A NOTE ON THE EXPRESSIVE POWER OF LINEAR ORDERS

NICOLE SCHWEIKARDT [a] AND THOMAS SCHWENTICK [b]

[a] Institut für Informatik, Goethe-Universität Frankfurt am Main, Germany
 e-mail address: schweika@informatik.uni-frankfurt.de

[b] Lehrstuhl Informatik I, Technische Universität Dortmund, Germany
 e-mail address: thomas.schwentick@tu-dortmund.de

ABSTRACT. This article shows that there exist two particular linear orders such that first-order logic with these two linear orders has the same expressive power as first-order logic with the Bit-predicate FO(Bit). As a corollary we obtain that there also exists a built-in permutation such that first-order logic with a linear order and this permutation is as expressive as FO(Bit).

1. INTRODUCTION

There are various ways in which arithmetic (i.e., addition and multiplication) on finite structures can be encoded by other numerical predicates. The following theorem summarises the results from [2, 4, 7, 5, 3]; see [9] for a survey. Precise definitions are given in Section 2.

Theorem 1.1. *The following logics have the same expressive power (on the class of all finite structures):*

> FO(Bit),   FO($<, Bit$),   FO($+, \times$),   FO($+, Squares$),   FO($<, \times$),
> FO($<, +, \times, Exp, Bit, Squares$)

and each of them can describe exactly those string-languages that belong to DLOGTIME-*uniform* AC^0.

From Theorem 1.1 one might get the impression that relations with an involved arithmetical structure are necessary to encode arithmetic in a first-order fashion. Contradicting this intuition, we show in this article that arithmetic can also be encoded by two particular linear orders. More precisely, our main result exposes two linear orders $<, \prec$ such that FO($<, \prec$) has the same expressive power as FO(Bit). A weaker version of this result (with three further built-in orders) had been announced in [1, 8] (cf., Corollary 5.5(d) in [1] and Theorem 4.5(d) in [8]), both referring to an "unpublished manuscript on *MonadicNP* with built-in grid structures" by Schweikardt and Schwentick. This paper finally presents this result along with a detailed proof. As an easy corollary we also obtain a particular built-in permutation π such that FO($<, \pi$) has the same expressive power as FO(Bit).

Organisation. The remainder of this paper is structured as follows: In Section 2 our terminology is fixed. In Section 3 we introduce two linear orders $<, \prec_0$ and two unary predicates C, Q and show that $\mathrm{FO}(<, \prec_0, C, Q)$ is as expressive as $\mathrm{FO}(Bit)$. In Section 4 we show that $\mathrm{FO}(<, \prec_0)$ is strictly less expressive than $\mathrm{FO}(Bit)$; the proof utilises the so-called Crane Beach property that might be interesting in its own right. In Section 5 we show how $\prec_0$ and the unary predicates C, Q can be replaced by a single linear order $\prec$, and we show how to represent $\prec$ by a permutation π. Section 6 concludes the paper.

2. Preliminaries

We write $\mathbb{N}$ to denote the set $\{0, 1, 2, \ldots\}$ of all natural numbers. For each $n \in \mathbb{N}$ we write $[n]$ for the set $\{0, \ldots, n\}$ of all natural numbers of size up to n. We assume that the reader is familiar with *first-order logic* (FO, for short), cf., e.g., the textbook [6].

A *k-ary numerical predicate* is a relation $P \subseteq \mathbb{N}^k$. Particular numerical predicates that were mentioned in the introduction are

$$< := \{\, (a, b) \in \mathbb{N}^2 \ : \ a < b \,\},$$
$$+ := \{\, (a, b, c) \in \mathbb{N}^3 \ : \ a + b = c \,\},$$
$$\times := \{\, (a, b, c) \in \mathbb{N}^3 \ : \ a \cdot b = c \,\},$$

$Squares := \{\, a \in \mathbb{N} \ : \ \text{there exists a } b \in \mathbb{N} \text{ such that } a = b^2 \,\},$

$$Exp := \{\, (a, b, c) \in \mathbb{N}^3 \ : \ a^b = c \,\},$$

$Bit := \{\, (a, i) \in \mathbb{N}^2 \ : \ \text{the } i\text{-th Bit in the binary representation of } a \text{ is 1, i.e. } 2 \nmid \left\lfloor \frac{a}{2^i} \right\rfloor \,\}.$

A *k-ary built-in predicate* is a sequence $(R^n)_{n \in \mathbb{N}}$ of relations, where, for each $n \in \mathbb{N}$, $R^n \subseteq [n]^k$. Clearly, every k-ary numerical predicate P naturally induces a k-ary built-in predicate via $P^n := P \cap [n]^k$. Note that if P is a strict linear order on $\mathbb{N}$ (i.e., $P \subseteq \mathbb{N}^2$ is transitive, and for all $a, b \in \mathbb{N}$ we have either $a = b$ or $(a, b) \in P$ or $(b, a) \in P$), then P^n is a strict linear order on $[n]$, for every $n \in \mathbb{N}$.

3. Capturing FO(Bit) with Two Linear Orders and Two Unary Predicates

This section's aim is to present numerical predicates $\prec_0$, C, Q such that $\mathrm{FO}(<, \prec_0, C, Q)$ captures $\mathrm{FO}(Bit)$. Here, C and Q will be unary, and $\prec_0$ will be a linear order on $\mathbb{N}$.

The underlying idea is illustrated in Figure 1. We consider the elements of $\mathbb{N}$ to be distributed into a lower right triangular matrix with infinitely many columns and rows, where for every $i \in \mathbb{N}$, the i-th column consists of $i+1$ consecutive numbers, and the i-th row contains infinitely many numbers: The 0-th column consists of the number 0, the 1-st column consists of the numbers 1 and 2, the 2-nd column consists of the numbers 3, 4, and 5, and the i-th column consists all numbers z with $q_i \leq z \leq q_i + i$, where q_i denotes the smallest element in this column. I.e., $q_0 = 0$ and $q_i = q_{i-1} + i$, for all $i > 0$. Thus, $q_i = \frac{i(i+1)}{2}$, for all $i \in \mathbb{N}$.

We number the rows from bottom up and the columns from left to right, starting with 0. For each $x \in \mathbb{N}$, we write $c(x)$ and $r(x)$ to denote the *column number* and the *row number* of x in Figure 1, and we let $q(x)$ denote the bottom-most element in the same column as x. Thus,

$$c(x) = \max\{i \in \mathbb{N} : q_i \leq x\}, \qquad q(x) = q_{c(x)}, \qquad r(x) = x - q(x). \qquad (3.1)$$
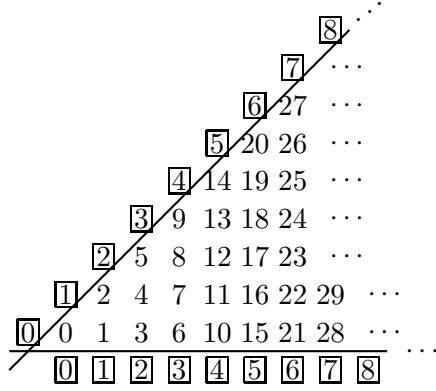
Figure 1: Illustration of columns and rows for the definition of $\prec_0$. Row numbers and column numbers are framed.

As an example, $c(13) = 4$, $q(13) = 10$, and $r(13) = 3$. Note that, by definition, we have

$$x = q(x) + r(x) \qquad \text{and} \qquad 0 \leq r(x) \leq c(x), \tag{3.2}$$

for every $x \in \mathbb{N}$. Clearly, all numbers x of the same column agree on $q(x)$. We thus sometimes call $q(x)$ the *q-value* of the column of a number x.

Of course, the standard order $<$ on $\mathbb{N}$ is just the bottom-to-top, left-to-right, column major order of this matrix. That is, for all $x, y \in \mathbb{N}$ we have

$$x < y \quad \Longleftrightarrow \quad c(x) < c(y) \quad \text{or} \quad \big(c(x) = c(y) \text{ and } r(x) < r(y) \big). \tag{3.3}$$

We define $\prec_0$ as the left-to-right, bottom-to-top, row major order. I.e., for all $x, y \in \mathbb{N}$ we let

$$x \prec_0 y \quad \Longleftrightarrow \quad r(x) < r(y) \quad \text{or} \quad \big(r(x) = r(y) \text{ and } c(x) < c(y) \big). \tag{3.4}$$

Thus, we have

$$0 \prec_0 1 \prec_0 3 \prec_0 6 \prec_0 10 \prec_0 \cdots \prec_0 2 \prec_0 4 \prec_0 7 \prec_0 \cdots \prec_0 5 \prec_0 8 \prec_0 \cdots \prec_0 9 \prec_0 \cdots.$$

We use the relations C and Q to induce binary strings on the columns of the matrix. The number encoded by the string induced by C on the i-th column (with the bottom-most element of this column representing the least significant bit) shall[1] be $i + 1$, and the number induced by Q on the i-th column shall be q_{i+1}. That is,

$$C := \{x \in \mathbb{N} \ : \ \text{bit } r(x) \text{ of the binary representation of } c(x)+1 \text{ is 1, i.e., } 2 \nmid \left\lfloor \tfrac{c(x)+1}{2^{r(x)}} \right\rfloor \},$$

$$Q := \{x \in \mathbb{N} \ : \ \text{bit } r(x) \text{ of the binary representation of } q_{c(x)+1} \text{ is 1, i.e., } 2 \nmid \left\lfloor \tfrac{q_{c(x)+1}}{2^{r(x)}} \right\rfloor \}.$$

See Figure 2 for an illustration of C and Q. As an example, the restriction of C to column 3 is the set $\{8\}$ (representing the bit string 100), and the restriction of Q to column 3 is the set $\{7, 9\}$ (representing the bit string 1010).

---

[1] Why we represent $i + 1$, respectively q_{i+1}, instead of i and q_i will be explained in Footnote 2.
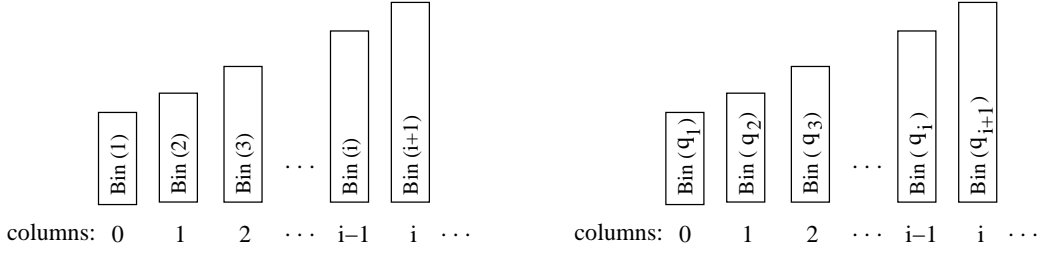
Figure 2: Illustration of the unary predicates C (left) and Q (right) assigning to each column i the binary representations $\text{Bin}(i{+}1)$ and $\text{Bin}(q_{i+1})$ of the numbers $i{+}1$ and q_{i+1}, respectively. The least significant bit of binary representations is in the bottommost row.

Note that, for every i, the i-th column contains sufficiently many elements to encode $q_{i+1} = \frac{(i+1)(i+2)}{2}$, since the i-th column has length $i{+}1$ and can thus encode binary representations of numbers of size up to $2^{i+1}{-}1 \geq q_{i+1}$.

The remainder of this section is devoted to the proof of the following theorem.

Theorem 3.1. $\text{FO}(<, \prec_0, C, Q)$ *has the same expressive power as* $\text{FO}(\textit{Bit})$.

Proof. That $\text{FO}(\textit{Bit})$ is at least as expressive as $\text{FO}(<, \prec_0, C, Q)$ is an immediate consequence of the following lemma.

Lemma 3.2. *There are* $\text{FO}(\textit{Bit})$*-formulas* $\varphi_<(x,y)$, $\varphi_{\prec_0}(x,y)$, $\varphi_C(x)$, $\varphi_Q(x)$ *such that, when evaluated in* $([n], \textit{Bit}^n)$ *for some* $n \in \mathbb{N}$, $\varphi_<(x,y)$ *expresses that* $x < y$, $\varphi_{\prec_0}(x,y)$ *expresses that* $x \prec_0 y$, $\varphi_C(x)$ *expresses that* $x \in C$, *and* $\varphi_Q(x)$ *expresses that* $x \in Q$.

Proof. The existence of the formula $\varphi_<(x,y)$ follows from Theorem 1.1. Using Theorem 1.1, it is straightforward to find $\text{FO}(\textit{Bit})$-formulas $\varphi_c(x,y)$, $\varphi_r(x,y)$, and $\varphi_q(x,y)$ which, when interpreted in $([n], \textit{Bit}^n)$, express that $c(x) = y$, $r(x) = y$, and $q(x) = y$, respectively. Using these formulas (and Theorem 1.1), it is an easy exercise to find formulas $\varphi_{\prec_0}(x,y)$, $\varphi_C(x)$, $\varphi_Q(x)$, expressing the statement of equation (3.4) and the definitions of the predicates C and Q. $\qquad\square$

To prove the opposite direction, we will construct an $\text{FO}(<, \prec_0, C, Q)$-formula that expresses the *Bit*-predicate. The construction of this formula will be established by a sequence of auxiliary formulas and lemmas.

For every $P \in \{<, \prec_0\}$ there are $\text{FO}(P)$-formulas $\varphi_{max,P}(x)$ and $\varphi_{succ,P}(x,y)$ expressing that x is the maximum element w.r.t. the linear order P, resp., that y is the successor of x w.r.t. P:

$$\varphi_{max,P}(x) \;:=\; \neg \exists z \; xPz \qquad \text{and} \qquad \varphi_{succ,P}(x,y) \;:=\; \big(xPy \wedge \neg \exists z (xPz \wedge zPy)\big).$$

For every $c \in \mathbb{N}$ there is an $\text{FO}(<)$-formula $\varphi_{=c}(x)$ expressing that x is interpreted with the natural number c:

$$\varphi_{=0}(x) \;:=\; \neg \exists z \; z < x \qquad \text{and} \qquad \varphi_{=c+1}(x) \;:=\; \exists z \big(\varphi_{=c}(z) \wedge \varphi_{succ,<}(z,x)\big).$$

To improve readability of formulas, we will henceforth often write

$$x = c, \quad x = max_P, \quad y = succ_P(x), \quad y = pred_P(x)$$

instead of $\varphi_{=c}(x)$, $\varphi_{max,P}(x)$, $\varphi_{succ,P}(x,y)$, $\varphi_{succ,P}(y,x)$. Furthermore, we will write

$$x \leq y \quad \text{and} \quad x \preceq_0 y$$

as shorthands for $(x < y \ \lor \ x = y)$ and $(x \prec_0 y \ \lor \ x = y)$.

Lemma 3.3. *There are formulas $\varphi_{same\text{-}col}(x,y)$, $\varphi_{same\text{-}row}(x,y)$, $\varphi_q(x,y)$, and $\varphi_{rc}(x,y)$ in* FO$(<, \prec_0)$ *such that, when evaluated in $([n], <^n, \prec_0^n)$ for some $n \in \mathbb{N}$,*

- *$\varphi_{same\text{-}col}(x,y)$ expresses that $c(x) = c(y)$, i.e., x is in the same column as y,*
- *$\varphi_{same\text{-}row}(x,y)$ expresses that $r(x) = r(y)$, i.e., x is in the same row as y,*
- *$\varphi_q(x,y)$ expresses that $q(x) = y$, i.e., y is the bottom-most element in the same column as x,*
- *$\varphi_{rc}(x,y)$ expresses that $r(x) = c(y)$, i.e., x's row-number is the same as y's column-number.*

Proof. Note that the bottom-most row consists of exactly those elements that are smaller than 2 w.r.t. $\prec_0$. Thus we can choose

$$\varphi_{\text{bot}}(x) \ := \ \forall z \ (z = 2 \to x \prec_0 z)$$

to express that x is an element in the bottom row.

Two elements x and y are in different columns iff there exists an element in the bottom row that lies between x and y w.r.t. $<$. Thus, we can choose

$$\varphi_{\text{same-col}}(x,y) \ := \ \neg \exists z \ \big(\varphi_{\text{bot}}(z) \land (x < z \leq y \ \lor \ y < z \leq x)\big).$$

Obviously, $q(x) = y$ iff y lies in the bottom row and in the same column as x. Thus, we can choose

$$\varphi_q(x,y) \ := \ (\ \varphi_{\text{bot}}(y) \land \varphi_{\text{same-col}}(x,y)\ ).$$

For $n \in \mathbb{N}$ we say that the *last column of $[n]$ is full* iff there is an $i \in \mathbb{N}$ such that $n = q_i + i$. Note that the last column of $[n]$ is full iff $n=0$ or the $<$-predecessor of n is also the $\prec_0$-predecessor of n and is different from 0. This can be expressed by the sentence

$$\varphi_{\text{last-col-full}} \ := \ \exists z \ \big(z = max_< \ \land \ \big(z = 0 \ \lor \ \exists y \ (y = pred_<(z) \ \land \ y = pred_{\prec_0}(z) \ \land \ \neg y=0)\big)\big).$$

An element x lies on the diagonal (i.e., $r(x) = c(x)$) iff either its $<$-successor lies in the bottom row, or x is the maximum element w.r.t. $<$ and the last column is full. Thus, we can choose

$$\varphi_{\text{diag}}(x) \ := \ \big(\exists y \ (y = succ_<(x) \land \varphi_{\text{bot}}(y)) \ \lor \ (x = max_< \ \land \ \varphi_{\text{last-col-full}})\big)$$

to express that x lies on the diagonal.

Two elements x and y lie in different rows iff there exists an element on the diagonal that lies between x and y w.r.t. $\prec_0$. Thus, we can choose

$$\varphi_{\text{same-row}}(x,y) \ := \ \neg \exists z \ \big(\varphi_{\text{diag}}(z) \land (x \prec_0 z \preceq_0 y \ \lor \ y \prec_0 z \preceq_0 x)\big).$$

Finally, for two elements x and y we have $r(x) = c(y)$ iff the diagonal element z that is in the same row as x, is in the same column as y. Thus we can choose

$$\varphi_{rc}(x,y) \ := \ \exists z \ (\varphi_{\text{diag}}(z) \land \varphi_{\text{same-row}}(x,z) \land \varphi_{\text{same-col}}(z,y)).$$

This completes the proof of Lemma 3.3. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

Lemma 3.4. *There are* $FO(<, \prec_0, C, Q)$*-formulas* $\varphi_{q,Bit,r}(x, u)$ *and* $\varphi_{r,Bit,r}(x, u)$ *which, when evaluated in* $([n], <^n, \prec_0^n)$ *for some* $n \in \mathbb{N}$*, express that the* $r(u)$*-th bit of the binary representation of* $q(x)$*, respectively, of* $r(x)$*, is 1.*

Proof. Note that if $x=0$, then $q(x)=0$, and thus the $r(u)$-th bit of the binary representation of $q(x)$ is 0. If $x > 0$, then the binary representation of the number $q(x)$ is given by relation Q on the elements of the column left to x's column.[2] Thus, the $r(u)$-th bit of $q(x)$ is 1 iff an element z with $r(z) = r(u)$ and $c(z) = c(x) - 1$ exists and belongs to Q. Therefore, we can choose $\varphi_{q,Bit,r}(x, u) :=$

$$\exists y \, \exists z \, \Big(\varphi_{\text{same-col}}(x, y) \wedge z = pred_{\prec_0}(y) \wedge \varphi_{\text{same-row}}(z, y) \wedge \varphi_{\text{same-row}}(z, u) \wedge Q(z) \Big).$$

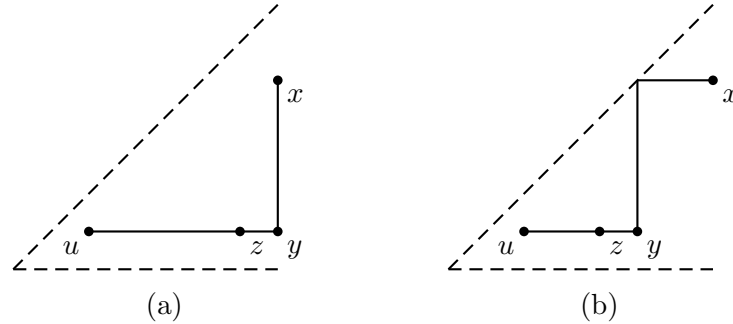The definition of $\varphi_{q,Bit,r}(x, u)$ is illustrated in Figure 3(a).

(a)   (b)

Figure 3: Illustration of the meaning of the variables used in (a) $\varphi_{q,Bit,r}(x, u)$ and (b) $\varphi_{r,Bit,r}(x, u)$.

Similarly, if $x=0$, then $r(x)=0$, and thus the $r(u)$-th bit of the binary representation of $r(x)$ is 0. If $x > 0$, then the binary representation of the number $r(x)$ is given by relation C on the elements of the *column* of number $r(x) - 1$. Thus, the $r(u)$-th bit of $r(x)$ is 1 iff an element z with $r(z) = r(u)$ and $c(z) = r(x) - 1$ exists and belongs to C. Therefore, we can choose $\varphi_{r,Bit,r}(x, u) :=$

$$\exists y \, \exists z \, \Big(\varphi_{rc}(x, y) \; \wedge \; z = pred_{\prec_0}(y) \; \wedge \; \varphi_{\text{same-row}}(z, y) \wedge \varphi_{\text{same-row}}(z, u) \wedge C(z) \Big).$$

The definition of $\varphi_{r,Bit,r}(x, u)$ is illustrated in Figure 3(b). $\qquad\square$

Lemma 3.5. *There is an* $FO(<, \prec_0, C, Q)$*-formula* $\varphi_{Bit,r}(x, z)$ *which, when evaluated in* $([n], <^n, \prec_0^n)$ *for some* $n \in \mathbb{N}$*, expresses that the* $r(z)$*-th bit of the binary representation of* x *is 1.*

Proof. Recall from equation (3.2) that $x = q(x)+r(x)$. We construct the formula $\varphi_{Bit,r}(x, z)$ in such a way that it expresses that the $r(z)$-th bit in the binary representation of $q(x)+r(x)$ is 1.

For this, we use an auxiliary formula $\varphi_{q+r,\text{carry},r}(x, z)$ which expresses that the addition of the binary representations of the numbers $q(x)$ and $r(x)$ produces a carry-bit to be added at the $r(z)$-th position. Note that when adding two binary numbers $a_\ell \cdots a_1 a_0$ and $b_\ell \cdots b_1 b_0$ (where the least significant bit is at the rightmost position), a carry-bit has to be added at

---

[2]Here, it is helpful that the $q(x)$ is represented in column $c(x) - 1$, as this column is guaranteed to be full.

position j iff there is a position $i < j$ such that $a_i = b_i = 1$ and for all positions k with $i < k < j$ at least one of the values a_k, b_k is 1. Thus, we can choose

$$\varphi_{q+r,\text{carry},r}(x, z) \ := \ \exists u \ \big(\varphi_{\text{same-col}}(u, z) \ \wedge \ u < z \ \wedge \ \varphi_{q,Bit,r}(x, u) \ \wedge \ \varphi_{r,Bit,r}(x, u)$$
$$\wedge \ \forall v \ (u < v < z \rightarrow (\varphi_{q,Bit,r}(x, v) \vee \varphi_{r,Bit,r}(x, v)))\big).$$

Note that the $r(z)$-th bit of the binary representation of $q(x) + r(x)$ is 1 if, and only if, either no carry-bit has to be added at position $r(z)$ and the $r(z)$-th bits of $q(x)$ and $r(x)$ are different, or a carry-bit has to be added at position $r(z)$ and the $r(z)$-th bits of $q(x)$ and $r(x)$ are the same. Thus, we can choose

$$\varphi_{Bit,r}(x, z) \ := \ \Big(\ \big(\neg\varphi_{q+r,\text{carry},r}(x, z) \ \wedge \ (\varphi_{q,Bit,r}(x, z) \leftrightarrow \neg\varphi_{r,Bit,r}(x, z))\big) \ \vee$$
$$\big(\ \varphi_{q+r,\text{carry},r}(x, z) \ \wedge \ (\varphi_{q,Bit,r}(x, z) \leftrightarrow \ \varphi_{r,Bit,r}(x, z))\big) \ \ \Big). \qquad \square$$

Lemma 3.6. *There is an* $\mathrm{FO}(<, \prec_0, C, Q)$-*formula* $\varphi_r(x, y)$ *which, when evaluated in* $([n], <^n, \prec_0^n, C^n, Q^n)$ *for some* $n \in \mathbb{N}$, *expresses that* $r(x) = y$.

Proof. Note that $r(x) = y$ iff the following is true: for every u, the $r(u)$-th bit of $r(x)$ is 1 iff the $r(u)$-th bit of y is 1. We can thus use the formulas $\varphi_{r,Bit,r}(x, z)$ and $\varphi_{Bit,r}(y, z)$ from the Lemmas 3.4 and 3.5 to define

$$\varphi_r(x, y) \ := \ \forall u \ (\varphi_{r,Bit,r}(x, u) \leftrightarrow \varphi_{Bit,r}(y, u)). \qquad \square$$

Now, the *Bit*-predicate can be expressed by the $\mathrm{FO}(<, \prec_0, C, Q)$-formula stating that there is a number u such that $r(u) = y$ and the $r(u)$-th bit of x is 1. I.e., we can choose

$$\varphi_{Bit}(x, y) \ := \ \exists u \ (\varphi_r(u, y) \wedge \varphi_{Bit,r}(x, u)).$$

This finally completes the proof of Theorem 3.1. $\qquad \square$

4. $\mathrm{FO}(<, \prec_0)$ Does Not Capture $\mathrm{FO}(Bit)$

In this section we show that the linear orders $<$ and $\prec_0$ alone are not sufficient to capture $\mathrm{FO}(Bit)$.

Theorem 4.1. $\mathrm{FO}(<, \prec_0)$ *is strictly less expressive than* $\mathrm{FO}(Bit)$.

Proof. Lemma 3.2 tells us that $\mathrm{FO}(<, \prec_0)$ is at most as expressive as $\mathrm{FO}(Bit)$. To show that $\mathrm{FO}(<, \prec_0)$ does not have the same expressive power as $\mathrm{FO}(Bit)$, we make use of the so-called *Crane Beach property* [1], which is defined as follows:

- Let ℓ be a list of built-in predicates. The logic $\mathrm{FO}(\ell)$ is said to have the *Crane Beach property* if the following is true: Every string-language L that is definable in $\mathrm{FO}(\ell)$ and that has a *neutral letter*, is also definable in $\mathrm{FO}(<)$. Here, a letter e is called *neutral for* L, if for all strings w_1, w_2 we have $w_1 w_2 \in L \iff w_1 e w_2 \in L$.

Clearly, $\mathrm{FO}(<)$ has the Crane Beach property by definition. From [1] we know that $\mathrm{FO}(Bit)$ does *not* have the *Crane Beach property*. In the remainder of this proof, we show that $\mathrm{FO}(<, \prec_0)$ has the Crane Beach property. This, in particular, will tell us that $\mathrm{FO}(<, \prec_0)$ does not have the same expressive power as $\mathrm{FO}(Bit)$.

The basic idea of the proof that $\mathrm{FO}(<, \prec_0)$ has the Crane Beach property is that the order $\prec_0$ is useless on structures in which all columns but the rightmost column contain only neutral letters. For the proof we follow the methodology of [1] and use *Ehrenfeucht-Fraïssé games* (EF-game, for short), cf., e.g., [6]. Let L be a language that is definable in $\mathrm{FO}(<, \prec_0)$ and that has a neutral letter. Let Σ be the alphabet of L (i.e., $L \subseteq \Sigma^*$), let $e \in \Sigma$ denote the neutral letter of L, and let k be the quantifier rank of the $\mathrm{FO}(<, \prec_0)$-formula that defines L. Our aim is to show that L is also definable in $\mathrm{FO}(<)$.

Towards a contradiction, let us assume that L is *not* definable in $\mathrm{FO}(<)$. Then, in particular, there are (non-empty) strings u and v such that $u \in L$, $v \notin L$, and the duplicator has a winning strategy in the $2k$-round EF-game on the structures

$$\mathcal{A} := \big([n^u], n^u, <^{n^u}, (Q_\sigma^u)_{\sigma \in \Sigma}\big) \qquad \text{and} \qquad \mathcal{B} := \big([n^v], n^v, <^{n^v}, (Q_\sigma^v)_{\sigma \in \Sigma}\big)$$

where, for any string w, we let $n^w := |w| - 1$. For each letter σ of Σ we let $Q_\sigma^w := \{i \in [n^w] : w_i = \sigma\}$, where $w = w_0 w_1 \cdots w_{n^w}$ with $w_i \in \Sigma$ for all $i \in [n^w]$. Henceforth, the $2k$-round EF-game on $\mathcal{A}$ and $\mathcal{B}$ will be called the *small game*.

Since L has neutral letter e, we can assume without loss of generality that u and v have the same length. (If not, we can proceed as in [1]: Append u with $2^{2k} + |v|$ neutral letters e, append v with $2^{2k} + |u|$ neutral letters e, and note that the duplicator has a winning strategy in the $2k$-round EF-game on the padded versions of $\mathcal{A}$ and $\mathcal{B}$.)

We use n to denote $n^u = n^v$, and we let $u = u_0 u_1 \cdots u_n$ and $v = v_0 v_1 \cdots v_n$ with $u_i, v_i \in \Sigma$. Now let $N := q_n + n$, and let U and V be strings of length $N+1$ of the form $e^* u$ and $e^* v$, respectively. In particular, we know that $U \in L$ and $V \notin L$. Note that $U = U_0 U_1 \cdots U_N$ is the string which, for all i with $0 \le i \le n$ carries letter u_i on position $q_n + i$, and which carries the neutral letter on all other positions; and analogously V is obtained from v. An illustration of how U and V are embedded in $([N], <^N, \prec_0^N)$ is given in Figure 4.
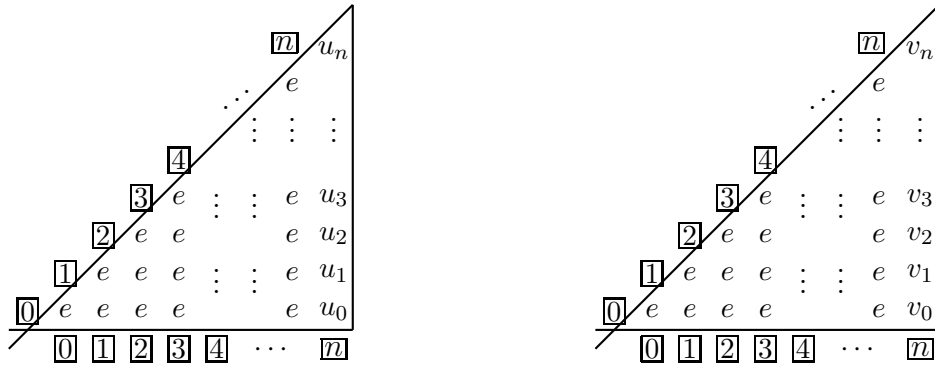

Figure 4: Illustration of the strings U (left) and V (right), embedded in $([N], <^N, \prec_0^N)$. Row and column numbers are framed.

We will now translate the duplicator's winning strategy in the small game into a winning strategy for the k-round EF-game on the structures

$$\mathfrak{A} := \big([N], <^N, \prec_0^N, (Q_\sigma^U)_{\sigma \in \Sigma}\big) \qquad \text{and} \qquad \mathfrak{B} := \big([N], <^N, \prec_0^N, (Q_\sigma^V)_{\sigma \in \Sigma}\big).$$

Henceforth, the EF-game on $\mathfrak{A}$ and $\mathfrak{B}$ will be called the *big game*. Note that N, U, and V were chosen in such a way that with respect to the triangular matrix illustrated in Figure 1

and restricted to the numbers in $[N]$, the strings u and v are in the rightmost column of $\mathfrak{A}$ and $\mathfrak{B}$.

To find a winning strategy for the big game, the duplicator in parallel plays (according to her given winning strategy) the small game and translates moves for the small game into moves for the big game. To be precise, for every round $i \in \{1,\ldots,k\}$ of the big game, the duplicator plays two rounds (namely, rounds $2i{-}1$ and $2i$) in the small game and proceeds as follows: If the spoiler chooses an element $\mathfrak{a}_i \in [N]$ in $\mathfrak{A}$, the duplicator lets a virtual spoiler choose $a_{2i-1} := c(\mathfrak{a}_i)$ and $a_{2i} := r(\mathfrak{a}_i)$ in the small game (thus, $\mathfrak{a}_i = q_{a_{2i-1}} + a_{2i}$), considers the duplicator's answer b_{2i-1} and b_{2i} following her winning strategy, and chooses $\mathfrak{b}_i := q_{b_{2i-1}} + b_{2i}$ as her answer in the big game (thus, $b_{2i-1} = c(\mathfrak{b}_i)$ and $b_{2i} = r(\mathfrak{b}_i)$). If the spoiler chooses an element $\mathfrak{b}_i$ in $\mathfrak{B}$, the duplicator's choice of $\mathfrak{a}_i$ in $\mathfrak{A}$ is determined in the analogous way.

After the k-th round of the big game, we know that the duplicator has won the small game, since she played according to her winning strategy. Thus, we have

(1) $u_{a_i} = v_{b_i},$ for all i with $1 \le i \le 2k$,
(2) $a_i < a_j \iff b_i < b_j,$ for all i,j with $1 \le i,j \le 2k$.

Our aim is to show that the duplicator has won the big game, i.e., that

(1') $U_{\mathfrak{a}_i} = V_{\mathfrak{b}_i},$ for all i with $1 \le i \le k$,
(2') $\mathfrak{a}_i < \mathfrak{a}_j \iff \mathfrak{b}_i < \mathfrak{b}_j,$ for all i,j with $1 \le i,j \le k$,
(3') $\mathfrak{a}_i \prec_0 \mathfrak{a}_j \iff \mathfrak{b}_i \prec_0 \mathfrak{b}_j,$ for all i,j with $1 \le i,j \le k$.

Concerning (1'), note that if $a_{2i-1} = n$ then $b_{2i-1} = n$ and $\mathfrak{a}_i = q_n + a_{2i}$, $\mathfrak{b}_i = q_n + b_{2i}$, $U_{\mathfrak{a}_i} = u_{a_{2i}}$, and $V_{\mathfrak{b}_i} = v_{b_{2i}}$. Thus, due to (1) we have $U_{\mathfrak{a}_i} = V_{\mathfrak{b}_i}$. Furthermore, if $a_{2i-1} < n$ then $b_{2i-1} < n$ and $\mathfrak{a}_i = q_{a_{2i-1}} + a_{2i} < q_n$ and $\mathfrak{b}_i = q_{b_{2i-1}} + b_{2i} < q_n$. Thus, $U_{\mathfrak{a}_i} = V_{\mathfrak{b}_i}$ is the neutral letter.

To obtain (3'), note that we have

$$
\begin{array}{lll}
\mathfrak{a}_i \prec_0 \mathfrak{a}_j & \iff & r(\mathfrak{a}_i) < r(\mathfrak{a}_j) \text{ or } (r(\mathfrak{a}_i) = r(\mathfrak{a}_j) \text{ and } c(\mathfrak{a}_i) < c(\mathfrak{a}_j)) \quad \text{(by equation (3.4))}\\
& \iff & a_{2i} < a_{2j} \text{ or } (a_{2i} = a_{2j} \text{ and } a_{2i-1} < a_{2j-1}) \quad \text{(by def. of } a_{2i-1}, a_{2i})\\
& \iff & b_{2i} < b_{2j} \text{ or } (b_{2i} = b_{2j} \text{ and } b_{2i-1} < b_{2j-1}) \quad \text{(by (2))}\\
& \iff & r(\mathfrak{b}_i) < r(\mathfrak{b}_j) \text{ or } (r(\mathfrak{b}_i) = r(\mathfrak{b}_j) \text{ and } c(\mathfrak{b}_i) < c(\mathfrak{b}_j)) \quad \text{(by def. of } \mathfrak{b}_i)\\
& \iff & \mathfrak{b}_i \prec_0 \mathfrak{b}_j \quad \text{(by equation (3.4))}.
\end{array}
$$

Note that (2') can be obtained in the same way, using equation (3.3).
In summary, the duplicator has won the big game. We hence obtain that the structures $\mathfrak{A}$ and $\mathfrak{B}$ satisfy the same first-order sentences of quantifier rank k. However, since $U \in L$ and $V \notin L$, this contradicts our assumption that L is definable by an $\mathrm{FO}(<, \prec_0)$-sentence of quantifier rank k. Thus, the proof of Theorem 4.1 is complete. $\square$

5. Capturing FO(Bit) with Two Linear Orders

In this section, we show that in Theorem 3.1 the numerical predicates $\prec_0, C, Q$ can be replaced by one particular linear order. The proof will immediately follow by combining Theorem 3.1 with the following Lemma 5.1.

If $R, R_1, \ldots, R_k$ are numerical predicates, we say that R is definable in $\mathrm{FO}(R_1, \ldots, R_k)$ *in every finite prefix* if there is an FO-formula that defines R^n on $([n], R_1^n, \ldots, R_k^n)$, for every $n \in \mathbb{N}$.

Lemma 5.1. *For all $k \geq 1$ and all unary relations $U_1, \ldots, U_k$ on $\mathbb{N}$, there is a linear order $\prec$ on $\mathbb{N}$, such that $\mathrm{FO}(<, \prec)$ is at least as expressive as $\mathrm{FO}(<, \prec_0, U_1, \ldots, U_k)$ on the class of finite structures. Furthermore, if $U_1, \ldots, U_k$ are $\mathrm{FO}(Bit)$-definable in every finite prefix then $\prec$ can be chosen $\mathrm{FO}(Bit)$-definable in every finite prefix as well.*

Proof. Within this proof, we will use the row numbers, column numbers, and q-numbers defined in equation (3.1). Our goal is to encode $\prec_0$ and the unary predicates into a single linear order $\prec$. To this end, the crucial observations are the following:

(1) For every number ℓ, the order $\prec_0$ can be recovered in a first-order fashion from $<$ and a sub-relation of $\prec_0$ that orders only every ℓ-th row (i.e., the rows $0, \ell, 2\ell, \ldots$).
(2) If ℓ is chosen large enough with respect to some number m, the remaining rows allow to encode m bits of information per element.

For the given number k, we will choose a sufficiently large number ℓ. All rows whose number is a multiple of ℓ will be called *backbone rows*, and the elements in these rows will be called *backbone elements*. In $\prec$, the backbone elements are ordered just as in $\prec_0$, and every backbone element is smaller w.r.t. $\prec$ than every non-backbone element. The number 2 is the smallest non-backbone element w.r.t. $\prec$. Thus, backbone elements can be identified by the $\mathrm{FO}(<, \prec)$-formula

$$\varphi_{\mathrm{backbone}}(x) \ := \ \forall y \ (y = 2 \to x \prec y).$$

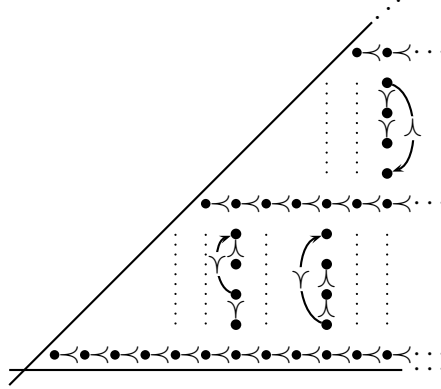Figure 5 gives an illustration of the overall shape of $\prec$.

Figure 5: Illustration of the definition of $\prec$ with $\ell = 5$. For the lower left interval, the corresponding permutation π is given by $\pi(1) = 1$, $\pi(2) = 2$, $\pi(3) = 4$, $\pi(4) = 3$, resulting in $u + 1 \prec u + 2 \prec u + 4 \prec u + 3$.

We call a set $\{u+1, \ldots, u+\ell-1\} \subseteq \mathbb{N}$ a *complete interval* if u and $u+\ell$ but none of the elements $u+1, \ldots, u+\ell-1$ are backbone elements. In this case, we call u *complete*. We say that u is *complete within* $[n]$ if u is complete and $u+\ell \in [n]$. Note that there is an $\mathrm{FO}(<, \prec)$-formula which, when evaluated in $([n], <^n, \prec^n)$ for some $n \in \mathbb{N}$, expresses that u is complete within $[n]$. This formula simply states that u is a backbone element, $u+\ell$ exists, and none of the elements $u+1, \ldots, u+\ell-1$ is a backbone element.

The elements of complete intervals will be ordered in such a way that the order $\prec$ on every complete interval $\{u+1, \ldots, u+\ell-1\}$ encodes the unary predicates on the elements $u, u+1, \ldots, u+3\ell-1$. Note that the encoding is sufficiently redundant to make sure that,

even though there are elements in intervals that are not complete within $[n]$ (i.e., elements close to the diagonal or close to n), the information whether x is an element of a set U_i is encoded in *some* complete interval, for every $x > q_{\ell+1}$.

To describe the order $\prec$ on each complete interval, we use the following notation. For every number $x \in \mathbb{N}$, let $B(x)$ be the bit-string of length k, where the i-th bit is 1 if and only if $x \in U_i$. For every complete element u we let $\vec{B}(u)$ be the bit-string of length $3k\ell$ with

$$\vec{B}(u) \ := \ B(u) \ B(u+1) \cdots B(u+3\ell-1).$$

We view each bit-string of length $3k\ell$ as the binary representation of a number from the set $\{0, 1, \ldots, 2^{3k\ell}-1\}$, and we write $b(u)$ to denote the according number associated with u by the bit-string $\vec{B}(u)$. We choose ℓ large enough such that $(\ell-1)! \geq 2^{3k\ell}$. Such an ℓ exists, since $n! = 2^{\Theta(n \log n)}$ (cf., Stirling's formula) and thus $(\ell-1)! = 2^{\Theta(\ell \log \ell)}$, and hence $(\ell-1)! \geq 2^{3k\ell}$ for all sufficiently large ℓ. Note that by our choice of ℓ we have $0 \leq b(u) \leq (\ell-1)! - 1$, for every complete element u.

Let $\pi_0, \ldots, \pi_{(\ell-1)!-1}$ be an enumeration of all permutations of the set $\{1, \ldots, \ell-1\}$. Now, the elements of every complete interval $\{u+1, \ldots, u+\ell-1\}$ are ordered in $\prec$ according to $\pi_{b(u)}$ via

$$u + \pi_{b(u)}(1) \quad \prec \quad u + \pi_{b(u)}(2) \quad \prec \quad \cdots \quad \prec \quad u + \pi_{b(u)}(\ell-1).$$

Note that it is straightforward to construct, for every permutation π of $\{1, \ldots, \ell-1\}$, an FO$(<, \prec)$-formula $\varphi_\pi(u)$ which, when evaluated in $([n], <^n, \prec^n)$ for some $n \in \mathbb{N}$, expresses that u is complete within $[n]$ and the interval $\{u+1, \ldots, u+\ell-1\}$ is ordered w.r.t. $\prec$ according to π.

How elements that do not belong to complete intervals, and how elements of different intervals, relate in $\prec$ does not matter for our proof. For concreteness, to fully fix $\prec$, we choose to let

$$x \prec y \iff x < y$$

for all natural numbers x, y for which the relationship has not yet been defined (neither directly nor transitively).

It remains to verify that

(a) the predicates $\prec_0, U_1, \ldots, U_k$ are FO$(<, \prec)$-definable in every finite prefix, and
(b) $\prec$ is FO(Bit)-definable in every finite prefix, provided that the unary relations $U_1, \ldots, U_k$ are FO(Bit)-definable in every finite prefix.

Towards (a), we can use the formulas $\varphi_\pi(u)$ to construct, for every $U_i \in \{U_1, \ldots, U_k\}$, an FO$(<, \prec)$-formula $\varphi_{U_i}(x)$ that, when evaluated in $([n], <^n, \prec^n)$ for some $n \in \mathbb{N}$, expresses that $U_i(x)$ holds. Note that either $x < q_{\ell+1}$ or $x = u + j$ where u is an element complete within $[n]$ and $0 \leq j < 3\ell$. In the former case, the information whether $U_i(x)$ holds can be "hard-coded" into an FO$(<, \prec)$-formula, as $q_{\ell+1}$ is a constant. In the latter case, the information whether $U_i(x)$ holds, can be inferred from the particular permutation π for which $\varphi_\pi(u)$ holds.

To express the predicate $\prec_0$ by an FO$(<, \prec)$-formula, we use that, for all $x, y \in \mathbb{N}$, we have $x \prec_0 y$ if, and only if, $x = u + i$ and $y = v + j$ where u, v are backbone elements and $0 \leq i, j < \ell$, such that the following is true:

(i) $r(u) < r(v)$, or
(ii) $r(u) = r(v)$ and either $i < j$ or ($i = j$ and $u \prec v$).

We note that, for backbone elements u and v, we have $r(u) < r(v)$ iff there is a backbone element w that is the rightmost element in its row, and $u \preceq w \prec v$. Furthermore, a backbone element w is rightmost in its row if either it is the maximal backbone element w.r.t. to $\prec$ or its $\prec$-successor w' is a backbone element on the diagonal. The latter can be recognized by the fact that w' and $w'+1$ are backbone elements. We can use this to obtain a formula $\varphi_{\prec_0}(x,y)$ expressing that $x \prec_0 y$. This concludes (a).

For proving (b) it suffices (due to Theorems 1.1 and 3.1) to show that $\prec$ is FO($Bit, \prec_0$, $U_1, \ldots, U_k$)-definable in every finite prefix. First of all, it is easy to identify the backbone rows. Furthermore, it is straightforward (though tedious) to infer $b(u)$ for a complete element u provided that $u + 3\ell - 1 \leq n$. To infer $b(u)$ for (the at most two) complete elements u with $u + 3\ell - 1 > n$, we use the fact that, for every FO(Bit)-formula $\psi(x)$ and every $i \in \mathbb{N}$ one can construct an FO(Bit)-sentence ψ_i such that $([n], Bit^n) \models \psi_i$ if and only if $([n+i], Bit^{n+i}) \models \psi(n+i)$. $\qquad\square$

From Theorem 3.1, Lemma 5.1 and the fact that the predicates C and Q are FO(Bit)-definable in every finite prefix, we immediately obtain the main result of this article.

Theorem 5.2. *There is a linear order $\prec$ on $\mathbb{N}$ such that* FO($<, \prec$) *has the same expressive power as* FO(Bit) *on the class of all finite structures.*

Using Theorem 5.2, one also obtains the analogous result, where the linear order $\prec$ is replaced by a built-in permutation $\pi = (\pi^n)_{n \in \mathbb{N}}$, that associates, with every $n \in \mathbb{N}$, a permutation π^n on the set $[n]$.

Corollary 5.3. *There is a built-in permutation π such that* FO($<, \pi$) *is as expressive as* FO(Bit).

Proof. Let $\prec$ be the linear order from Theorem 5.2. For every $n \in \mathbb{N}$ we define π^n as follows: For every $i \in [n]$ let $\pi^n(i)$ be the *index* of i w.r.t. $\prec$, i.e., $\pi^n(i) := |\{j \in [n] : j \prec i\}|$. Then, for all $i, j \in [n]$ the following is true:

$$i \prec^n j \quad \Longleftrightarrow \quad \pi^n(i) <^n \pi^n(j).$$

Hence, $\prec$ is definable by the FO($<, \pi$)-formula $\varphi_\prec(x,y) := \pi(x) < \pi(y)$. Therefore, due to Theorem 5.2, FO($<, \pi$) is at least as expressive as FO(Bit).

For the opposite direction, we need to find an FO(Bit)-formula $\varphi_{\text{index}}(x,y)$ which expresses that y is the index of x w.r.t. $\prec$, i.e., $y = \pi^n(x)$. Using our particular choice of the linear order $\prec$ fixed in the proof of Lemma 5.1, is not difficult to construct FO(Bit)-formulas which express that

- z is the total number of backbone elements,
- y is the number of backbone elements that are smaller w.r.t. $\prec$ than some backbone element x, and
- y' is the number of non-backbone elements that are smaller w.r.t. $<$ than some backbone element x'.

With the help of these formulas the formula $\varphi_{\text{index}}(x,y)$ can be constructed. To work out the details on the precise definition of this formula is a tedious, but easy exercise on FO(Bit)-definability. $\qquad\square$

Let us note that [10] already exposed a built-in unary function f such that $\mathrm{FO}(<, f)$ has the same expressive power as $\mathrm{FO}(Bit)$ (see the proof of Theorem 3 in [10] — the additional predicate for multiples of 8 can easily be encoded into f). The function f obtained there, however, is not a permutation.

6. Final Remarks

We have exposed two linear orders $<, \prec$ and a built-in permutation π such that both, $\mathrm{FO}(<, \prec)$ and $\mathrm{FO}(<, \pi)$ have the same expressive power as $\mathrm{FO}(Bit)$ (Theorem 5.2 and Corollary 5.3).

Of course, it can be debated whether linear orders are really "simpler" than addition and multiplication or the Bit predicate. Actually, this article precisely shows that, with respect to expressive power of first-order logic, they are not. However, in an intuitive sense, linear orders appear to be simpler, as they are just the transitive closure of a linear number of edges, and thus the structure of one linear order is more homogenous than, say, the structure of Bit. The characterisation given in Corollary 5.3 even shows that $\mathrm{FO}(Bit)$ can be captured by using $<$ and the linear number of edges provided by the built-in permutation π.

We note that there is no set M of *unary* built-in predicates such that $\mathrm{FO}(<, M)$ has at least the expressive power of $\mathrm{FO}(Bit)$. This is due to the fact that, according to [1], $\mathrm{FO}(<, M)$ has the Crane Beach property while $\mathrm{FO}(Bit)$ does not have this property.

Acknowledgement

References

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

[2] David A. Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within NC^1. *J. Comput. Syst. Sci.*, 41(3):274–306, 1990.

[3] J. H. Bennett. *On spectra*. PhD thesis, Princeton University, Princeton, NJ, 1962.

[4] Anuj Dawar, Kees Doets, Steven Lindell, and Scott Weinstein. Elementary properties of the finite ranks. *Math. Log. Q.*, 44:349–353, 1998.

[5] Troy Lee. Arithmetical definability over finite structures. *Math. Log. Q.*, 49(4):385–392, 2003.

[6] Leonid Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[7] James F. Lynch. Complexity classes and theories of finite models. *Math. Syst. Theory*, 15(2):127–144, 1982.

[8] Nicole Schweikardt. *On the Expressive Power of First-Order Logic with Built-In Predicates*. PhD thesis, Institute for Computer Science, Johannes Gutenberg-Universität Mainz, 2001. Published at Logos Verlag Berlin, 2002.

[9] Nicole Schweikardt. Arithmetic, first-order logic, and counting quantifiers. *ACM Trans. Comput. Log.*, 6(3):634–671, 2005.

[10] Thomas Schwentick. Padding and the expressive power of existential second-order logics. In *Proc. of 11th International Workshop on Computer Science Logic (CSL'97), Selected Papers*, volume 1414 of *Lecture Notes in Computer Science*, pages 461–477. Springer, 1997.