely, $U = U_1 \cdots U_{m2k+2k-1}$, with $U_{j2k} = u_j$, and $U_{j2k+i} = e$, for any $j \leq m$, $i < 2k$. Similarly, we construct V from v. Since e is neutral, we have $U \in L, V \notin L$.
Assume that Spoiler chooses positions $a_1, \dots, a_k$ in U (the other case is symmetric). Some (possibly all, or none) of the U_{a_j} will be neutral letters, others will be from $A \setminus \{e\}$. For the sake of notational simplicity we will assume, without loss of generality, that $U_{a_1}, \dots, U_{a_q} \in A \setminus \{e\}$, and $U_{a_{q+1}} = \cdots = U_{a_k} = e$. Then each a_j with $j \leq q$ is of the form $s_j 2k$, for some $s_j \in \{1, \dots, m\}$. Now Duplicator simulates a move of Spoiler in the game for $\leq$ on u, v in which Spoiler pebbles $s_1, \dots, s_q$ on u, and finds her reply, $s'_1, \dots, s'_q$ on v, according to her winning strategy. She then sets, for each j from 1 through q, b_j to be $s'_j 2k$. Then for each $j, j' \leq q$ it holds that

- $b_j \neq b_{j'}{+}1$ and $a_j \neq a_{j'}{+}1$,
- $b_j \leq b_{j'} \iff a_j \leq a_{j'}$, and
- $V_{b_j} = v_{s'_j} = u_{s_j} = U_{a_j}$.

To complete this move, Duplicator has to define $b_{q+1}, \dots, b_k$ such that $V_{b_{q+1}} = \cdots = V_{b_k} = e$, and that for all $j, j' \leq k$

- $b_j \leq b_{j'} \iff a_j \leq a_{j'}$,
- $b_j = b_{j'}{+}1 \iff a_j = a_{j'}{+}1$, and
- $b_j = 1 \iff a_j = 1, b_j = |V| \iff a_j = |U|$.

Such $b_{q+1}, \dots, b_k$ can easily be found, since between any two different b_i, b_j with $i, j \leq q$, there are at least $2k{-}1$ positions p where $V_p = e$.

Now let $\mathcal{N}$ be an arbitrary finite set of numerical predicates and assume that $L \notin BC(\Sigma_1[\leq])$. From what we have just shown it follows that, for every k, we can find strings $u \in L$, $v \notin L$ of the same length m such that Duplicator has a winning strategy in the single–round $2k{+}2$–game for $\leq, suc, \min, \max$ on u, v. We want to construct strings U and V by inserting neutral letters into u and v, respectively, in such a way that the original letters of u and v are moved onto positions $i_1, \dots, i_m$ which are, in a certain sense, highly indistinguishable. To this end, we define, for every number n, a coloring of subsets of size $h \leq 2k$ of $\{1, \dots, n\}$. This coloring was inspired by the one used by Straubing in [Str01], in his proof of Theorem 8. There he used the following extension of Ramsey's theorem, which will also help us here:

Theorem Let $m, k, c_1, \dots, c_k > 0$, with $k \leq m$. Let n be sufficiently large as a function of m and the c's. If all h–element subsets of $\{1, \dots, n\}$, with $1 \leq h \leq k$, are colored from a set of c_h colors, then there exists an m–element subset T of $\{1, \dots, n\}$ such that for each h with $1 \leq h \leq k$ there exists a color κ_h such that all h–element subsets of T are colored κ_h. $\square$

Let $\mathcal{T} = \{\tau_1, \dots, \tau_q\}$ be the set of all atomic formulas over $\mathcal{N}, \leq$ on variables $x_1, \dots, x_k, y_1, \dots, y_h$. The $\mathcal{N}, \leq$*–type* of a tuple $r = (r_1, \dots, r_k) \in \{1, \dots, n\}^k$ with respect to a h–element set $S = \{p_1 < \cdots < p_h\}$, $\alpha(r, S)$, is the set of all those formulas of $\mathcal{T}$ that are satisfied when x_i is interpreted as r_i, and y_j as p_j, for $i \leq k$ and $j \leq h$.
We now color, for each number n and every $h \leq 2k$, every h–element set $S = \{p_1 < \cdots < p_h\} \subseteq \{1, \dots, n\}$ with the set of all those $\alpha \subseteq \mathcal{T}$ for which there is a k–tuple r over $\{1, \dots, n\}$ such that r has $\mathcal{N}$–type α with respect to S. Clearly, for every $h \leq 2k$ there is a fixed number of possible colors, independent of n. The extension of Ramsey's theorem stated above tells us that for large enough n we can find numbers $i_1 < \cdots < i_m \leq n$ such that, for every $h \leq 2k$, all h–element subsets of $\{i_1, \dots, i_m\}$ have the same color. We now insert neutral letters into u in such a way that in the resulting string U we have $U_{i_s} = u_s$, for $s = 1, \dots, m$, and $U_i = e$ for all $i \notin \{i_1, \dots, i_m\}$. In the

same way we construct V from v. Let us call $i_1, \ldots, i_m$ the *special positions*.
We now show that Duplicator has a winning strategy in the k–game for $\leq, \mathcal{N}$ on U, V. Assume that Spoiler chooses $a = a_1, \ldots, a_k$ in U (again, the other case is symmetric). Then Duplicator finds, for every a_j the next smallest special position i_{s_j}, i.e, $i_{s_j} \leq a_j < i_{s_j+1}$. Let $S = \{i_{s_j}, i_{s_j+1} / j = 1, \ldots, k\}$. Duplicator now simulates a move of Spoiler in the $2k+2$–game for $\leq, suc, \min, \max$ on u, v, in which Spoiler plays all the points s_j and s_j+1, for $j = 1, \ldots, k$ on u, as well as $\min$ and $\max$. Using her winning strategy in this game, Duplicator finds a reply with which she wins the game for $\leq, suc$. Therefore, we can safely call these points t_j, t_j+1, for $j = 1, \ldots, k$, and we know that $u_{s_j} = v_{t_j}$, for $j = 1, \ldots, k$. Let T be the set $\{i_{t_j}, i_{t_j+1} / j = 1, \ldots, k\}$. $|T| = |S| = h \leq 2k$, so S and T have the same colour, and this implies that there is a tuple $b = (b_1, \ldots, b_k)$ with the same $\mathcal{N}$–type as a, and with $\omega(b, T) = \omega(a, S)$. Duplicator now puts her pebbles on $b_1, \ldots, b_k$ in V. We have to check the winning conditions. By construction, $\alpha(a, S = \alpha(b, T)$. In particular, this implies that

- $(a_1, \ldots, a_k)$ and $(b_1, \ldots, b_k)$ have the same $\mathcal{N}$–type,
- $a_j \leq a_{j'} \iff b_j \leq b_{j'}$, for all j, j',
- if $a_j = i_{s_j}$ then $b_j = i_{t_j}$ hence $U_{a_j} = u_{s_j} = v_{t_j} = V_{b_j}$. If a_j is not of this form then $i_{s_j} < a_j < i_{s_j+1}$, consequently, $i_{t_j} < b_j < i_{t_j+1}$ and $U_{a_j} = V_{b_j} = e$.

□

As we have seen, with addition and multiplication first–order logic has enough expressive power to defeat the neutral letter. Addition alone is, in many ways much weaker than addition and multiplication together. For example, this is witnessed by the fact that the first–order theory of the natural numbers with $+$ and $*$ is undecidable, whereas Presburger arithmetic, the first–order theory of the natural numbers with addition only, can be decided using quantifier elimination. Also note that at least our technique for producing a counterexample cannot work with addition only, since it is well known (see, e.g., page 12 of [Lyn82]) that $FO[\leq, +]$ cannot count up to any non-constant function.
It is therefore more than conceivable that addition alone is too weak to make the conjecture fail, and we now show that this is indeed the case.

3.12 Theorem
Every language $L \in$ *FO*$[\leq, +]$ that has a neutral letter is definable in *FO*$[\leq]$.

As indicated in the introduction, this theorem follows from collapse results for first–order queries over finite databases (e.g., Theorem 5.5 in [BST99]). However the terminology in which these results are formulated is rather alien to our setting here, so we will instead use a recent collapse result on *infinite* databases in [LS01]. First, however, let us give an intuitive explanation of the main idea behind the proof.

For simplicity, we concentrate on 0–1–strings u, v of the same (large) size and discuss what Duplicator has to do in order to win the k–round $+$–game on u and v. Let A be the set of indices a for which $u_a = 1$, similarly, $B = \{b / v_b = 1\}$. As in previous proofs, we will work with a set Q of indistinguishable positions, and choose u and v such that $A, B \subseteq Q$.
Assume that, after $i{-}1$ rounds $a^{(1)}, \ldots, a^{(i-1)}$ have been played in u, and $b^{(1)}, \ldots, b^{(i-1)}$ in v. Let Spoiler choose some element $a^{(i)}$ in u. When choosing $b^{(i)}$ in v, Duplicator has to make sure that any Spoiler moves for the remaining $k{-}i$ rounds in one structure can be matched in the other. In particular, this means that any sum over the $a^{(j)}$ behaves in relation to A exactly as the corresponding sum over the $b^{(j)}$ behaves in relation to B. For instance, for any sets $J, J' \subseteq \{1, \ldots, i\}$, it should hold that there is some $a \in A$ that lies between $\sum_{j \in J} a^{(j)}$ and $\sum_{j' \in J'} a^{(j')}$ if and only if there is some $b \in B$ that lies between $\sum_{j \in J} b^{(j)}$ and $\sum_{j' \in J'} b^{(j')}$. But it is not enough to consider simple sums over previously played elements. Since with $O(r)$ additions it is possible to generate $s \cdot a^{(i)}$ from $a^{(i)}$, for any $s \leq 2^r$, we also have to consider linear combinations with coefficients as large as this. Furthermore, since Spoiler is allowed to choose either structure to move in each time, it is necessary to deal with even more complex linear combinations. One can only handle all these complications because, as the game progresses, the number of rounds left for Spoiler to do all these things decreases. This means, for instance, that the coefficients and the length of the linear combinations we have to consider decrease: after the last round, the only relevant linear combinations are simple additions of chosen elements.
All the technical details necessary to make this strategy work are worked out in [Lyn82] in order to prove that for each first–order formula with addition φ there is a set $Q \subseteq \mathbb{N}$ such that φ cannot distinguish between subsets of Q if they are of equal cardinality, or both large enough. Drawing on Lynch's theorem, in [LS01] the authors prove a theorem, which, specialised to our setting can be formulated as follows.

Theorem ([LS01], Theorem 3.2)
For every $k \in \mathbb{N}$ there exists a number $r(k) \in \mathbb{N}$ and an order–preserving mapping $q : \mathbb{N} \to \mathbb{N}$ such that, for every signature σ the following holds: If σ^U and σ^V are interpretations of σ over $\mathbb{N}$, and if $n, m \in \mathbb{N}$ with $\langle \mathbb{N}, \sigma^U, n \rangle \equiv^{\leq}_{r(k)} \langle \mathbb{N}, \sigma^V, m \rangle$, then $\langle \mathbb{N}, q(\sigma^U, n) \rangle \equiv^{+}_{k} \langle \mathbb{N}, q(\sigma^V, m) \rangle$. □

Here, $q(\sigma^U, n)$ is short for $\sigma^{q,U}, q(n)$, where $\sigma^{q,U} = \{R^{q,U} \,/\, R \in \sigma\}$, and $R^{q,U} = \{q(i) \,/\, i \in R^U\}$.

Proof of 3.12, using the above theorem:
Assume that $L \notin$ *FO*$[\leq]$, and let $u = u_1 \cdots u_n \in L$, $v = v_1 \ldots v_m \notin L$, such that $u \equiv^{\leq}_{r(k)} v$. We construct strings $U \in L$, $V \notin L$ from u and v, respectively, by inserting neutral letters in such a way that $U_{q(i)} = u_i$ and $V_{q(j)} = v_j$, for $i = 1, \ldots, n, j = 1, \ldots, m$, where q is as in the theorem. u and v define σ_A–interpretations σ^U_A and σ^V_A, respectively, and the winning strategy of Duplicator on u and v can easily be extended to $\langle \mathbb{N}, \sigma^U, n\rangle$ and $\langle \mathbb{N}, \sigma^V, m\rangle$: If Spoiler plays a position $a_i \leq n$ on $\langle \mathbb{N}, \sigma^U, n\rangle$, this corresponds to a move on u, and Duplicator can choose her answer according to her winning strategy on v. If Spoiler plays a position $a_i > n$ on $\langle \mathbb{N}, \sigma^U, n\rangle$, then Duplicator replies with $b_i := m+(a_i-n)$. (The case where Spoiler plays on $\langle \mathbb{N}, \sigma^V, n\rangle$ is completely symmetric.) Clearly, this defines a winning strategy for Duplicator. Application of the theorem above gives us a winning strategy for Duplicator in the k round game for $\{\leq, +\}$ on $\langle \mathbb{N}, q(\sigma^U, n)\rangle$ and $\langle \mathbb{N}, q(\sigma^V, m)\rangle$. From this, we obtain a winning strategy for Duplicator in the k round game for $\{\leq, +\}$ on U and V: Every move of Spoiler in U is translated into a move on $\langle \mathbb{N}, q(\sigma^U, n)\rangle$, and Duplicator's reply on $\langle \mathbb{N}, q(\sigma^V, m)\rangle$ is translated back into a move on V. The winning condition of Duplicator on $\langle \mathbb{N}, q(\sigma^U, n)\rangle$ and $\langle \mathbb{N}, q(\sigma^V, m)\rangle$ directly translates into the winning condition for Duplicator on U and V, thus proving that $U \equiv^+_k V$. □

4 Discussion

Much of the above can be generalised from strings to arbitrary relational structures over the natural (or real) numbers. This programme is pursued in [LS01]. With regard to the questions here, the following problems remain open.

- It would be very good to have a proof of Theorem 3.8 that does not rely on [Ajt83, FSS84]. However, since Theorem 3.8 implies the nonexpressibility of PARITY, we expect this to be very difficult.

- What is the status of the conjecture for $FO[\leq, *]$? There is a construction of Julia Robinson [Rob49] defining addition from multiplication and the successor operation, but in our context this only suffices to define addition on some numbers (those less than $n^{1/4}$) from multiplication and order on *all* numbers. We conjecture that some variant of this construction will suffice to disprove the Crane Beach conjecture for $FO[\leq, *]$, perhaps by showing it equivalent to $FO[\leq, +, *]$.

- Can we find a set of numerical predicates that allows us to count up to $\lg^{(m)} n$, but not to $\lg n$? What about counting up to even smaller functions? We conjecture that the Crane Beach conjecture is true of a system iff it cannot count beyond a constant.

- Within $FO[\leq, +, *]$, we can consider the subclasses of formulas based on the number of quantifier alternations. The lg–counting operation requires Σ_3, and the construction of the counter example adds a few more levels. This leaves a gap between the upper bound of something like Σ_5 in Theorem 3.5, and a lower bound of $BC(\Sigma_1)$ in Theorem 3.11. Since in $BC(\Sigma_2)$, counting is only possible up to a constant (cf., [FKPS85]), it is conceivable that the lower bound can be improved.

- Theorem 3.12 places limits on the power of a particular uniform circuit complexity class, an "addition-uniform" version of AC^0. Can we use these techniques to place limits on the power of more powerful uniform versions of AC^0 (without using the non-uniform lower bounds) or on addition-uniform versions of more powerful classes? This has been done for one such class, an addition-uniform version of LOGCFL, by Lautemann, McKenzie, Schwentick, and Vollmer [LMSV99].

- It would also be of interest to study the conjecture for certain extensions of FO, such as FO with unary counting quantifiers or FO with modulo counting quantifiers. These each have various versions depending on the numerical predicates available.

References

[AB84] M. Ajtai and M. Ben-Or. A theorem on probabilistic constant depth computations. *Proc. 16th ACM STOC* (1983), 471-474.

[Ajt83] M. Ajtai. Σ^1_1–formulae on finite structures. *Annals of Pure and Applied Logic*, 24:1–48, 1983.

[BCST92] D.A.M. Barrington, K. Compton, H. Straubing, and D. Thérien. Regular languages in NC^1. *Journal of Computer and Systems Sciences*, 44:478–499, 1992.

[BIS90] D.A.M. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *J. Comp. Syst. Sci.* **41:3** (1990), 274-306.

[BL00] M. Benedikt and L. Libkin. Expressive power: The finite case. In G. Kuper, L. Libkin, and J. Paredaens, editors, *Constraint Databases*, pages 55–87. Springer, 2000.

[BST99] O.V. Belegradek, A.P. Stolboushkin, and M.A. Taitslin. Extended order-generic queries. *Annals of Pure and Applied Logic*, 97:85–125, 1999.

[DGS86] L. Denenberg, Y. Gurevich, and S. Shelah. Definability by constant-depth polynomial-size circuits. *Inf. and Control* **70** (1986) 216-240.

[EFT94] H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Springer-Verlag, New York, 2nd edition, 1994.

[ES35] P. Erdös and G. Szekeres. A combinatorial problem in geometry. *Compositio Math.* **2** (1935), 464-470.

[FKPS85] R. Fagin, M. Klawe, N. Pippenger, and L.J. Stockmeyer. Bounded-depth polynomial size circuits for symmetric functions. *Theoretical Computer Science*, 36:239–250, 1985.

[FSS84] M.L. Furst, J.B. Saxe, and M. Sipser. Parity, circuits, and the polynomial–time hierarchy. *Mathematical Systems Theory*, 17:13–27, 1984.

[Imm98] N. Immerman. *Descriptive and Computational Complexity*. Springer–Verlag, New York, 1998.

[LMSV99] C. Lautemann, P. McKenzie, T. Schwentick, and H. Vollmer. The descriptive complexity approach to LOGCFL. In *STACS 1999: 16th Annual Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science **1563**, Springer Verlag (1999), 444-454.

[LS01] C. Lautemann and N. Schweikardt. An Ehrenfeucht–Fraïssé approach to collapse results for first–order queries over embedded databases. In *STACS 2001: 18th Annual Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science **2010**, Springer Verlag (2001), 455-466.

[Lyn82] J. F. Lynch. On sets of relations definable by addition. *Journal of Symbolic Logic*, 47:659–668, 1982.

[Rob49] J. Robinson. Definability and decision problems in arithmetic. *Journal of Symbolic Logic* **14** (1949), 98-114.

[Sch97] Th. Schwentick. Padding and the expressive power of existential second-order logics. In Mogens Nielson and Wolfgang Thomas, editors, *Proceedings of the Annual Conference of the European Association for Computer Science Logic*, Lecture Notes in Computer Science, pages 461–477, 1997.

[ScSc] N. Schweikardt and Th. Schwentick. In preparation.

[Str94] H. Straubing. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhäuser, 1994.

[Str01] H. Straubing. Languages defined with modular counting quantifiers. *Information and Computation*, 2001. To appear.

[WWY92] I. Wegener, N. Wurm, and S.-Z. Yi. Symmetric functions in AC^0 can be computed in constant depth and very small size. in *Boolean Function Complexity*, ed. M. Paterson, *London Mathematical Society Lecture Notes* **169** (1992), 129-139.