MONADIC LOGICAL DEFINABILITY OF PROBLEMS COMPUTABLE IN NONDETERMINISTIC LINEAR TIME

Etienne Grandjean, Frédéric Olive

Abstract. It is well-known that monadic second-order logic with linear order captures exactly regular languages. On the other hand, if addition is allowed, then J.F.Lynch has proved that existential monadic second-order logic captures at least all the languages in NTIME(n), and then expresses some NP-complete languages (e.g. knapsack problem), when they are naturally encoded as words in a fixed finite alphabet.

It seems that most combinatorial NP-complete problems (e.g. travelling salesman, colorability of a graph) do not belong to NTIME(n). But it has been proved that they do belong to NLIN (the similar class for RAMs).

In the present paper, we prove that existential monadic second-order logic with addition captures the class NLIN, so enlarging considerably the set of natural problems expressible in this logic. Moreover, we also prove that this logic still captures NLIN even if the first-order part of the second-order formulas is required to be $\forall^*\exists^*$, so improving the recent similar result of J.F.Lynch about NTIME(n).

In addition, we explicitly state that a graph problem is recognizable in nondeterministic linear time $O(n+e)$ (where n, e are the numbers of vertices and edges, respectively) if and only if it can be defined in existential second-order logic with unary functions and only one variable on the vertices-edges domain.

Key words. Computational complexity, monadic second-order logic, unary functions, finite model theory, descriptive complexity, nondeterminism, NP-complete problem, linear time, random access machine, binary and dyadic notations of integers.

Introduction

What is the relation between the computational complexity of a decision [1] problem and the logical complexity of the language required to describe it? This question was formulated first by Immerman Imm89 (see also Gur84, Gur86, Imm86, Var82).
In complexity theory any decision problem is identified with a language $\mathcal{L} \subseteq \Sigma^*$ (e.g. $\Sigma = \{0,1\}$). Solving the problem means deciding the associated language, *i.e.*, deciding for each word w whether it belongs to $\mathcal{L}$ or not.
Now, a word w on a finite alphabet is easily identified with a finite structure A_w. The most common and natural encoding (see Büc60, Tho82, Pin95, Lyn82, Lyn92) consists in identifying each word $w \in \{0,1\}^*$ with the finite structure $A_w = \langle n, X, succ \rangle$ defined by:

- $n = \{0, 1, \ldots, length(w) - 1\}$;
- $X \subseteq n$ and $\forall i \in n :\ X(i) \Leftrightarrow$ the i^{th} letter of w is 1 ;
- $succ$ is the successor relation on n.

Such an encoding being chosen, the purpose is the following: try to associate to every complexity class $\mathcal{C}$ a class $\mathcal{F}$ of formulas of a given logical language, in such a way that the following holds: a language $\mathcal{L} \subseteq \Sigma^*$ is in $\mathcal{C}$ iff there exists a sentence $\Phi \in \mathcal{F}$ such that:

$$\forall w \in \Sigma^* \ : \ w \in \mathcal{L} \Leftrightarrow A_w \models \Phi.$$

If this is true for $\mathcal{C}$ and $\mathcal{F}$ we write $\mathcal{C} = \mathcal{F}$.

Among the results which were proved in that framework, let us mention those of Büchi and Fagin, often quoted in this way:

> *Regular Languages* = *monadic SO* = *monadic* $SO(\exists)$ (Büc60, Tho82, Pin95) and $NP = SO(\exists)$ (Fag74, Lyn82),

We would expect this correspondence between computational complexity and logical definability to allow us: (i) to use the flexibility of complexity-theory tools (models of computation for instance) to get results on logical definability; (ii) to export complexity questions into the field of logical definability. An important example of the second item is the following: in order to prove that a given problem belongs to a given complexity class, it is enough to exhibit

[1] This question is also studied for optimization problems. See PaYa91 or KoTh90 for instance.

an algorithm of that complexity which decides this problem. But of course, there is not guarantee that this algorithm is the "best" one. In other words, the difficulty is to obtain lower bounds of complexity.

In Lyn92, Lynch focuses on this precise question. He writes: "there are hundreds of known NP-complete problems, but until recently, not one of them had a provable nontrivial lower bound." (There is a noteworthy exception with the problem "Reduction of Incompletely Specified Automata"(RISA), which has been proved nonsolvable in deterministic linear time in Gra90.) This is the reason why Lynch wants to compare the class NTIME(n) (i.e. the class of languages recognized in nondeterministic linear time on multitape Turing machines) with some class of logical formulas $\mathcal{F}$. He hopes then to obtain results such as "$\Pi \notin$ NTIME(n)", for some natural NP-complete problems Π, by proving their nondefinability by formulas of $\mathcal{F}$. Actually, the inclusion "NTIME(n) $\subseteq \mathcal{F}$" is enough to get such a result, because it still allows the implication "Π nondefinable by $\mathcal{F} \Rightarrow \Pi \notin$ NTIME(n)".

The main result of Lyn82, Lyn92 may be stated as follow:

Theorem (Lynch). *Let $\mathcal{L} \subseteq \{0,1\}^*$ belong to NTIME(n). Then there exists a first-order prenex formula Φ on a signature made up, in addition to the predicate symbols X, succ, plus of respective arity 1, 2, 3, of monadic predicate symbols $\overline{U} = (U_1, \ldots, U_s)$, such that:*

1. *the quantifier prefix of Φ has the form $\forall^*\exists^*$;*

2. $\forall w \in \{0,1\}^n : \ w \in \mathcal{L} \Leftrightarrow \langle n, X, succ, plus\rangle \models \exists \overline{U} \Phi(X, \overline{U}, succ, plus)$,

where succ, plus and X are interpreted on $n = \{0, 1, \ldots, n-1\}$ by the predefined relations: $X(i)$ iff the i^{th} letter of w is 1 ; $succ(i,j)$ iff $j = i+1$; $plus(i,j,k)$ iff $i + j = k$.

As Lynch wrote in Lyn92, "if one attempts to prove an undefinibility result, it is worthwile to restrict the form of the sentences as much as possible, with the hope that it will make the proof easier". In particular, this motivates Lynch's interest and ours in the quantifier prefix $\forall^*\exists^*$.

Concluding his paper (Lyn92), Lynch asks: "Can the (above) theorem be extended to random access models of computation, where the memory elements can be read and written in any order" ? The question is justified by the specificity of his proof, which uses a discrete analogue of the Intermediate Value Theorem for Turing machines, that is: if a tape head is at the position c at the time t and at c' at the time $t' > t$, then for every position c'' between

c and c', there is some time t'' between t and t' such that the head is at c'' at time t''. Of course, there is not equivalent result for RAMs.

The main purpose of this paper is to prove that Lynch's question has nevertheless a positive answer. More precisely, we show that the previous theorem remains true when replacing "NTIME(n)" by "NLIN", where NLIN is a complexity class (including NTIME(n)) elaborated by Grandjean (Gra94, Gra94b, Gra96) to formalize nondeterministic linear time on RAMs. The definition of this class will be recalled in 1.1 and studied in section 2. But let us before go back over some steps of the linear time formalization story.

Although it is commonly mentioned by algorithm designers, the linear time complexity notion is especially hard to formalize, because of its great sensitivity to models of computing and to problems encodings used to describe it. This lack of robustness, apparently inherent in linear time, leads Gurevich and Shelah (GuSh89) and Grädel (Grl90) to define two "robust closures" of linear time. Previously, Schnorr (Sch78) had similarly defined "quasilinear time" (that is time $O(n(\log n)^{O(1)})$ for some Turing machine) and proved that many NP-complete problems belong to nondeterministic quasilinear time and are complete for this class. Those authors defined extensions of linear time because, as GuSh89 explains: "It is possible that there is no universal notion of linear time and different versions of linear time are appropriate to different applications."

In Gra94, Gra94b, Gra96 Grandjean adopts the opposite point of view: he defines and justifies a unified, robust and powerful notion of linear time, both in deterministic and nondeterministic cases, with the classes DLIN and NLIN. Moreover, in Gra88, Gra90, Gra94, he gives a logical characterization of NLIN by second-order formulas written with $<$, 0 and unary function symbols. This characterization will be our main tool to prove that existential monadic second-order logic captures NLIN. Such a result appreciably extends the Lynch's, since NTIME(n) $\subseteq$ NLIN, and since all the twenty one Karp's NP-complete problems (Kar72) have been proved as belonging to NLIN (Gra94b, Gra96), whereas only two of them (`Knapsack` and `Partition`) are known as belonging to NTIME(n).

A preliminary and abbridged form of this paper appeared as a conference proceeding in GrOl95: neither the NLIN logical characterization with functions is proved in it, nor the precise proof that the first-order prefix of the existential monadic second-order NLIN characterization was $\forall^*\exists^*$ (main result) is given. The present paper is self-contained and exhibits complete proofs of the stated results. Moreover, beside the main result which gives a logical characterization of the NLIN class of languages (*i.e.* sets of words), we also prove here new precise results (Theorems 1.8, 2.7 and 2.9) which characterize the NLIN class of relational structure problems (e.g. graph problems) when the input is encoded as a unary structure.

This encoding and hence the associated logical characterization, may seem more natural than the similar one for NLIN languages, since it is exact (a relational structure problem is in NLIN *if and only if* it is definable in our logic) and since the encoding is simpler and respects the main complexity parameters of the input structure, e.g. a graph $G = (V, E)$ is encoded by a new unary structure of cardinality $|V| + |E|$. (Note that this is similar to the encoding of a graph in Cou96).

1. NLIN characterization and sketch of proof of the main result

1.1. Nondeterministic Random Access Machine and NLIN definition for sets of functions. A nondeterministic Random Access Machine (NRAM) $\mathcal{R}$ has a sequence of registers $R_0, R_1, \ldots$ and a program, that is a sequence of instructions $\mathcal{I}(1), \mathcal{I}(2), \ldots, \mathcal{I}(r)$ of the following types (where i, j, i_0, i_1 are integers and $1 \leq i_0, i_1 \leq r$):

(1)	$R_i := j$	(7)	*guess* (R_i)
(2)	$R_i := R_j$	(8)	*goto* $\mathcal{I}(i_0)$ *or* $\mathcal{I}(i_1)$
(3)	$R_i := R_j + 1$	(9)	*If* $R_i = R_j$ *then goto* $\mathcal{I}(i_0)$ *else goto* $\mathcal{I}(i_1)$
(4)	$R_i := R_j \dot{-} 1$	(10)	*accept*
(5)	$R_i := R_{R_j}$	(11)	*reject*
(6)	$R_{R_i} := R_j$		

Comments.

- $i \dot{-} j =_{def} max\{i - j, 0\}$.
- R_{R_i} denotes the register pointed to by the register R_i (*i.e.* the address of R_{R_i} is the content of R_i).
- (7) and (8) are nondeterministic instructions: *guess* (R_i) consists in storing in R_i the content of any register of $\mathcal{R}$.

- Our NRAM has no input instruction : the input is present in the first registers $\mathcal{R}$ at the beginning of the computation. More precisely, an input is a function $f : m \to m - \{0\}$ (where $m > 1$ is an integer depending on f) and the initial content of register R_i is $f(i)$ if $i < m$ and is 0 if $i \geq m$.

Definition 1.1. *An NRAM $\mathcal{R}$ accepts an input $f : m \to m - \{0\}$ in time $O(m)$ if there is an accepting computation of $\mathcal{R}$ on input f that uses $O(m)$ time (uniform cost mesure).*

Remark 1.2. *Clearly, such a computation only uses integers $O(m)$.*

Definition 1.3. *An NRAM $\mathcal{R}$ recognizes a set $\mathcal{F}$ of functions f ($f : m \to m - \{0\}$ for some $m > 1$ depending on f) in time $O(m)$ if:*

1. *$\mathcal{R}$ accepts f iff $f \in \mathcal{F}$ and*
2. *$\mathcal{R}$ accepts $f \in \mathcal{F}$ in time $O(m)$.*

If there exists some NRAM $\mathcal{R}$ meeting 1 and 2, we say that the set $\mathcal{F}$ belongs to the class NLIN.

Remark 1.4.

- *Having one-variable functions as input is rather natural: any configuration of a RAM - in particular the input configuration - can be regarded as a function f: adresses$\to$registers contents. Moreover, the input of natural problems (e.g. graph problems) are naturally encoded by such unary structures (for more details, see remark 2.3 p. 17).*
- *The notion of time complexity we define with NRAMs is very robust: it does not depend on the precise set of allowed NRAM instructions. For example, the NLIN class does not change if addition and subtraction are allowed with integers (i.e. addresses and contents of registers) $O(m^{O(1)})$ (see Gra94, Gra94b, Gra96).*

1.2. Reading decomposition of words and NLIN definition for languages. The main result of this paper is a generalization of Lynch's result that concern languages, *i.e.* sets of words over a finite alphabet Σ.

From now on, we assume that Σ is the dyadic alphabet ($\Sigma = \{1, 2\}$). We use $\{1, 2\}$ intead of $\{0, 1\}$ for technical convenience only (see remark 3.1 p. 20 for details). Of course our main result still holds for alphabet $\{0, 1\}$. We denote by k a fixed integer greater than or equal to 2. In this section, we describe a decomposition of the words of Σ^* suited to the NRAM definition given above. This decomposition, called *reading decomposition*, will allow to encode every word $w \in \Sigma^*$ as a finite structure of the $\langle m, f, <, 0\rangle$ type, where f denotes a unary function on $m = \{0, 1, \ldots, m-1\}$ and $<$ is the usual order on the domain m. This encoding is used in an essential manner in the logical characterization of NLIN languages stated in Gra94, Gra94b and proved in section 2.

Let $w \in \Sigma^*$ and let us denote $n = length(w)$, $l = \lceil \frac{\log(n+1)}{k} \rceil$ (in this paper, log denote the logarithm in base 2) and $m = \lceil \frac{n}{l} \rceil$. Then we can break down w into m words $w_0, w_1, \ldots, w_{m-1}$, all of which have length l exactly, except the last one, which may be of length less than or equal to l. We have $w = w_0 \frown w_1 \frown \ldots \frown w_{m-1}$, and if w_i^d denotes the integer whose dyadic writing is w_i, it is easy to show that

- $m = \Theta(n/\log n)$ and
- $w_i^d < m$ for large enough n.

We can now identify the word $w \in \{1, 2\}^*$, with reading decomposition $w = w_0 \frown w_1 \frown \ldots \frown w_{m-1}$, with the finite structure $\langle m, f_w, <, 0\rangle$ defined by:

- $m = \{0, 1, \ldots, m-1\}$;
- $f_w : \begin{cases} m \rightarrow m - \{0\} \\ i \mapsto f_w(i) = w_i^d \end{cases}$
- $<$ is the natural linear order on m and 0 is the "real" 0 of m.

REMARK 1.5. *Compare this with the usual identification of a word $w \in \{0, 1\}^n$ with the finite structure $\langle n, X, <\rangle$, where X is the unary relation on n such that $X(i)$ holds iff the i^{th} letter of w is 1 (see Lyn82 or Tho82): the "reading decomposition" $\langle m, f_w\rangle$ is, in some sense, a generalization of this representation.*

DEFINITION 1.6. *A language $\mathcal{L} \subseteq \{1, 2\}^*$ is in NLIN if the set of its associated unary functions $\{f_w,\ w \in \mathcal{L}\}$ belongs to NLIN.*

REMARK 1.7.

- *As shown in Gra94b, Gra96, the NLIN class of languages is very robust: in particular, it does not depend on the input process, e.g. the precise value of integer k is immaterial. This justifies the reading decomposition (of an input word) which looks unnatural at the first glance.*

- *As proved in Gra94b, Gra96, the main interests of our NLIN class are he following:*
  - *NLIN includes most of the natural NP-complete languages;*
  - *NLIN contains some natural NLIN-complete problems (for DTIME(n)-computable reductions), e.g. "Reduction of Incompletly Specified Automata" (RISA).*

1.3. NLIN characterization and main result.

Notations In the following, we denote by $FO(\forall^i\exists^j)$ the class of first-order formulas in prenex form, the quantifier prefix of which has the pattern $\forall^i\exists^j$. We omit the exponent i or j when it equals 1. $FO(\forall^*\exists^*)$ is $\bigcup_{i,j} FO(\forall^i\exists^j)$ and $FO(\forall\exists^*)$ is $\bigcup_j FO(\forall\exists^j)$. $FO(qf)$ is the class of quantifier-free first-order formulas. These notations naturally extend to define any class of formulas logically equivalent to formulas of the concerned class.

With the same notations as in 1.2, the NLIN characterization, essentially proved in Gra88, Gra90, Gra94, may be written:

Theorem 1.8. (Logical characterization of NLIN sets of functions). *A set $\mathcal{F}$ of unary functions is in NLIN iff there exists a formula $\Psi \in FO(qf)$ with only one first-order variable on signature $\{f, g_1, \ldots, g_p, <, 0\}$, where f and the g_i's denote unary function symbols, $<$ is a binary predicate symbol and 0 is a constant symbol, such that for every function $f : m \to m - \{0\}$:*

$$f \in \mathcal{F} \text{ iff } \langle m, f, <, 0\rangle \models \exists g_1 \ldots \exists g_p \forall x\, \Psi(x, f, \overline{g}, <, 0) \qquad (1.1)$$

Corollary 1.9. (Logical characterization of NLIN languages). *A language $\mathcal{L} \subseteq \{1,2\}^*$ is in NLIN iff there exists a formula $\Psi \in FO(wq)$with only one first-order variable on signature $\{f, g_1, \ldots, g_p, <, 0\}$, where f and the g_i's denote unary function symbols, $<$ is a binary predicate symbol and 0 is a constant symbol, such that for every word $w \in \{1,2\}^*$:*

$$w \in \mathcal{L} \text{ iff } \langle m, f_w, <, 0\rangle \models \exists g_1 \ldots \exists g_p \forall x\, \Psi(x, f, \overline{g}, <, 0) \qquad (1.2)$$

where $\langle m, f_w, <, 0\rangle$ is the finite structure associated to w as above.

This exact characterization of NLIN languages will allow to prove our main result, that is:

MAIN THEOREM. *If a language $\mathcal{L} \subseteq \{1,2\}^*$ is in NLIN, then there exists a sentence $\Theta \in FO(\forall^*\exists^*)$ on signature $\{X, U_1, \ldots, U_s, +\}$, where X and the U_i's are unary predicate symbols, $+$ is a ternary predicate symbol, such that for every word $w \in \{1,2\}^*$:*

$$w \in \mathcal{L} \;\text{ iff }\; \langle n, X, + \rangle \models \exists U_1 \ldots \exists U_s \, \Theta(X, \overline{U}, +) \qquad (1.3)$$

where $+$ is the predefined addition on n and X is the predicate naturally encoding w on n, i.e. defined by: $\forall i < n : X(i) \Leftrightarrow$ the letter of index i of w is 2.

1.4. A simplified view of the proof of the Main Theorem.

• We know that each positive integer x may be encoded by a word of $\{0,1\}^*$ of length $\lfloor \log(x) + 1 \rfloor$ (binary representation of integers). In particular, if E is a finite set of integers, we can identify every $x \in E$ with a word $B(x) \in \{0,1\}^*$ of length $L = Max\{\lfloor \log(x) + 1 \rfloor, x \in E\}$ exactly, by padding with 0's the binary representations of the elements of E. Equality between two integers of E is equivalent to equality of their representative words. Namely, if for $x \in E$ we denote $B(x)_t$ the bit of $B(x)$ of rank t ($0 \le t < L$ with convention: the bit of rank 0 is the least significant bit), we have for all $x, y \in E$:

$$x = y \;\text{ iff }\; (\forall t < L)(B(x)_t = B(y)_t).$$

• This encoding of integers by fixed length words can be extended to functions in the following way: since every $x < m$ is encoded by $B(x) \in \{0,1\}^L$, where $L = \lfloor \log(m) + 1 \rfloor$, a function $f : m \to m$ may be identified with the word:

$$B^f = B(f(0))^\frown B(f(1))^\frown \ldots ^\frown B(f(m-1)) \in \{0,1\}^{mL}.$$

If the t^{th} bit of B^f is denoted $(B^f)_t$, we have: $\forall f, g : m \to m$, $\forall x, y < m$:

$$f(x) = g(y) \text{ iff } (\forall t < L)(\, (B(f(x))_t = B(g(y))_t \,),$$

hence

$$f(x) = g(y) \text{ iff } (\forall t < L)(\, (B^f)_{xL+t} = (B^g)_{yL+t} \,). \quad (*)$$

Finally, we know that a word of length mL is naturally associated with the unary predicate $X \subseteq mL$ defined by:

$$\forall t < mL : X(t) \Leftrightarrow (w)_t = 1$$

If for $h : m \to m$, the corresponding capital H is the unary predicate of mL associated in this way with the word $B^h \in \{0,1\}^{mL}$, the equivalence $(*)$ becomes, for all $x, y \in m$:

$$f(x) = g(y) \;\text{ iff }\; (\forall t < L)(F(xL + t) \leftrightarrow G(yL + t)).$$

Thus, we convert a unary functional formula into a monadic relational formula. This result may be extended to a large family of formulas on a signature made up, in addition to unary function symbols, of the predefined symbols $<$ and 0 (natural linear order on integers and zero). This "translation", from unary functional language to the less expressive monadic relational language, has a cost: the domain of finite structures considered must be expanded (we pass from unary functions on domain $m = \Theta(n/\log n)$ to unary relations on domain mL, which is $\Theta(m \log m) = \Theta(n)$), and the predefined symbols $<$, 0 must be replaced by the much more expressive predicate $+$.

The study of this flattening of functions in unary predicates constitutes the core of this paper. Combined with the above-mentioned characterization of a language $\mathcal{L} \in$ NLIN (see (1.2) p. 8), it allows to prove that for every $\mathcal{L} \in$ NLIN, there exists $\Theta \in SO(\exists, mon, +)$ (*i.e.* Θ is an existential monadic second-order formula with addition), such that:

$$w \in \mathcal{L} \ \text{ iff } \ \langle n, X, + \rangle \models \Theta,$$

where $n = length(w)$ and X is the unary predicate on n encoding w.

More precisely, the paper is organized as follows:

In **section 2** we prove the logical characterization of NLIN (Corollary 1.9). We give this proof although it is essentially presented in Gra88, Gra90, Gra94 because we want to have a self-contained paper, and because the statements and the proofs are scattered in these papers and *are not exactly* those of Theorem 1.9: the NLIN definition was introduced only recently in references Gra94, Gra94b. Moreover, we give in this section three very natural versions of the same result concerning functions (Theorem 1.8) and unary structures (Theorems 2.7 and 2.9) as inputs instead of words. This allows us to express the natural NP-complete graph problems - e.g. Hamilton circuit - in the same simple logic.

In **section 3** we first specify the above-mentioned identification between a unary function $f : m \to m$ and a unary predicate $F \subseteq mL$ ($L = \lfloor \log(m) + 1 \rfloor$). Actually, for technical reasons, the predicate F is built on a domain cn (for a fixed constant c) slightly bigger than mL. We will call F "flattening of f on cn". Then, we state precisely the passage from functional language to relational language: For each formula $\Psi \in FO(\forall)$ on signature $\{f, g_1, \ldots, g_p, <, 0\}$, where the g_i's are unary function symbols, there exists $\tilde{\Psi} \in FO(\forall^*\exists^*)$, on signature $\{F, G_1, \ldots, G_q, +\}$, where F and the G_i's are monadic predicate symbols, such that: if $f : m \to m$ and if F is the flattening of f on cn:

$$\langle m, f, <, 0 \rangle \models \exists \overline{g}\, \Psi \ \text{ iff } \ \langle cn, F, + \rangle \models \exists \overline{G}\, \tilde{\Psi}.$$

Then we combine this result with the NLIN characterization of Corollary 1.9 to prove: for every $\mathcal{L} \subseteq \{1,2\}^*$ in NLIN, there exists a formula $\hat{\Psi}$ in $FO(\forall^*\exists^*)$ such that for every $w \in \{1,2\}^n$:

$$w \in \mathcal{L} \ \text{ iff } \ \langle cn, F, + \rangle \models \exists \overline{G} \hat{\Psi}(F, \overline{G}, +) \tag{1.4}$$

where F is a unary predicate which encodes w on cn.

Actually, the formula $\hat{\Psi}$ above-mentioned is written with a set of predefined arithmetical predicates including $\{+\}$. To obtain the equivalence 1.4, we have to prove that all these predicates are definable from $+$. This is done in **4**, taking specific care with the first-order part of the involved formulas, to guarantee that $\hat{\Psi}$ does belong to Lynch's logic (*i.e.* has a $\forall^*\exists^*$ first-order prefix).

In **section 5**, we prove that the equivalence (1.4) still holds when the predicate F is replaced by a predicate $\mathcal{X}$ encoding the word w in a more natural way.

Section 6 focuses on a technical result which makes it possible to provide the preceding result with a more standard form. We first prove that if $R_1, \ldots, R_p$ are predicates with respective arities $r_1, \ldots, r_p$, on a domain of the form cn, and if Φ is a first-order formula on signature $\{R_1, \ldots, R_p\}$, then there exists a division of each R_i into a series R_i^* of $r_i - ary$ predicates on n, and a first-order formula Φ^* on signature $\{R_1^*, \ldots, R_p^*\}$, such that:

$$\langle cn, R_1, \ldots, R_p \rangle \models \Phi \ \text{ iff } \ \langle n, R_1^*, \ldots, R_p^* \rangle \models \Phi^*.$$

Moreover, the first-order prefix of Φ^* has the form $\forall^*\exists^*$, like Φ.

Associated to the result of section 5, this equivalence allows us to conclude: if $\mathcal{L} \subseteq \{1,2\}^*$ belongs to NLIN, then there exists $\Theta \in SO(\exists, mon, +)$ with first-order prefix $\forall^*\exists^*$ such that for every $w \in \{1,2\}^*$:

$$w \in \mathcal{L} \ \text{ iff } \ \langle n, X, + \rangle \models \Theta,$$

where X is the unary predicate naturally encoding w.

2. NLIN characterization with unary functions and only one variable

2.1. Proof of Theorem 1.8.

• We leave it as an exercice to the reader to prove that if there exists a formula $\Psi \in FO(qf)$ such that equivalence (1.2) holds for all $f : m \to m - \{0\}$ then

$\mathcal{F} \in$ NLIN. (Hint: an NRAM that recognizes $\mathcal{F}$ in NLIN has the following algorithm for an input $f : m \to m - \{0\}$: *guess* the p functions $g_1, \ldots, g_p : m \to m$ and *check* that the structure so expanded satisfies the one-variable formula $\forall x \Psi(x)$.)

• Let us prove the converse implication. Let $\mathcal{F}$ belong to NLIN. That means that $\mathcal{F}$ is recognized by an NRAM $\mathcal{R}$ in time cm, where c is a constant integer. Let us assume, without loss of generality, that $\mathcal{R}$ has two extra registers a and b, called "accumulators" (whose initial content is 0) and that its program is a sequence of instructions $\mathcal{I}(1), \mathcal{I}(2), \ldots, \mathcal{I}(r)$ of the following types:

(1)	$a := 0$	(7)	$R_a := b$
(2)	$a := a + 1$	(8)	*goto* $\mathcal{I}(i_0)$ *or* $\mathcal{I}(i_1)$
(3)	$a := a \dot{-} 1$	(9)	*If* $a = b$ *then goto* $\mathcal{I}(i_0)$ *else goto* $\mathcal{I}(i_1)$
(4)	*guess* (a)	(10)	*accept*
(5)	$a := R_a$	(11)	*reject*
(6)	$b := a$		

Comments.

- R_a denotes the register pointed to by accumulator a (i.e. the address of R_a is the content of a).
- In fact, only a is used for accessing the sequence of registers $R_0, R_1, \ldots$ (like the control head of a Turing machine).
- (4) and (8) are nondeterministic instructions: *guess* (a) consists in storing in a any integer less than or equal to cm.
- We can assume, without loss of generality, that any integer manipulated during the computation is less than or equal to cm.

Let $F : cm \to cm$ be the extension of f defined by $F(i) = f(i)$ if $i < m$ and $F(i) = 0$ otherwise. Let M denote the integer cm. Notice that the following formula of $FO(\forall\exists)$:

$$\phi_0 \equiv \forall x \exists y\, (succ(y) = x) \wedge (\forall x \neq\ last)(x < succ(x)) \wedge (succ(last) = 0)$$

defines the successor function $succ : M \to M$ such that $succ(i) = i + 1$ modulo M and the constant $last = M - 1$ on structure $\langle M, <, 0 \rangle$. That means that the structure $\langle M, <, 0, succ, last \rangle$ satisfies ϕ_0 iff the function $succ$ and the constant $last$ have their right values over M.

It is convenient to describe a computation of the NRAM $\mathcal{R}$ (of time cm on input f) in a first-order structure of domain $M = cm$ (notice: M is the time of the computation and is greater than any integer it involves) as we do in the following proposition similar to Theorem 1.8.

PROPOSITION 2.1. *Let $\mathcal{F}$ be a set of unary functions recognized by an NRAM $\mathcal{R}$ in time $M = cm$, where m is defined as above. Then there exists a formula $\Psi \in FO(\forall\exists^*)$ with only one first-order variable universally quantified on signature $\{F, G_1, \dots, G_p, <, 0, succ, last\}$, where F, the G_i's and $succ$ denote unary function symbols, $<$ is a binary predicate symbol and 0, $last$ are constant symbols, so that for each $f : m \to m - \{0\}$:*

$$f \in \mathcal{F} \text{ iff } \langle M, F, <, 0, succ, last\rangle \models \exists G_1 \dots \exists G_p \forall x \Psi(x, F, \overline{G}, <, 0, succ, last)$$

where $\langle M, F, <, 0, succ, last\rangle$ is the finite structure associated to f as above.

PROOF. Let us prove the proposition. Let $\mathcal{I}(1)$, $\mathcal{I}(2)$, ..., $\mathcal{I}(r)$ be the program of $\mathcal{R}$ (*i.e.* a list of instructions of type $(1) - (11)$). We assume that the only instruction *accept* of $\mathcal{R}$ is $\mathcal{I}(r)$. Let I, A, B, R_A, R'_A be unary function symbols whose intended meaning is the following:

- $I(t)$ is the index of the instruction performed at the instant t ;
- $A(t)$ is the content of accumulator a at the instant t ;
- $B(t)$ is the content of accumulator b at the instant t ;
- $R_A(t)$ is the content of R_a at the instant t (i.e. the content of $R_{A(t)}$) ;
- $R'_A(t)$ is the integer stored in R_a between instants t and $t+1$ according to instruction $\mathcal{I}(I(t))$, i.e. the content of register $R_{A(t)}$ after the execution of $\mathcal{I}(I(t))$.
- Moreover, we easily see that $F(i)$ is the content of register R_i at instant 0 for each $i < M$.

Let $\mathcal{S}_c$ be the signature $\{F, I, A, B, R_A, R'_A, <, 0, succ, last\}$. We now give some first-order $\mathcal{S}_c$-formulas in order to describe:
- the initial configuration of $\mathcal{R}$ on input w ;
- a computation step from instant t to instant $t + 1$ for each $t < M - 1$;
- the fact that $\mathcal{R}$ reaches the instruction *accept*.
We also use constants $1, 2, 3, \dots$ which abbreviate $succ(0)$, $succ(succ(0))$, $\dots$

• ϕ_1 describes the initial conditions:

$$\phi_1 \equiv I(0) = 1 \wedge A(0) = 0 \wedge B(0) = 0 \wedge R_A(0) = F(0)$$

• ϕ_2 describes a computation between any two successive instants, using for each instruction $\mathcal{I}(i)$ a formula $\Psi_i(t)$ corresponding to the type of $\mathcal{I}(i)$.

$$\phi_2 \equiv (\forall t \neq last) \bigwedge_{i=1}^{r} [I(t) = i \rightarrow \Psi_i(t)]$$

For example, if $\mathcal{I}(i)$ is the instruction "$a := a + 1$", then $\Psi_i(t)$ is the formula:

$$A(t+1) = A(t) + 1 \wedge I(t+1) = I(t) + 1 \wedge B(t+1) = B(t) \wedge R'_A(t) = R_A(t)$$

Figure 1 gives for each of the 11 types of instructions $\mathcal{I}(i)$ the corresponding formula $\Psi_i(t)$. For purpose of readability, we use some abbreviated formulas called $(same\ \ldots)(t)$, which indicate the registers among a, b, R_a that do not change their content. For instance, $(same\ b, R_a)(t)$ abbreviates the conjunction: $B(t+1) = B(t) \wedge R'_A(t) = R_A(t)$.

$\mathcal{I}(i)$	$\Psi_i(t)$
$a := a+1$ (similarly for $a := a \dot{-} 1$ and $a := 0$)	$A(t+1) = A(t) + 1$ $\wedge \ (\mathit{same}\ b, R_a)(t)$ $\wedge \ I(t+1) = I(t) + 1$
guess(a)	$(\mathit{same}\ b, R_a)(t)$ $\wedge$ $I(t+1) = I(t) + 1$
$a := R_a$	$A(t+1) = R_A(t)$ $\wedge \ (\mathit{same}\ b, R_a)(t)$ $\wedge \ I(t+1) = I(t) + 1$
$b := a$	$B(t+1) = A(t)$ $\wedge \ (\mathit{same}\ a, R_a)(t)$ $\wedge \ I(t+1) = I(t) + 1$
$R_a := b$	$R'_A(t) = B(t)$ $\wedge \ (\mathit{same}\ a, b)(t)$ $\wedge \ I(t+1) = I(t) + 1$
goto $\mathcal{I}(i_0)$ or $\mathcal{I}(i_1)$	$[I(t+1) = i_0 \vee I(t+1) = i_1]$ $\wedge$ $(\mathit{same}\ a, b, R_a)(t)$
If $a = b$ then goto $\mathcal{I}(i_0)$ else goto $\mathcal{I}(i_1)$	$(\mathit{same}\ a, b, R_a)(t)$ $\wedge \ (A(t) = B(t) \rightarrow I(t+1) = i_0)$ $\wedge \ (A(t) \neq B(t) \rightarrow I(t+1) = i_1)$
accept (or reject)	$I(t+1) = I(t)$

figure 1

The formula ϕ_2 above is not sufficient to compell the function $R_A(t)$ to be equal to $R_{A(t)}$ for each $t < M$. We have two cases:

First case: let t' be the last instant before t when $A(t') = A(t)$. Then the content of register $R_{A(t)}$ at instant t is the integer stored in R_a between instants t' and $t'+1$, that is $R'_A(t')$.

Second case: otherwise, if the register pointed by a at instant t, *i.e.* register $R_{A(t)}$, has never been examined before instant t, then its content is the initial content. That is: $R_{A(t)} = F_w(A(t))$.

In order to express the two above conditions with a formula in $FO(\forall\exists^*)$ (which can obviously be transformed into a $FO(\forall)$-formula by Skolemization), we now introduce new unary function symbols.

Let $\mathcal{R}_A = \{(A(t), t) \mid t < M\}$, and let $\prec$ denote the lexicographical order of $\mathcal{R}_A$. *Figure 2* shows how pairs $(A(t), t)$ represented by cells are lexicographically numbered.

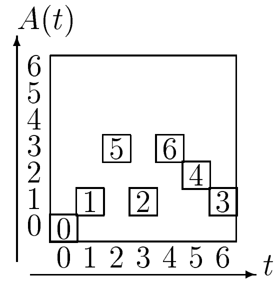

figure 2

We easily see how to rephrase the two above conditions: Let $t < M$ and $t' < M$ be such that $(A(t'), t')$ is the predecessor of $(A(t), t)$ for $\prec$. Then:

First case: if $A(t') = A(t)$ then $R_A(t) = R'_A(t')$.

Second case: if $A(t') \neq A(t)$ then $R_A(t) = F(A(t))$.

(Remark that the particular case $t = 0$ was already considered in formula ϕ_1.)

Let $N : M \to M \times M$ denote the lexicographical numbering of pairs $(A(t), t)$. In other words, N is the only strictly increasing function from $(M, <)$ to $(\mathcal{R}_A, \prec)$.

- N is defined by the following conjunction ϕ_3 (to be interpreted on domain M):

$$\phi_3 \equiv \forall x \exists t[N(x) = (A(t), t)] \wedge (\forall x \neq last)[N(x) \prec N(x+1)]$$

  More formally, function N should be represented by its two projections $N_1, N_2 : M \to M$, and ϕ_3 should be consequently rephrased.

- Using N (i.e. N_1, N_2), the two above conditions are rewritten as follows:

$$\begin{aligned}\phi_4 \equiv (\forall t \neq 0)\, \exists t' \, \exists x \ : \ & \{N(x) = (A(t'), t') \wedge N(x+1) = (A(t), t)\} \wedge \\ & \{A(t') = A(t) \to R_A(t) = R'_A(t')\} \wedge \\ & \{A(t') \neq A(t) \to R_A(t) = F(A(t))\}\end{aligned}$$

- The fact that $\mathcal{R}$ reaches the instruction *accept* is expressed by:

$$\phi_5 \equiv \exists t\,(I(t) = r)$$

Clearly, $\mathcal{R}$ accepts f (in time $M = cm$) if and only if $\langle M, F, <, 0, succ, last\rangle \models \phi_1 \wedge \phi_2 \wedge \phi_3 \wedge \phi_4 \wedge \phi_5$. Moreover, $\bigwedge_1^5 \phi_i$ can be put into the form $FO(\forall\exists^*)$, and hence into the form $FO(\forall)$, by Skolemization. This concludes the proof of Proposition 2.1. □

2.2. End of the proof of Theorem 1.8. Using the result of Proposition 2.1, we have to encode a first-order structure $\langle cm, F, <, 0, G_1, \ldots, G_p\rangle$ into a structure $\langle m, f, <, 0, \ldots\rangle$ on domain m.

The idea is to replace each function $G : cm \to cm$ by $2c$ functions:

- ◦ c functions $G_i^0 : m \to m$ $(i < c)$
- ◦ c functions $G_i^1 : m \to c$ $(i < c)$

with $G_i^0(y) = G(im + y)$ *modulo* m and $G_i^1(y) = \lfloor G(im + y)/m \rfloor$ (integer quotient) for every $i < c$, $y < m$.

In particular, for $G = F$ we have $G_0^0 = f$ and G_i^j is zero for each $(i, j) \neq (0, 0)$.

The transformation of the formulas is similar to the one we will describe in Section 6 for predicate symbols instead of function symbols, so we leave the details to the reader.

This concludes the proof of Theorem 1.8. □

2.3. Representation of relational structures over unary signatures: a generalization of Theorem 1.8. In this section, Theorem 1.8 is generalized to unary structures and other relational structures (e.g. graphs).

Definition 2.2. *A signature of the form $S = \{f_1, \ldots, f_p, c_1, \ldots, c_r\}$ where the f_i's are unary functions and the c_j's are constant symbols, is called a* unary signature.

A unary structure *is a first-order structure of domain $m > 0$ and unary signature.*

Remark 2.3. *When studying the precise (computational or logical) complexity of problems, it is often natural to encode the input as a unary structure in a sparse way. For example, a directed graph $G = (V, E)$ where $V = n = \{0, \ldots, n-1\}$ and $|E| = e$ is naturally encoded by the unary structure $\langle n+e, f_1, f_2\rangle$ where $E = \{n, \ldots, n+e-1\}$ (i.e. $V \cup E = n+e$) and functions $f_1, f_2 : V \cup E \to V \cup E$ are defined by:*

> *$f_1(x) = f_2(x) = x$ for each $x \in V$;*
>
> *for each edge α, $f_1(\alpha) = x$ (resp. $f_2(\alpha) = y$) if the first endpoint (resp. the second endpoint) of edge α is x (resp. y).*

More generally, any first-order structure can be similarly encoded as a unary structure.

DEFINITION 2.4. *Let $\mathcal{F}$ be a set of unary S-structures as above. The set $\mathcal{F}$ is in NLIN if it is recognized by an NRAM $\mathcal{R}$ in time $O(m)$ with the following initial conditions: at the beginning of the computation, registers $R_0, R_1, \ldots, R_r$ respectively contain $m-1$ and the values $c_1, \ldots, c_r$. Similarly, the initial content of R_{im+x} is $f_i(x)$ for each $i \leq p$ and $x < m$.*

REMARK 2.5. *The above initial conditions (i.e. the way the input structure is stored) make sense only if $m > r$; we chose them to obtain an easy encoding in our proof of Theorem 2.7 that follows. In fact, our NLIN class (of sets of unary structures) is* rather insensitive to the presentation of the input, *as the similar NLIN class of languages.*

REMARK 2.6. *Natural NP graph problems, e.g.* `Hamilton circuit`, *are in NLIN when they are encoded with unary structures (see Remark 2.3), i.e. are recognized on NRAMs within time $O(n + e)$ where n is the number of vertices and e is the number of edges.*

THEOREM 2.7. *Let $S = \{f_1, \ldots, f_p, c_1, \ldots, c_r\}$ ($p \geq 1$) be a unary signature. Let $\mathcal{F}$ be a set of S-structures.*

$\mathcal{F}$ is in NLIN iff there are a unary signature $S' = \{g_1, \ldots, g_q, c'_1, \ldots, c'_s\}$ and a quantifier-free first-order formula Ψ with only one variable, on signature $S \cup S' \cup \{<\}$, such that for each S-structure $\mathcal{A} = \langle m, f_1, \ldots, f_p, c_1, \ldots, c_r \rangle$:

$$\mathcal{A} \in \mathcal{F} \text{ iff } \langle \mathcal{A}, < \rangle \models \exists \overline{g}\, \exists \overline{c'}\, \forall x\, \Psi(x, \overline{f}, \overline{g}, \overline{c}, \overline{c'}, <).$$

PROOF. The proof is similar to the proof of Theorem 1.8. In place of function $F : cm \to cm$, we use a function $F_{\mathcal{A}} : cm \to cm$ that still represents precisely the way the input $\mathcal{A}$ is initially stored by the NRAM (cf. Definition 2.4):

> $F_{\mathcal{A}}(0) = m - 1$; $F_{\mathcal{A}}(i) = c_i$ for $i = 1, \ldots, r$;
>
> $F_{\mathcal{A}}(im + x) = f_i(x)$ for each $i = 1, \ldots, p$ and $x < m$.

The encoding of the first-order structure $\langle cm, F_{\mathcal{A}}, <, 0, \ldots\rangle$ into a structure $\langle \mathcal{A}, <, \ldots\rangle$ is very similar to the one described in subsection 2.2. So we leave the details to the reader. $\square$

The natural graph problems (and, more generally, relational structure problems) are closed under isomorphism.

DEFINITION 2.8. *A set $\mathcal{F}$ of (unary) S-structures is* closed under isomorphism *if for any S-structure $\mathcal{A} = \langle m, (f_i)_{i\leq p}, (c_j)_{j\leq r}\rangle$ and any permutation $\sigma : m \to m$,*

$$\mathcal{A} \in \mathcal{F} \text{ iff } \mathcal{A}^\sigma \in \mathcal{F},$$

where $\mathcal{A}^\sigma = \langle m, (f_i^\sigma)_{i\leq p}, (c_j^\sigma)_{j\leq r}\rangle$ and $f_i^\sigma(\sigma(x)) = \sigma(f_i(x))$ for all $i \leq p$ and $x < m$, and $c_j^\sigma = \sigma(c_j)$ for each $j \leq r$.

If $\mathcal{F}$ is a set of unary S-structures closed under isomorhism, the theorem 2.7 still holds for $\mathcal{F}$ even if we forbid the use of the linear order. More precisely:

THEOREM 2.9. *Let $S = \{f_1, \ldots, f_p, c_1, \ldots, c_r\}$ $(p \geq 1)$ be a unary signature. Let $\mathcal{F}$ be a set of S-structures* closed under isomorphism. *Then $\mathcal{F}$ is in NLIN iff there are a unary signature $S' = \{g_1, \ldots, g_p, c'_1, \ldots, c'_s\}$ and a quantifier-free first-order formula Ψ with only one variable, on signature $S \cup S'$, such that for each S-structure $\mathcal{A} = \langle m, f_1, \ldots, f_p, c_1, \ldots, c_r\rangle$:*

$$\mathcal{A} \in \mathcal{F} \text{ iff } \mathcal{A} \models \exists \overline{g}\, \exists \overline{c'}\, \forall x\, \Psi(x, \overline{f}, \overline{g}, \overline{c}, \overline{c'}).$$

PROOF. This follows immediately from Theorem 2.7 and from the fact, proved in Gra90b, that we can get rid of linear order by using new unary functions and formulas with only one first-order variable. $\square$

REMARK 2.10. *We easily see that we need only one constant symbol in $S \cup S'$, that is $r + s \geq 1$. More precisely, if $r \geq 1$ (i.e. the input structure contains some constants), then we can choose $s = 0$ (no quantification over constant symbols) and we can take $s = 1$ in case $r = 0$.*

Theorem 2.9 can be applied to the natural encoding (see Remark 2.3) of concrete NP-complete problems, since they belong to NLIN (see **?**).

EXAMPLE 2.11. *Let* `Hamilton Path` *(*`HP`*) denote the problem of the existence of a Hamilton path between two fixed vertices c_1, c_2 in a (directed) graph $G = (V, E)$. Let $\mathcal{A}(G, c_1, c_2)$ denote the unary structure $\langle V \cup E, f_1, f_2, c_1, c_2\rangle$ defined as in Remark 2.3. There is a formula $\Psi \in FO(qf)$ with one variable, on unary signature $\{f_1, f_2, \overline{g}, c_1, c_2\}$, such that:*

$$(G, c_1, c_2) \in \texttt{HP} \text{ iff } \mathcal{A}(G, c_1, c_2) \models \exists \overline{g} \forall x \Psi(x, f_1, f_2, \overline{g}, c_1, c_2).$$

We let the reader construct explicitly such a formula.

3. From unary functional language to monadic relational language

3.1. Notations.

- $\mathbb{N}$ is the set of positive integers.
- For $q \in \mathbb{N}$, $B(q)$ denotes the reverse binary representation of q (e.g. $B(6) = 011$).
- For $i \in \mathbb{N}$, 0^i is the concatenation of i 0-symbols.
- Let q, L be positive integers such that $L \geq \lfloor \log(q) + 1 \rfloor = length(B(q))$. We note $B_L(q) = B(q)^\frown 0^{L-length(B(q))}$ (e.g. $B_5(6) = 01100$). $B_L(q)$ is in fact the word encoding q in reverse binary which is forced to reach the length of L, possibly padding it with 0's on its right.
- For $m \in \mathbb{N}$, we also denote m the domain $\{0, 1, \ldots, m-1\}$. Note that for m and $L \geq \lfloor \log(m) + 1 \rfloor$ fixed in $\mathbb{N}$, the map $B_L : m \to \{0,1\}^L$, which maps q to $B_L(q)$, is injective.
- For $m \in \mathbb{N}$, Id_m denotes the identity function on m.
- For $w = a_0 a_1 \ldots a_{n-1} \in \{0,1\}^*$ and for $t < n$, we will call "t^{th} bit of w" the symbol a_t, *i.e.* the $(t+1)^{th}$ letter of the word w.
- Let $w = b_0 b_1 \ldots b_{n-1} \in \{0,1\}^n$. We will call *predicate naturally encoding* w the unary predicate $X_w \subseteq n$ defined by $\forall t < n : X_w(t) \Leftrightarrow b_t = 1$.

  In the same way, if $w = d_0 d_1 \ldots d_{n-1} \in \{1,2\}^n$ is a word on the dyadic alphabet, the "predicate naturally coding w" will be the unary predicate X_w on n defined by $\forall t < n : X_w(t) \Leftrightarrow d_t = 2$.

REMARK 3.1. *In the present paper, both the binary and the dyadic representation of positive integers are used. (Recall that dyadic representation consists in encoding each positive integer x by the word $D(x) = x_p \ldots x_1 x_0 \in \{1,2\}^*$, where $(x_0, x_1, \ldots, x_p)$ is the only tuple on $\{1,2\}$ such that: $x = x_0 + x_1 2 + \ldots + x_p 2^p$.) Binary representation is used when padded and uniform length notations are helpful. Dyadic one allows to identify any nonempty word of $\{1,2\}^*$ with a positive integer, in a one-one way.*

3.2. Flattening of a function. For this whole paragraph, m, L, and m' denote three integers such that: $L = \lfloor \log(m) + 1 \rfloor$ and $m' \geq mL$. We show that, using a simple encoding, each function $h : m \to m$ can be identified with a unary relation H on m'.

Let $h : m \to m$. For every $i < m$, let us denote $b_i = B_L(h(i))$. The image of each $i < m$ being thus encoded by a word of $\{0,1\}^L$, h can be encoded by a word w_h, the length of which is exactly m':

$$w_h = b_0^\frown b_1^\frown \ldots ^\frown b_{m-1}^\frown 0^{m'-mL}$$

Remark 3.2. *If for $x < m$ and $t < L$, b_x^t denotes the t^{th} bit of b_x, then:*

$$h(x) = \sum_{t=0}^{L-1} b_x^t 2^t = \sum_{\substack{t<L\, st \\ b_x^t=1}} 2^t$$

Every function from m to m is thus represented in a unique way by a word of $\{0,1\}^{mL}.0^{m'-mL}$. Besides, one usually represents a word $v \in \{0,1\}^*$ of length q by the unary predicate $V \subseteq q$ defined as follows:

$$\forall i < q : \ V(i) \ \Leftrightarrow \ the\ t^{th}\ bit\ of\ v\ is\ 1.$$

Composition of these two encodings makes it possible to associate to each unary function $h : m \to m$ a unary predicate $H \subseteq m'$ defined as follows:

$$\forall \alpha < m' : \ H(\alpha) \Leftrightarrow the\ \alpha^{th}\ bit\ of\ w_h\ is\ 1$$

$$i.e. \qquad H(\alpha) \Leftrightarrow \alpha = x_\alpha L + t_\alpha,\ with\ x_\alpha < m\,,\ t_\alpha < L\,,\ and\ b_{x_\alpha}^{t_\alpha} = 1,$$

which implies the following:

$$\forall x < m\,, \forall t < L \ : \ H(xL + t) \Leftrightarrow b_x^t = 1.$$

Definition 3.3. *From now on the predicate $H \subseteq m'$ thus constructed from the function $h : m \to m$ will be called the* flattening of h on m'.

Remark 3.4. *The previous remark immediately results in:*

$$\forall x < m \ : h(x) = \sum_{\substack{t<L\, st \\ H(xL+t)}} 2^t.$$

Definition 3.5. *We call $Bit_m^{m'}$ the flattening of the identity function Id_m : $m \to m$ on m'.*

In other words, $Bit_m^{m'}$ is the unary predicate on m' naturally associated to the word $B_L(0)^\frown B_L(1)^\frown \ldots {}^\frown B_L(m-1)^\frown 0^{m'-mL} \in \{0,1\}^{m'}$. Or:

$$\forall \alpha < m' : \ Bit_m^{m'}(\alpha) \Leftrightarrow \begin{cases} \alpha = x_\alpha L + t_\alpha, \ \textit{with } x_\alpha < m\,, \ t_\alpha < L\,, \\ \textit{and the } t_\alpha^{th} \textit{ bit of } B_L(x_\alpha) \textit{ is } 1 \end{cases}$$

In fact, to make the notations simpler, we will denote this predicate Bit, the (m, m') pair which it refers to being implicitly given by the context.

3.3. From functional formulas to relational formulas. Let $k > 1$ be a fixed integer. Recall that for every integer n, we denote: $l = \lceil \frac{\log(n+1)}{k} \rceil$, $m = \lceil \frac{n}{l} \rceil$ and $L = \lfloor \log(m) + 1 \rfloor$. Given these notations, we get: $m = \Theta(n/\log n)$ and therefore: $mL = \Theta(n)$. This guarantees the existence of an integer c such that:

$$\forall n \in \mathbb{N},\, mL < cn \quad (*)$$

Let c be a fixed integer meeting $(*)$. Therefore we have $mL < cn$ for all $n \in \mathbb{N}$, and it is possible to refer to the flattening of a function $h : m \to m$ on cn.

In this paragraph, we associate to each language $\mathcal{L} \in$ NLIN, an existential monadic second-order formula $\Phi_{\mathcal{L}}$, in such a way that a word $w \in \{1,2\}^*$ is in $\mathcal{L}$ iff a certain finite structure, in which w is encoded by the flattening of f_w, is a model of $\Phi_{\mathcal{L}}$. This will be done along Lemmas 3.6, 3.7, 3.8, 3.9 in using Corollary 1.9 and stating successively that:

• Flattenings are quite adapted to translate certain functional formulas, which we could call "simple (atomic) formulas", into relational formulas, provided we add to the language a series of predefined arithmetical symbols including $+$.

• Does mean changing a little its first-order quantifier prefix, the formula which characterizes a language $\mathcal{L} \in$ NLIN (by Corollary 1.9) can be required to be a boolean combination of these "simple formulas".

• The conjunction of the two preceding facts allows to associate to each $\mathcal{L} \in$ NLIN an existential monadic second-order formula whose signature contains the above-mentioned arithmetical objects. Moreover, if the transformation is made carefully, the first-order part of this formula can be required to belong to the class $FO(\forall^*\exists^*)$.

• At last, we state that, excepted for $+$, all these arithmetical symbols can be removed, without changing the structure of the formula.

All this will lead to Corollary 3.10, which establishes the existence of the formula $\Phi_{\mathcal{L}}$.

Lemma 3.6. *Let f, g be two unary functions on m, F, G their flattenings on cn. Then, for all $x, y < m$, the assertions $m \models f(x) = g(y)$ and $m \models f(x) < g(y)$*

are respectively equivalent to :

$$cn \models (\forall t < L)(F(xL+t) \leftrightarrow G(yL+t))$$

and

$$cn \models (\exists t < L)(\forall t' < L)\left[\begin{array}{l} t' > t \to (F(xL+t') \leftrightarrow G(yL+t')\wedge \\ G(yL+t) \wedge \neg F(xL+t)) \end{array}\right]$$

PROOF. We just have to recall the connection between a function $h : m \to m$, its associated word $B_L(h(0))^\frown \ldots ^\frown B_L(h(m-1)) \in \{0,1\}^*$ and its flattening on cn to see that these two equivalences respectively express the following facts:

1. $f(x) = g(y)$ iff the words $B_L(f(x))$ and $B_L(g(y))$ are equal, *i.e.* coincide on their L successive bits.
2. $f(x) < g(y)$ iff $B_L(f(x)) < B_L(g(y))$ for the reverse lexicographic order on $\{0,1\}^L$.

LEMMA 3.7. *Let $\mathcal{L} \subseteq \Sigma^*$. If $\mathcal{L} \in$ NLIN, then there exists $\Psi^o \in FO(\forall^*)$ on a signature made up, beside the symbols $<, 0$, of unary function symbols $f, g_1 \ldots g_r$, such that:*

(i) $\forall w \in \Sigma^*,\ w \in \mathcal{L} \ \Leftrightarrow\ \langle m, f_w, <, 0\rangle \models \exists g_1 \ldots \exists g_r \Psi^o(f, \overline{g}, <, 0)$.

(ii) Ψ^o *has the following form:* $\Psi^o \equiv \forall \overline{x} \bigwedge_i \bigvee_j A_{ij}$, *where each* A_{ij} *can be written:* $\alpha(u) = \beta(v)$ *or* $\neg[\alpha(u) = \beta(v)]$ *or* $\neg[\alpha(u) < \beta(v)]$, *for some* $\alpha, \beta \in \{f, \overline{g}, Id\}$ *and some* $u, v \in \{0, \overline{x}\}$.

PROOF. Let $\Psi \in FO(qf)$ be the formula associated to $\mathcal{L}$ by Corollary 1.9. So we have, for all $w \in \Sigma^*$:

$$w \in \mathcal{L} \ \Leftrightarrow\ \langle m, f_w, <, 0\rangle \models \exists g_1 \ldots \exists g_p \forall x \Psi(x, f, \overline{g}, <, 0).$$

We can assume, without loss of generality, that Ψ is in conjunctive normal form. Then an atomic subformula of Ψ has the form:

$$h_1 \circ \ldots \circ h_p(u) \propto k_1 \circ \ldots \circ k_s(v),$$

with $h_i, k_i \in \{f, \overline{g}, Id\}$ (where Id is the identity function), $u, v \in \{0, x\}$ and $\propto \in \{=, \neq, <, \not<\}$. But this formula is clearly equivalent to:

$$\begin{array}{ll}\forall x & [\alpha_{p-1}(x) = h_{p-1} \circ h_p(x) \wedge \bigwedge_{i=1}^{i=p-2} \alpha_i(x) = h_i \circ \alpha_{i+1}(x) \wedge \\ & \beta_{s-1}(x) = k_{s-1} \circ k_s(x) \wedge \bigwedge_{i=1}^{i=s-2} \beta_i(x) = k_i \circ \beta_{i+1}(x) \wedge \\ & \alpha_1(u) \propto \beta_1(v)] \ ,\end{array}$$

where the α_i's and β_i's are new function symbols. Thus, if we still denote $\overline{g}$ the tuple $g_1, \ldots, g_r$ made up of $g_1, \ldots, g_p$ in addition to as many new function symbols as necessary, this allows to write Ψ in the following equivalent form:

$$\Psi \equiv \bigwedge \alpha \circ \beta(x) = \gamma(x) \wedge \bigwedge \bigvee \varepsilon(u) \propto \varepsilon'(v),$$

with $\alpha, \beta, \gamma \in \{f, \overline{g}\}$, $\varepsilon, \varepsilon' \in \{f, \overline{g}, Id\}$, $\propto \in \{=, \neq, <, \not<\}$. Besides, replacing each subformula $\alpha \circ \beta(x) = \gamma(x)$ of Ψ by $\forall y[\beta(x) = y \to \alpha(y) = \gamma(x)]$, and writing the obtained formula in prenex conjunctive normal form, we get for $\forall x \Psi$ an equivalent formula of the form:

$$\forall \overline{x} \bigwedge \bigvee \alpha(u) \propto \beta(v)$$

with $\alpha, \beta \in \{f, \overline{g}, Id\}$, $u, v \in \{0, \overline{x}\}$, $\propto \in \{=, \neq, <, \not<\}$, which can be written more precisely: $\forall \overline{x} \bigwedge_i \bigvee_j B_{ij}$, where each B_{ij} is one of the formulas $\alpha(u) = \beta(v)$, $\alpha(u) \neq \beta(v)$, $\alpha(u) < \beta(v)$, $\alpha(u) \not< \beta(v)$ for some $\alpha, \beta \in \{f, \overline{g}, Id\}$ and some $u, v \in \{0, \overline{x}\}$.

At last, we observe that the formula $\alpha(u) < \beta(v)$ can be expressed with the others (namely, those of the form $\alpha(u) \neq \beta(v)$, $\alpha(u) \not< \beta(v)$) by a universal formula using only the connectives $\wedge$ and $\vee$. Indeed we have:

$\alpha(u) < \beta(v) \Leftrightarrow \neg[\beta(v) < \alpha(u)] \wedge \neg[\beta(v) = \alpha(u)]$

The prenex conjunctive normal form of the formula produced by this last rewriting is the sought formula Ψ^o. □

Lemma 3.8. *Let $\mathcal{L} \subseteq \Sigma^*$. If $\mathcal{L} \in$ NLIN, then there exists $\tilde{\Psi} \in FO(\forall^*\exists^*)$ on signature $\{F, G_1, \ldots, G_r, +, <, \times, Bit, m, L\}$, where $F, G_i, +, <, \times, Bit$ are predicate symbols of respective arities 1, 1 (for every $i \leq r$), 3, 2, 3, 1, and m, L are constants symbols, such that, for every $w \in \Sigma^*$:*

$$w \in \mathcal{L} \ \Leftrightarrow \ \langle cn, F \rangle \models \exists G_1 \ldots \exists G_r \tilde{\Psi}(F, \overline{G}, +, <, \times, Bit, m, L),$$

where F is the flattening of f_w on cn, $+, <, \times$ are respectively the predefined addition, linear order and product on cn, Bit is the flattening of Id_m on cn, and m, L are the two predefined constants defined at the beginning of this subsection.

PROOF. Let $\Psi^o \equiv \forall\overline{x} \bigwedge_i \bigvee_j A_{ij}$ be the formula associated to $\mathcal{L}$ by Lemma 3.7. We compute $\tilde{\Psi}$ from Ψ^o in translating each subformula A_{ij} according to the equivalences stated in Lemma 3.6. Namely, we have the following translations:

$$\begin{array}{rcl} \alpha(u)=\beta(v) & \text{becomes} & (\forall t<L)[A(uL+t)\leftrightarrow B(vL+t)]\ ; \\ \neg[\alpha(u)=\beta(v)] & \text{becomes} & (\exists t<L)\neg[A(uL+t)\leftrightarrow B(vL+t)]\ ; \\ \neg[\alpha(u)<\beta(v)] & \text{becomes} & (\forall t<L)(\exists t'<L) \\ & & \neg\left[\begin{array}{l} t'>t\rightarrow(A(uL+t')\leftrightarrow B(vL+t')\wedge \\ B(vL+t)\wedge\neg A(uL+t)) \end{array}\right] \end{array}$$

Notice that all these formulas are in $FO(\forall^*\exists^*)$. Thus, rewriting the obtained formula into prenex conjunctive normal form, we get a new formula with the following pattern:

$$(\forall\overline{x}<m)(\forall\overline{t}_1<L)(\exists\overline{t}_2<L)\bigwedge\bigvee\pm H(yL+t),$$

with $H\in\{F,\overline{G},Bit\}$, $y\in\overline{x}$, $t\in\{\overline{t}_1,\overline{t}_2\}$. This is the sought formula $\tilde{\Psi}$. $\square$

We can achieve a stronger result in showing that the previous lemma still holds when using $+$ as the only arithmetical predefined symbol.

Recall that for k, c two fixed integers and for $n\in\mathbb{N}$, we denote: $l=\lceil\frac{\log(n+1)}{k}\rceil$, $m=\lceil\frac{n}{l}\rceil$, $L=\lfloor\log(m)+1\rfloor$. The following result, essentially due to Lynch, states that the constants k, c, n, l, m, L, as well as the predicates $<$, $\times$, Bit, are definable in cn by existential monadic second-order formulas on a signature including the predefined predicate $+$. Moreover, this definability preserves the $\forall^*\exists^*$ quantifier pattern of the formulas. More precisely, we have the:

LEMMA 3.9. *Let σ_0 denote the set of the predefined symbols k, c, n, l, m, L, Bit, $\times$, $<$, $+$. Let $\overline{U}$ be a series of unary predicate symbols. Let $\Phi\in FO(\forall^*\exists^*)$ be a sentence on the signature $\sigma_0\cup\{\overline{U}\}$. Then there exists $\Phi'\in FO(\forall^*\exists^*)$, on the signature $\{+\}\cup\{\overline{U}\}\cup\{D_1,\ldots,D_j\}$, where D_i's are new unary predicate symbols, such that:*

$$cn\models\Phi(\sigma_0,\overline{U})\ \Leftrightarrow\ cn\models\exists D_1\ldots\exists D_j\Phi'(\overline{U},+,\overline{D}).$$

PROOF. Definability of the predicates $<$, $\times$, Bit, has been proved in Lyn82. We will recall the main steps of this proof and we will show the definability of the arithmetical constants k, c, n, l, m, L, taking a specific care to the quantifier prefixes of the used formulas. But since these proofs are long and rather technical, we put them together in a following section, especially devoted to definability of arithmetical predicates and constants. $\square$

Corollary 3.10. *Let $\mathcal{L}$ be a language on $\Sigma = \{1, 2\}$. If $\mathcal{L} \in NLIN$, then there exists a sentence $\tilde{\Psi} \in FO(\forall^*\exists^*)$ on signature $\{F, G_1, \ldots, G_q, +\}$, where F and the G_i's are unary predicate symbols, $+$ is a ternary predicate symbol, such that:*

$$[w \in \mathcal{L}] \Leftrightarrow [\langle cn, F, + \rangle \models \exists G_1 \ldots \exists G_q \tilde{\Psi}(F, G_1, \ldots, G_q, +)]$$

where F is the flattening of f_w on cn, and $+$ is the predefined addition on cn.

4. Definability of arithmetical objects

4.1. Square, exponential, product, arithmetical constants.

Notations. X^2, $X^2 - X$, 2^X, $2^X - X$ denote the following unary predicates on cn :
$X^2 = \{x^2,\ 4 \leq x^2 < cn\}$
$X^2 - X = \{x^2 - x,\ 4 \leq x^2 - x < cn\}$
$2^X = \{2^x,\ 4 \leq 2^x < cn\}$
$2^X - X = \{2^x - x,\ 4 \leq 2^x < cn\}$

Lemma 4.1. (Lyn82, p.134) *There exist two formulas $S(U_0, U_1)$ and $s(x, y, U_0, U_1)$, respectively belonging to $FO(\forall^3\exists)$ and $FO(\exists\forall)$, such that:*
(i) There exist $U_0, U_1 \subseteq cn$ s.t. $\langle cn, + \rangle \models S(U_0, U_1)$.
(ii) If $U_0, U_1 \subseteq cn$ and $\langle cn, + \rangle \models S(U_0, U_1)$, then for all $x, y < cn$:
$\langle cn, + \rangle \models s(x, y, U_0, U_1) \Leftrightarrow y = x^2$.

Lemma 4.2. (Lyn82, p.135) *There exist two formulas $E(V_0, V_1)$ and $e(x, y, V_0, V_1)$, respectively belonging to $FO(\forall^3\exists)$ and $FO(\exists\forall)$, such that :*
(i) There exist $V_0, V_1 \subseteq cn$ s.t. $\langle cn, + \rangle \models E(V_0, V_1)$.
(ii) If $V_0, V_1 \subseteq cn$ and $\langle cn, + \rangle \models E(V_0, V_1)$, then for all $x, y < cn$:
$\langle cn, + \rangle \models e(x, y, V_0, V_1) \Leftrightarrow y = 2^x$.

Remark 4.3. *$S(U_0, U_1)$ and $E(V_0, V_1)$ respectively express :*
"$U_0 = X^2,\ U_1 = X^2 - X$" and "$V_0 = 2^X,\ V_1 = 2^X - X$".

Lemma 4.4. *There exist two formulas $d_0(a_0)$ and $d_1(a_0, a_1, U_0, U_1)$, respectively belonging to $FO(\forall)$ and $FO(\forall^3\exists)$, such that for every $a_0, a_1 < cn$:*
(i) $cn \models d_0(a_0)$ iff $a_0 = cn - 1$.
(ii) $cn \models S(U_0, U_1) \wedge d_0(a_0) \wedge d_1(a_0, a_1, U_0, U_1)$ iff $a_0 = cn - 1$ and $a_1 = \lfloor \sqrt{cn - 1} \rfloor$.

PROOF.

$$\begin{aligned} d_0(a_0) &\equiv \ \forall x(x \leq a_0) \\ d_1(a_0,a_1) &\equiv \ \forall x(a_1^2 = x \to x = \lfloor \sqrt{cn-1} \rfloor^2) \\ &\equiv \ \forall x(\neg s(a_1,x,U_0,U_1) \vee [U_0(x) \wedge \forall y(U_0(y) \to y \leq x)] \end{aligned}$$

LEMMA 4.5. (LYN82, P.135) *There exists* $p(x,y,z,a_0,a_1,U_0,U_1) \in FO(\exists^*\forall)$ *s.t.:*
If U_0, U_1 *are unary relations on* cn *and* a_0, a_1 *two integers* $< cn$ *such that* $cn \models S(U_0,U_1) \wedge d_0(a_0) \wedge d_1(a_0,a_1)$, *then for all* $x,y,z < cn$: $cn \models p(x,y,z,a_0,a_1)$ *iff* $z = xy$.

LEMMA 4.6. *There exist* $d_2(x_0,x_1,x_2)$, $d_3(x_0,x_1,x_2,x_3)$, $d_4(x_0,x_1,x_2,x_3,x_4)$, $d_5(x_0,x_1,x_2,x_3,x_4,x_5)$, *formulas belonging to* $FO(\forall^*\exists)$, *such that for all* U_0, U_1, V_0, $V_1 \subseteq cn$ *and* $a_0, a_1, \ldots, a_5 < cn$:
If $\langle cn,+,\overline{U},\overline{V}\rangle \models S(U_0,U_1) \wedge E(V_0,V_1) \wedge d_0(a_0) \wedge d_1(a_0,a_1)$ *then* :
$\langle cn,+,\overline{U},\overline{V}\rangle \models d_2(a_0,a_1,a_2)$ *iff* $a_2 = n-1$;
$\langle cn,+,\overline{U},\overline{V}\rangle \models d_3(a_0,\ldots,a_3)$ *iff* $a_3 = l = \lceil \frac{\log(n+1)}{k} \rceil$;
$\langle cn,+,\overline{U},\overline{V}\rangle \models d_4(a_0,\ldots,a_4)$ *iff* $a_4 = m = \lceil \frac{n}{l} \rceil$;
$\langle cn,+,\overline{U},\overline{V}\rangle \models d_5(a_0,\ldots,a_5)$ *iff* $a_5 = L = \lfloor \log(m)+1 \rfloor$.

PROOF.

$$\begin{aligned} a_2 = n-1 \ &\Leftrightarrow ca_2 + (c-1) = a_0 \\ a_3 = l \ &\Leftrightarrow a_3 = \lceil \tfrac{\log(n+1)}{k} \rceil \\ &\Leftrightarrow 2^{k(a_3-1)} \leq n < 2^{ka_3} \\ &\Leftrightarrow a_3 \text{ is the greatest } x < cn \text{ s.t. } 2^{k(x-1)} \leq n \\ &\Leftrightarrow \forall x(2^{k(x-1)} \leq n \to x \leq a_3) \wedge 2^{k(a_3-1)} \leq n \\ &\Leftrightarrow \left[\begin{array}{l} \forall x \forall y \forall z[(y = k(x-1) \wedge z = 2^y \wedge z \leq n) \to x \leq a_3] \\ \wedge \forall y \forall z[(y = k(a_3-1) \wedge z = 2^y) \to z \leq n] \end{array} \right] \\ &\Leftrightarrow \forall x \forall y \forall z \left[\begin{array}{l} [(y = k(x-1) \wedge e(y,z) \wedge z \leq a_2+1) \to x \leq a_3] \\ \wedge[(y = k(a_3-1) \wedge e(y,z)) \to z \leq a_2+1] \end{array} \right] \end{aligned}$$

and $e(y,x) \in FO(\exists\forall)$ implies $\neg e(y,x) \in FO(\forall\exists)$, which gives the sought result.

$$\begin{aligned} a_4 = m \ &\Leftrightarrow a_4 = \lceil \tfrac{n}{l} \rceil \\ &\Leftrightarrow l(a_4-1) \leq n-1 < la_4 \\ &\Leftrightarrow a_4 \text{ is the greatest } x < cn \text{ s.t. } l(x-1) \leq n-1 \\ &\Leftrightarrow \forall x \forall y \ [((y = l(x-1) \wedge y \leq n-1) \to x \leq a_4) \wedge \\ &\qquad\qquad (y = l(a_4-1) \to y \leq n-1)] \\ &\Leftrightarrow \forall x \forall y \ [((p(a_3,x-1,y) \wedge y \leq a_2) \to x \leq a_4) \wedge \\ &\qquad\qquad (p(a_3,a_4-1,y) \to y \leq a_2) \end{aligned}$$

and the assertion $p(a_3, a_4, x) \in FO(\exists^*\forall)$ gives the sought result.

$$\begin{aligned}
a_5 = L &\Leftrightarrow a_5 = \lfloor \log(m) + 1 \rfloor \\
&\Leftrightarrow a_5 \leq \log(m) + 1 < a_5 + 1 \\
&\Leftrightarrow 2^{a_5 - 1} \leq m < 2^{a_5} \\
&\Leftrightarrow 2^{a_5} \leq 2a_4 < 2 \cdot 2^{a_5} \\
&\Leftrightarrow \forall x(x = 2^{a_5} \rightarrow x \leq 2a_4 < 2x) \\
&\Leftrightarrow \forall x(\neg e(a_5, x) \vee x \leq 2a_4 < 2x)
\end{aligned}$$

but $e(a_5, x) \in FO(\exists\forall)$, hence the result. □

4.2. BIT-definability .

4.2.1 Preliminaries.

We have :
$\forall \alpha < cn\ \exists! x, t < cn\ s.t.\ \alpha = xL + t\ and\ t < L$ (euclidean division of α by L).
$\forall x < cn\ \forall t < L\ \exists! q, r < cn\ s.t.\ x = q2^t + r\ and\ r < 2^t$ (euclidean division of x by 2^t). Hence:

$$\forall \alpha < cn\ \exists!(q, x, r, t) \in (cn)^4 \text{ s.t. } \begin{cases} x = q2^t + r \\ \alpha = xL + t \\ t < L \text{ and } r < 2^t \end{cases}$$

Which can be written, by further decomposing the used operations :

$$\forall \alpha < cn\ \exists!(q, x, r, t, y_1, y_2, y_3) \in (cn)^7 \text{ s.t. } \begin{cases} y_1 = 2^t \\ y_2 = qy_1 \\ x = y_2 + r \\ y_3 = xL \\ \alpha = y_3 + t \\ t < L \\ r < y_1 \end{cases} \quad (4.5)$$

Let us call *good decomposition of* α and denote $gd(\alpha)$, the only seven-tuple $(q, x, r, t, y_1, y_2, y_3)$ of elements of cn satisfying (4.5).
Let us denote $GD(\alpha, q, x, r, t, y_1, y_2, y_3, L)$ the conjunction:

$$\begin{aligned}
&(y_1 = 2^t) \wedge (y_2 = qy_1) \wedge (x = y_2 + r) \wedge (y_3 = xL) \wedge \\
&(\alpha = y_3 + t) \wedge (t < L) \wedge (r < y_1)
\end{aligned}$$

$$\begin{aligned}
i.e.\ GD(\alpha, q, x, r, t, y_1, y_2, y_3, L) \equiv{}& e(t, y_1) \wedge p(q, y_1, y_2) \wedge (x = y_2 + r) \wedge \\
& p(x, L, y_3) \wedge (\alpha = y_3 + t) \wedge (t < L) \wedge (r < y_1).
\end{aligned}$$

Then we have, for each $(\alpha, q, x, r, t, y_1, y_2, y_3) \in (cn)^8$:
$(q, x, r, t, y_1, y_2, y_3) = gd(\alpha) \Leftrightarrow \langle cn, + \rangle \models GD(\alpha, q, x, r, t, y_1, y_2, y_3, L)$.

REMARK 4.7. *$GD \in FO(\exists^*\forall)$.*

4.2.2 Definability of BIT with a formula of $FO(\forall^*\exists^*)$.

We have seen (definition 3.5) that BIT is defined on each $\alpha < cn$ by:
$BIT(\alpha) \Leftrightarrow \exists x, t < cn\ s.t.\ \alpha = xL + t,\ t < L,\ x < m$, and the t^{th} bit of $B_L(x)$ is 1, where $B_L(x)$ denotes the reverse binary encoding of lenght L of x.
But the t^{th} bit of $B_L(x)$ is 1 iff $x = q2^t + r$ with $r < 2^t$ and q odd. So, we have $BIT(\alpha)$ iff the first term of $gd(\alpha)$ is odd and the second is $< m$. That can be easily written as :

$$BIT(\alpha) \Leftrightarrow cn \models \exists q, x, r, t, y_1, y_2, y_3[GD(\alpha, q, \ldots, y_3, L) \wedge (x < m) \wedge \exists q'(q = 2q' + 1)]$$

We will denote : $BIT(\alpha) \Leftrightarrow cn \models B_1(\alpha, L, m)$
On the other hand, existence and uniqueness of $gd(\alpha)$ for each α allows to characterize BIT in a second way :

$$BIT(\alpha) \Leftrightarrow cn \models \forall q, x, r, t, y_1, y_2, y_3[GD(\alpha, q, \ldots, y_3, L) \rightarrow ((x < m) \wedge \exists q'(q = 2q' + 1))],$$

that we will denote: $BIT(\alpha) \Leftrightarrow cn \models B_2(\alpha, L, m)$.

REMARK 4.8. *From $GD \in FO(\exists^*\forall)$, one immediately deduces $B_1 \in FO(\exists^*\forall)$ and $B_2 \in FO(\forall^*\exists)$.*

From the two previous equivalences, we deduce a third characterization of BIT:
Let $B \subseteq cn$. Then:
$B = BIT \;\Leftrightarrow\; cn \models \mathrm{Def}_{bit}(B, m, L)$, where Def_{bit} is the following formula:
$\mathrm{Def}_{bit}(B, m, L) \equiv \forall\alpha[(B(\alpha) \rightarrow B_2(\alpha, L, m)) \wedge (B_1(\alpha, L, m) \rightarrow B(\alpha))]$

REMARK 4.9. *The previous remark leads to: $\mathrm{Def}_{bit}(B, m, L) \in FO(\forall^*\exists^*)$.*

4.3. Elimination of arithmetical functions.

LEMMA 4.10. *Let σ_0 denote the set of the predefined symbols k, c, n, l, m, L, Bit, $\times$, $<$, $+$. Let $F, G_1, \ldots, G_p$ be a series of unary predicate symbols. Let Φ be a sentence on the signature $\sigma_0 \cup \{F, \overline{G}\}$ which has the form :*

$$(\forall\overline{x} < m)(\forall\overline{t}_1 < L)(\exists\overline{t}_2 < L) \bigwedge \bigvee \pm H(xL + t)$$

with $x \in \overline{x}$, $t \in \{\overline{t}_1, \overline{t}_2\}$, $H \in \{F, \overline{G}, BIT\}$.
Then there exists $\Phi' \in FO(\forall^*\exists^*)$, on the signature $\{+\} \cup \{F, \overline{G}\} \cup \{D_1, \ldots, D_j\}$, where the D_i's are new unary predicate symbols, such that:

$$\langle cn, F\rangle \models \exists\overline{G}\Phi(F, \overline{G}, \sigma_0,) \;\Leftrightarrow\; \langle cn, F\rangle \models \exists\overline{G}\exists\overline{D}\Phi'(F, \overline{G}, \overline{D}, +).$$

Proof.

Let $A_0, \ldots, A_5$ be unary predicate symbols. We denote $T(A_0, \ldots, A_5)$ the following formula :

$$\begin{array}{ll} \forall x_0 \ldots \forall x_5 & [(A_0(x_0) \to d_0(x_0)) \wedge ((A_0(x_0) \wedge A_1(x_1)) \to d_1(x_0, x_1)) \\ & \wedge \ldots \wedge ((A_0(x_0) \wedge \ldots \wedge A_5(x_5)) \to d_5(x_0, \ldots, x_5))] \wedge \\ \exists a_0 \ldots \exists a_5 & [\bigwedge A_i(a_i)] \end{array}$$

which expresses : $A_0, \ldots, A_5$ are the singletons $\{cn-1\}$, $\{\lfloor\sqrt{cn-1}\rfloor\}$,$\{n-1\}$, $\{l\}$, $\{m\}$, $\{L\}$. Then the assertion :

$$cn \models \exists G_1 \ldots \exists G_p (\forall \overline{x} < m)(\forall \overline{t}_1 < L)(\exists \overline{t}_2 < L) \bigwedge \bigvee \pm H(xL + t),$$

with $x \in \overline{x}$, $t \in \{\overline{t}_1, \overline{t}_2\}$, $H \in \{F, \overline{G}, BIT\}$, is equivalent to :

$$\begin{array}{ll} cn \models & \exists G_1 \ldots \exists G_p \, \exists U_0 \exists U_1 \, \exists V_0 \exists V_1 \, \exists A_0 \ldots \exists A_5 \, \exists B : \\ & S(U_0, U_1) \wedge E(V_0, V_1) \wedge T(A_0, \ldots, A_5) \wedge \\ & \forall a_0 \ldots \forall a_5 [\bigwedge A_i(a_i) \to \mathrm{Def}_{bit}(B, a_4, a_5)] \wedge \\ & \forall a_0 \ldots \forall a_5 \forall \overline{x} \forall \overline{t}_1 \exists \overline{t}_2 \; [(\bigwedge A_i(a_i) \wedge \overline{x} < a_4 \wedge \overline{t}_1 < a_5) \to \\ & \qquad ((\overline{t}_2 < a_5) \wedge \bigwedge \bigvee \pm H(xa_5 + t))] \end{array}$$

where H denotes B each time it was denoting BIT in the previous formula. Let $DEF(\overline{U}, \overline{V}, \overline{A}, B)$ denote the formula:

$$S(\overline{U}) \wedge E(\overline{V}) \wedge T(\overline{A}) \wedge \forall \overline{a} [\bigwedge A_i(a_i) \to \mathrm{Def}_{bit}(B, a_4, a_5)]$$

Let $ELIM(F, \overline{G}, \overline{U}, \overline{V}, \overline{A}, B)$ denote the formula:

$$\begin{array}{c} (\forall \overline{a}, \overline{x}, \overline{t}_1)(\exists \overline{t}_2) \\ \{[\bigwedge A_i(a_i) \wedge (\overline{x} < a_4) \wedge (\overline{t}_1 < a_5)] \to [(\overline{t}_2 < a_5) \wedge \bigwedge \bigvee \pm H(xa_5 + t)]\} \end{array}$$

where $H \in \{F, \overline{G}, B\}$, $x \in \overline{x}$ and $t \in \{\overline{t}_1, \overline{t}_2\}$.
For $\overline{U}, \overline{V}, \overline{A}, B$ fixed unary relations in cn such that $cn \models DEF(\overline{U}, \overline{V}, \overline{A}, B)$, we have :

$$\begin{array}{c} cn \models ELIM(F, \overline{G}, \overline{U}, \overline{V}, \overline{A}, B) \\ \text{iff} \end{array}$$

$$\begin{array}{rl} cn \models & (\forall \overline{a}, \overline{x}, \overline{\mathbf{u_x}}, \overline{\mathbf{t}}_\mathbf{1})(\exists \overline{\mathbf{t}}_\mathbf{2})\{[\bigwedge \mathbf{A_i}(\mathbf{a_i}) \wedge (\overline{\mathbf{x}} < \mathbf{a_4}) \wedge (\overline{\mathbf{t}}_\mathbf{1} < \mathbf{a_5}) \wedge \bigwedge_{\mathbf{x} \in \overline{\mathbf{x}}} (\mathbf{u_x} = \mathbf{xa_5})] \\ & \to [(\overline{t}_2 < a_5) \wedge \bigwedge \bigvee \pm H(u_x + t)]\} \end{array}$$

where each $u_x \in \overline{u_x}$ is a new variable associated to $x \in \overline{x}$.

Let us denote :
$\overline{\alpha} = (\overline{a}, \overline{x}, \overline{u_x}, \overline{t}_1)$;

$\phi(\overline{\alpha}) \equiv \bigwedge A_i(a_i) \wedge (\overline{x} < a_4) \wedge (\overline{t}_1 < a_5) \wedge \bigwedge_{x\in\overline{x}}(u_x = xa_5)$;
$\psi(\overline{\alpha}, \overline{t}_2) \equiv (\overline{t}_2 < a_5) \wedge \bigwedge \bigvee \pm H(u_x + t)$.
As $\overline{t}_2$ does not appear in $\overline{\alpha}$, the following equivalence holds :
$cn \models \forall\overline{\alpha}\exists\overline{t}_2\{\phi(\overline{\alpha}) \rightarrow \psi(\overline{\alpha}, \overline{t}_2)\}$
iff
$cn \models \forall\overline{\alpha}\{\phi(\overline{\alpha}) \rightarrow \exists\overline{t}_2\psi(\overline{\alpha}, \overline{t}_2)\}$.

Whence we get:

$$\begin{array}{ll} cn \models & \exists\overline{G}(\forall\overline{x} < m)(\forall\overline{t}_1 < L)(\exists\overline{t}_2 < L) \bigwedge \bigvee \pm H(xL + t) \\ & \text{with } x \in \overline{x},\ t \in \{\overline{t}_1, \overline{t}_2\},\ H \in \{F, \overline{G}, BIT\} \end{array}$$

iff

$$cn \models \quad \exists\overline{G}\, \exists\overline{U}\, \exists\overline{V}\, \exists\overline{A}\, \exists B\, \{\, DEF(\overline{U}, \overline{V}, \overline{A}, B) \wedge \forall\overline{\alpha}[\phi(\overline{\alpha}) \rightarrow \exists\overline{t}_2\psi(\overline{\alpha}, \overline{t}_2)]\,\}$$

To conclude, we just have to see that :
$\phi(\overline{\alpha})$ is in $FO(\exists^*\forall)$ because of the expression of the product seen above;
$\psi(\overline{\alpha}, \overline{t}_2)$ is quantifier free;
hence $\{\forall\overline{\alpha}[\phi(\overline{\alpha}) \rightarrow \exists\overline{t}_2\psi(\overline{\alpha}, \overline{t}_2)]\,\} \in FO(\forall^*\exists^*)$;
at last $DEF(\overline{U}, \overline{V}, \overline{A}, B)$, obtained by conjunction of the $FO(\forall^*\exists^*)$-formulas $S(\overline{U})$, $E(\overline{V})$, $T(\overline{A})$ and $\forall\overline{a}[\bigwedge A_i(a_i) \rightarrow \mathrm{Def}_{bit}(B, a_4, a_5)]$, is itself in $FO(\forall^*\exists^*)$.
Thus the formula $DEF \wedge \forall\overline{\alpha}[\phi(\overline{\alpha}) \rightarrow \exists\overline{t}_2\psi(\overline{\alpha}, \overline{t}_2)]$ is the sought formula Φ'. $\square$

REMARK 4.11. *From now on, we will freely use subformulas such as $x < m$, $Bit(yL + t)$, ... as the abbreviations of existential monadic formulas on $\{+\}$.*

5. Connection between F_w and X_w

Let us now come back to Corollary 3.10. It affirms that a set of words belonging to NLIN can be characterized by an existential second-order formula whose second-order variables are monadic predicates and addition, *via* an encoding of words w into finite structures $\langle cn, F, +\rangle$. Our purpose being to get back to finite structures in which the words w are encoded by their naturally associated predicate X, we make explicit, in this section, the link between F and X.

Before stating the general result, we give the main ideas with an example. Starting with a given word of $\{1, 2\}^*$, we successively compute its associated function f_w and the flattening F of f_w. Then we study the connection between F and the predicate X mentioned above. In particular, we will pass from dyadic notations of integers to binary ones.

5.1. An example. Given two integers k and c, each word $w \in \{1, 2\}^*$ defines a function $f_w : m \rightarrow m$, where m and f_w depend on $n = \text{length}(w)$ and k. The problem is to determine successively:

- the function f_w;
- its flattening F on cn;
- the link between F and the predicate $X \subseteq n$ naturally associated to w, *i.e.* defined by : $\forall i < n$, $X(i)$ iff the i^{th} bit of w is 2.

Let us assume, for instance, that $k = 3$, $c = 4$, and w is the word:
w=112111221222212121211
Then we have, with the notations of 1.2:
$n = \text{length}(w) = 21$, hence $l = \lceil \frac{\log(n+1)}{k} \rceil = 2$; $m = \lceil \frac{n}{l} \rceil = 11$; $L = \lfloor \log(m) + 1 \rfloor = 4$.

Computing f_w. The reading decomposition of w can then be written :

$$w = w_0 {}^\frown w_1 {}^\frown \ldots {}^\frown w_{10}$$

with:

$$\begin{array}{llllll} w_0 = 11\,; & w_1 = 21\,; & w_2 = 11\,; & w_3 = 22\,; & w_4 = 12\,; & w_5 = 22\,; \\ w_6 = 21\,; & w_7 = 21\,; & w_8 = 21\,; & w_9 = 21\,; & w_{10} = 1\,. & \end{array}$$

Denoting w_i^d the integer whose reverse dyadic representation is w_i, we have :

$$\begin{array}{llllll} w_0^d = 3\,; & w_1^d = 4\,; & w_2^d = 3\,; & w_3^d = 6\,; & w_4^d = 5\,; & w_5^d = 6\,; \\ w_6^d = 4\,; & w_7^d = 4\,; & w_8^d = 4\,; & w_9^d = 4\,; & w_{10}^d = 1\,. & \end{array}$$

And the function $f_w : m \to m$ defined by: $\forall i < m : f_w(i) = w_i^d$, may be represented by the following array:

i	0	1	2	3	4	5	6	7	8	9	10
$f_w(i)$	3	4	3	6	5	6	4	4	4	4	1

Computing the flattening F of f_w. Let us now describe the construction of the flattening of f_w on $cn = 4 \times 21 = 84$.
The main work consists in computing $B_L(f_w(i))$ for each $i < m$. The word obtained by the $B_L(f_w(i))$'s concatenation will immediately result in F.
Since f_w takes its values in $\{1, 3, 4, 5, 6\}$, we have just to compute:

$$B_4(1) = 1000 \;;\; B_4(3) = 1100 \;;\; B_4(4) = 0010 \;;\; B_4(5) = 1010 \;;\; B_4(6) = 0110.$$

From which we deduce for instance :

- $B_L(f_w(0)) = B_L(f_w(2)) = 1100$;
- $B_L(f_w(1)) = B_L(f_w(6)) = B_L(f_w(7)) = B_L(f_w(8)) = B_L(f_w(9)) = 0010$.

The word of $\{0,1\}^{cn}$ encoding f_w is therefore the concatenation of the words $B_L(f_w(i))$, in the increasing order of the i's, padded on its right with as many 0's as necessary to obtain a word of length $cn = 84$. Let us call this word w'. We have:

$$w' = 1100\,0010\,1100\,0110\,1010\,0110\,0010\,0010\,0010\,0010\,1000^\frown 0^{40}.$$

Hence, F is the subset of $cn = 84$ given as follows:

$$\boldsymbol{F = \{0, 1, 6, 8, 9, 13, 14, 16, 18, 21, 22, 26, 30, 34, 38, 40\}}.$$

Connection between F and X. The predicate $X \subseteq n$ naturally associated with w is defined by :

$$X(i) \iff \textit{the } i^{th} \textit{ bit of } w \textit{ is } 2.$$

The connection between F and X is quite determined by the connection between w and w'.
Let us write these two words in the following manner:

w	11	21	11	22	12	22	21	21	21	21	1	
w'	1100	0010	1100	0110	1010	0110	0010	0010	0010	0010	1000	0^{40}

We easily see that we pass from w to w' in computing, for each $i < m$, the word $B_L(w_i^d)$.
All in all, our problem consists in the translation from the reverse dyadic representation of an integer into its reverse binary representation of fixed length L. A simple finite transducer automaton can realize this translation. Rather than describing it, we just give here the result of its computation:

for every positive integer x, let us denote $D(x)$ (resp.$B(x)$) its reverse dyadic (resp. binary) representation.
It is easy to show that $\text{length}(B(x)) = \text{length}(D(x))$ or $\text{length}(B(x)) = \text{length}(D(x))+1$.
So for $x \neq 0$, possibly padding $B(x)$ with one 0, we can write :

$$D(x) = d_0 d_1 \dots d_q \in \{1,2\}^*$$
$$B(x) = b_0 b_1 \dots b_q b_{q+1} \in \{0,1\}^*$$

Then we have for every $i \leq q+1$:

$$(b_i = 1) \Leftrightarrow \left(\begin{array}{c} d_0 = d_1 = \ldots = d_i = 1 \\ \text{or} \\ (i = q+1 \text{ or } d_i = 2) \text{ and there exists } j < i \text{ s.t. } d_j = 2 \end{array} \right)$$

Examples :

For $x = 15$: $D(x) = 1111$
$B(x) = 11110$

For $x = 35$: $D(x) = 11211$
$B(x) = 110001$

For $x = 51$: $D(x) = 11212$
$B(x) = 110011$

To translate $D(x)$ into the reverse binary representation of *fixed length* L of x, we just have to pad $B(x)$ on its right with as many 0's as necessary.
This result allows to express the logical connection between F and X, in the general case, in a straightforward way.

5.2. Connection between F and X: the general case. Let w be a word of $\{1,2\}^n$. Let us denote

$$w = d_0^0 \ldots d_0^{l-1} d_1^0 \ldots d_1^{l-1} \ldots d_{m-2}^0 \ldots d_{m-2}^{l-1} d_{m-1}^0 \ldots d_{m-1}^{s-1},$$

its reading decomposition into m words $w_i = d_i^0 \ldots d_i^{l-1}$ for $i < m-2$ and $w_{m-1} = d_{m-1}^0 \ldots d_{m-1}^{s-1}$ $(0 < s \leq l)$. The function $f_w : m \to m$ is then defined by:

$$\forall i < m,\ f_w(i) = w_i^d = \begin{cases} \sum_{j=0}^{l-1} d_i^j 2^j & \text{if } i < m-1 \\ \sum_{j=0}^{s-1} d_{m-1}^j 2^j & \text{if } i = m-1 \end{cases}$$

Every integer w_i^d may be encoded by its reverse binary representation of length L: $b_i^0 \ldots b_i^{L-1}$. Hence f_w may be described by a binary word:

$$w' = b_0^0 \ldots b_0^{L-1} \ldots b_{m-1}^0 \ldots b_{m-1}^{L-1}.0^{cn-mL} \in \{0,1\}^{cn}.$$

Relationship between the words w, w' and the function f_w is therefore expressed: $\forall i < m$, $d_i^0 \ldots d_i^{l-1}$ is the reverse dyadic representation of $f_w(i)$ (replace l by s for $f_w(m-1)$) and $b_i^0 \ldots b_i^{L-1}$ is the reverse binary representation of length L of $f_w(i)$. We deduce: $\forall x < m$, $\forall t < L$:

$$\begin{array}{c} \{b_x^t = 1\} \\ \text{iff} \\ \left\{ \begin{array}{lll} & (x < m-1) & \wedge \left[\begin{array}{ll} & \{(t < l) \wedge (\forall t' \leq t)(d_x^{t'} = 1)\} \\ \vee & \{(t < l) \wedge (d_x^t = 2) \wedge (\exists t' < t)(d_x^{t'} = 2)\} \\ \vee & \{(t = l) \wedge (\exists t' < t)(d_x^{t'} = 2)\} \end{array} \right] \\ \vee & (x = m-1) & \wedge \qquad \text{[the same, replacing } l \text{ by } s] \end{array} \right\} \end{array} \tag{5.6}$$

With the chosen notations, the predicate X naturally encoding w on n and the flattening F of f_w on cn are defined by:

$$\begin{array}{l} \forall x < m, \forall t < l : [n \models X(xl+t)] \Leftrightarrow [d_x^t = 2] \\ \forall x < m, \forall t < l : [cn \models F(xL+t)] \Leftrightarrow [b_x^t = 1] \end{array}$$

The equivalence (5.6) can therefore be written: $\forall x < m$, $\forall t < l$:

$$\begin{array}{c} F(xL+t) \\ \text{iff} \\ \left\{ \begin{array}{lll} & (x < m-1) & \wedge \left[\begin{array}{ll} & \{(t < l) \wedge (\forall t' \leq t) \neg X(xl+t')\} \\ \vee & \{(t < l) \wedge X(xl+t) \wedge (\exists t' < t) X(xl+t')\} \\ \vee & \{(t = l) \wedge (\exists t' < t) X(xl+t')\} \end{array} \right] \\ \vee & (x = m-1) & \wedge \qquad \text{[the same, replacing } l \text{ by } s] \end{array} \right\} \end{array}$$

More precisely, if $\mathcal{X}$ denotes the trivial extension of X to cn (that is, $\mathcal{X} \subseteq cn$ and $\forall i < cn : \mathcal{X}(i) \Leftrightarrow [i < n \wedge X(i)]$), and if $\Phi(x, t, X)$ denotes the formula written between braces in the above equivalence, then we have, $\forall x < m$, $\forall t < L$:

$$[cn \models F(xL+t)] \Leftrightarrow [cn \models \Phi(x, t, X)] \ ,$$

and since $\mathcal{X}$ coincides with X on mL, which is the only part of cn concerned by the formula Φ:

$$[cn \models F(xL+t)] \Leftrightarrow [cn \models \Phi(x, t, \mathcal{X})].$$

Then, a unary predicate $F \subseteq cn$ is the flattening of f_w on cn iff $cn \models D(\mathcal{X}, F)$, where $D(\mathcal{X}, F)$ is the formula which makes explicit the connection between $\mathcal{X}$ and the flattening F of f_w on cn.

$$i.e. \;\; D(\mathcal{X}, F) \equiv \forall \alpha [F(\alpha) \leftrightarrow (\exists x < m)(\exists t < L)\{(\alpha = xL+t) \wedge \Phi(x, t, \mathcal{X})\}]$$

Since we have:

$$[w \in \mathcal{L}] \Leftrightarrow [\langle cn, F, + \rangle \models \exists G_1 \ldots \exists G_q \tilde{\Psi}(F, \overline{G}, +)],$$
where F is the flattening of f_w on cn
(this is Corollary 3.10)

we finally deduce:

$$[w \in \mathcal{L}] \Leftrightarrow [\langle cn, \mathcal{X}, + \rangle \models \exists F \exists G_1 \ldots \exists G_q (D(\mathcal{X}, F) \wedge \tilde{\Psi}(F, \overline{G}))]$$
(since $D(\mathcal{X}, F)$ characterizes the flattening F of f_w on cn.)

Whence, denoting $\hat{\Psi}(\mathcal{X}, F, G_1, \ldots, G_q) \equiv D(\mathcal{X}, F) \wedge \tilde{\Psi}(F, \overline{G})$:

$$[w \in \mathcal{L}] \Leftrightarrow [\langle cn, \mathcal{X}, + \rangle \models \exists F \exists G_1 \ldots \exists G_q \hat{\Psi}(\mathcal{X}, F, G_1, \ldots, G_q)]$$

And at last, denoting G_0 the quantified second-order variable F, we have:

Lemma 5.1. *Let $\mathcal{L}$ be a language on $\Sigma = \{1, 2\}$. If $\mathcal{L} \in$ NLIN, then there exists a sentence $\hat{\Psi} \in FO(\forall^*\exists^*)$ on signature $\{\mathcal{X}, G_0, G_1, \ldots, G_q, +\}$, where $\mathcal{X}$ and the G_i's are unary predicate symbols, $+$ is a ternary predicate symbol, such that:*

$$w \in \mathcal{L} \ \text{ iff } \ \langle cn, \mathcal{X}, + \rangle \models \exists G_0 \ldots \exists G_q \hat{\Psi}(\mathcal{X}, G_0, \ldots, G_q, +)$$

where $\mathcal{X}$ is the trivial extension to cn of the predicate naturally encoding w on n, and $+$ is the predefined addition on cn.

Proof. The assertion $\hat{\Psi} \in FO(\forall^*\exists^*)$ has now to be justified. Since $\hat{\Psi} \equiv D \wedge \tilde{\Psi}$, and since $\tilde{\Psi} \in FO(\forall^*\exists^*)$, it is sufficient to prove that $D \in FO(\forall^*\exists^*)$. But $D \equiv \forall\alpha[F(\alpha) \leftrightarrow (\exists x < m)(\exists t < L)\{(\alpha = xL + t) \wedge \Phi\}]$, can also be written: $D \equiv \forall\alpha[F(\alpha) \leftrightarrow (\forall x < m)(\forall t < L)\{(\alpha = xL + t) \rightarrow \Phi\}]$. Using these two writings and observing that:
- $\alpha = xL + t$ can be replaced by a formula in $FO(\forall^*\exists^*)$ and in $FO(\exists^*\forall^*)$;
- Φ can be written with either a $\forall^*\exists^*$ or an $\exists^*\forall^*$-quantifier pattern,

the reader will easily convince himself or herself that the result holds. □

6. From cn to n

This part focuses on a technical result which is essentially due to Lynch (see Lyn82 or Gra85). Let us suppose that n and c are fixed in $\mathbb{N}$. Let us call *division of R on n* the set of unary predicates $R^* = \{R_0, \ldots, R_{c-1}\}$ defined on n, for all $i < c$, by :

$$\forall y < n \ : \ R_i(y) \Leftrightarrow R(in + y).$$

As each $x < cn$ may be written $x = in + y$, with $i < c$ and $y < n$, we have :

$$\begin{aligned} cn \models \forall x R(x) \Leftrightarrow \;& cn \models (\forall y < n)[R(y) \wedge R(n+y) \wedge \ldots \wedge R((c-1)n+y)] \\ \Leftrightarrow \;& n \models \forall y [R_0(y) \wedge \ldots \wedge R_{c-1}(y)] \end{aligned}$$

i.e.

$$cn \models \forall x R(x) \Leftrightarrow \; n \models \forall y \textstyle\bigwedge_{i<c} R_i(y)$$

In the same way, we have :

$$cn \models \exists x R(x) \Leftrightarrow \; n \models \exists y \textstyle\bigvee_{i<c} R_i(y)$$

On this model, we can build, for each first-order sentence Φ on signature $\{R\}$, a first-order sentence Φ^* on signature $\{R^*\} = \{R_0, \ldots, R_{c-1}\}$, such that :

$$cn \models \Phi(R) \Leftrightarrow n \models \Phi^*(R^*).$$

This result can be generalized to any-arity predicates on cn : if R is an r-ary predicate on cn, we call “division”of R on n the set of r-ary predicates on n, $R^* = \{R_{i_1 \ldots i_r},\ i_1, \ldots, i_r < c\}$, defined, for all $i_1, \ldots, i_r < c$, by :

$$\forall (y_1, \ldots, y_r) \in n^r \ : \ R_{i_1 \ldots i_r}(y_1, \ldots, y_r) \Leftrightarrow R(i_1 n + y_1, \ldots, i_r n + y_r).$$

We can still prove in this case that for each first-order sentence Φ on signature $\{R\}$, there exists a first-order sentence Φ^* on signature $\{R^*\}$, such that :

$$cn \models \Phi(R) \Leftrightarrow n \models \Phi^*(R^*).$$

In the same way, if Φ is a first-order sentence on signature $\{R_1, \ldots, R_s\}$, where each R_i is a r_i-ary predicate on cn, we build a first-order sentence Φ^* on signature $\{R_1^*, \ldots, R_s^*\}$, where each R_i^* is the division of R_i on n (and so, is a set of c^{r_i} r_i-ary predicates on n), such that:

$$cn \models \Phi(R_1, \ldots, R_s) \Leftrightarrow n \models \Phi^*(R_1^*, \ldots, R_s^*).$$

Finally, the bijectivity of the mapping which associates every r-ary predicate on cn to its division $R^* = \{R_{i_1 \ldots i_r},\ i_j < c\}$ on n, allows us to affirm :

Lemma 6.1. *Let Φ be a first-order sentence of signature $\{R_1, \ldots, R_s, P_1, \ldots, P_t\}$ where each R_i is an r_i-ary predicate symbol, and each P_i is a p_i-ary relation on cn. Then there exists a first-order sentence Φ^* on signature $\{R_1^*, \ldots, R_s^*, P_1^*, \ldots, P_t^*\}$, where each R_i^* is a set of c^{r_i} r_i-ary predicates symbols, and the P_i^*'s are the respective divisions of the P_i's, such that:*

$$\langle cn, P_1, \ldots, P_t \rangle \models \exists R_1 \ldots \exists R_s \Phi(R_1, \ldots, R_s, P_1, \ldots, P_t)$$

iff

$$\langle n, P_1^*, \ldots, P_t^* \rangle \models \exists R_1^* \ldots \exists R_s^* \Phi^*(R_1^*, \ldots, R_s^*, P_1^*, \ldots, P_t^*)$$

Moreover, if $\Phi \in FO(\forall^*\exists^*)$, Φ^* can be required to be in $FO(\forall^*\exists^*)$.

Remark 6.2. *The precise construction of Φ^* from Φ, as well as the proof of its membership in $FO(\forall^*\exists^*)$ when Φ has this quantifier pattern, have a tedious expression but are easy to achieve. They are left to the reader.*

Our main result immediately follows from Lemma 5.1, Lemma 6.1, and a simple study of the divisions of $+$ (addition on cn) and $\mathcal{X}$ (trivial extension of the predicate naturally encoding w) on n:

Main Theorem. *If a language $\mathcal{L} \subseteq \{1,2\}^*$ is in NLIN, then there exists a sentence $\Theta \in FO(\forall^*\exists^*)$ on signature $\{X, U_1, \ldots, U_s, +\}$, where X and the U_i's are unary predicate symbols, $+$ is a ternary predicate symbol, such that for every word $w \in \{1,2\}^*$:*

$$[w \in \mathcal{L}] \Leftrightarrow [\langle n, X, + \rangle \models \exists U_1 \ldots \exists U_s \, \Theta(X, \overline{U}, +)]$$

where $+$ is the predefined addition on n and X is the predicate naturally encoding w on n.

Proof. Let Σ, $\mathcal{L}$, w, X and $+$ be like in the theorem hypothesis. Then, from Lemma 5.1, we have:

$$[w \in \mathcal{L}] \Leftrightarrow [\langle cn, \mathcal{X}, + \rangle \models \exists G_0 \ldots \exists G_q \hat{\Psi}(\mathcal{X}, G_0, \ldots, G_q, +)] \ ,$$

where $\mathcal{X}$ is the trivial extension of X to cn. This becomes, with Lemma 6.1:

$$[w \in \mathcal{L}] \Leftrightarrow [\langle n, \mathcal{X}^*, +^* \rangle \models \exists G_0^* \ldots \exists G_q^* (\hat{\Psi})^*(\mathcal{X}^*, G_0^*, \ldots, G_q^*, +^*)] \ ,$$

where $\mathcal{X}^*$ and $+^*$ are the respective divisions of $\mathcal{X}$ and $+$, and each G_i^* is a set of c unary predicate symbols. Replacing $\{G_0^*, \ldots, G_q^*\}$ by $\{U_1, \ldots, U_s\}$, with $s = (q+1)c$, and denoting Ψ^* instead of $(\hat{\Psi})^*$, we get:

$$[w \in \mathcal{L}] \Leftrightarrow [\langle n, \mathcal{X}^*, +^* \rangle \models \exists U_1 \ldots \exists U_s \Psi^*(\mathcal{X}^*, U_1, \ldots, U_s, +^*)] \ ,$$

We just have to write $+^*$ and $\mathcal{X}^*$ in a standard form to obtain the sought result.

Division of $+$ on n. $+$ is the ternary predicate on cn defined by:

$$[cn \models +(x, y, z)] \Leftrightarrow [x + y = z \text{ in } cn].$$

$+^*$ is then the set $\{+^{ijk}, \ i, j, k < c\}$ made up of c^3 ternary predicates on n, defined, for each $(i, j, k) \in c^3$, by:

$$\begin{array}{lll} n \models +^{ijk}(x, y, z) & \text{iff} & cn \models +(in + x, jn + y, kn + z) \\ & \text{iff} & (in + x) + (jn + y) = (kn + z) \text{ in } cn \\ & \text{iff} & n \models \ \{(i + j = k) \wedge (x + y = z)\} \vee \\ & & \qquad \{(i + j + 1 = k) \wedge (x + y - z - 1 = n - 1)\} \end{array}$$

Let us call $\phi_{ijk}(x, y, z)$ this last formula, in which $+$ now denotes the addition on n. Replacing the assertions "$i + j = k$" and "$i + j + 1 = k$" by their truth value (which is perfectly defined for each triple (i, j, k)), we have thus built, for each formula $+^{ijk}(x, y, z)$, an equivalent formula $\phi_{ijk}(x, y, z)$ on n whose signature is the only predicate $+$, namely, the addition on n.

Division of $\mathcal{X}$ on n: $\mathcal{X}$ is the trivial extension of X, i.e. the unary predicate on cn defined by:

$$\forall x < cn \;\; \mathcal{X}(x) \Leftrightarrow [x < n \text{ and } X(x)].$$

So $\mathcal{X}^*$ may be denoted $\mathcal{X}^* = \{\mathcal{X}^0, \ldots, \mathcal{X}^{c-1}\}$, where each $\mathcal{X}^i$ is a unary predicate on n defined by:

$$\begin{array}{rll} n \models \mathcal{X}^i(x) & \text{iff} & cn \models \mathcal{X}(in + x) \\ & \text{iff} & in + x < n \text{ and } n \models X(in + x)] \\ & \text{iff} & i = 0 \text{ and } n \models X(x) \end{array}$$

In other words, $\mathcal{X}^0 = X$ and for every $i > 0$, $\mathcal{X}^i$ is the empty relation of n.

Let Θ be the first-order sentence obtained from Ψ^* as follows:

1. each subformula $+^{ijk}(x, y, z)$ of Ψ^* is replaced by the formula $\phi_{ijk}(x, y, z)$ described above ;

2. each occurrence of $\mathcal{X}^i(x)$ is replaced by $X(x)$ if $i = 0$, and by $\perp$ (false) if $0 < i < c$.

Then Θ is the first-order sentence on the signature $\{X, U_1, \ldots, U_s, +\}$ announced in the statement of the theorem. □

Conclusion and final remarks

We have proved that NLIN, and therefore many NP-problems (the 21 NP-complete problems of Kar72), are definable in Lynch's logic, even with sentences in the class $FO(\forall^*\exists^*)$. In our opinion, this means that the Lynch's purpose announced in Lyn92, namely proving that a natural NP-problem is not expressible in that logic, is a considerably difficult problem, since it would imply that the concerned problem is strictly harder than each of the 21 above-mentioned NP-complete problems.

A natural question is to ask what is exactly definable in Lynch's logic, or in this logic with sentences in $FO(\forall^*\exists^*)$. We cannot prove a converse of our main result, but we easily see that the problems so definable are in $\Sigma_3 TIME(n)$ (a $\Sigma_k TIME(n)$-problem is computable in linear time on an alternating Turing machine with k alternations between existential and universal steps, the first one being existential). This is not far from the known upper bound: NLIN $\subseteq \Sigma_2 TIME(n)$ (guess the entire linear-time bounded RAM computation and check that it is correct; an incorrect computation is recognized within linear time on a nondeterministic Turing machine).

Our main result was proved by using the NLIN logical characterization - proved in this paper - where the input is a unary structure. This result has been refined very recently in Oli96: a set of unary structures is in NLIN iff it is definable in existential second-order logic with unary functions, only one variable and a *conjunction of equalities* (*i.e. without* negation and disjunction); Oli96 also gives a similar exact characterization for each class NTIME$_{RAM}(n^k)$.

Such logical normalizations are useful to prove strong completeness results, e.g. NLIN-completeness, and nonlinear lower bounds : e.g. problem RISA of Gra90 is NLIN-complete and consequently is not in DTIME(n) (see also Gra88, Gra94).

Aknowledgments

We are very grateful to the anonymous referee who read carefully and thoroughly this paper and suggested many corrections and helpful remarks and improvements.

References

Büc60: J.R. Büchi, *Weak second order arithmetic and finite automata*, Z. Math. Logik Grundlagen Math. 6 (1960), pp. 66-92.

Cou96: B. Courcelle, *The expression of graph properties and graph transformations in monadic second-order logic*, In G. Rozenberg, editor, Handbook of graph grammars, vol. 1: Foundations. World Scientific Publisher, 1996.

Fag74: R. Fagin, *Generalized first-order spectra and polynomial-time recognizable sets*, in Complexity of Computations, R.Karp, ed., SIAM-AMS Proc. 7, 1974, pp. 43-73.

Gra84: E. Grandjean, *The spectra of first-order sentence and computational complexity*, SIAM J.Comput., 13 (1984), pp. 356-373.

Gra85: E. Grandjean, *Universal quantifiers and time complexity of random access machines*, Math. Systems Theory, 18 (1985), pp. 171-187.

Gra88: E. Grandjean, *A natural NP-complete problem with a nontrivial lower bound*, SIAM J.Comput.,17 (1988), pp. 786-809.

Gra90: E. Grandjean, *A nontrivial lower bound for an NP problem on automata*, SIAM J. Comput., 19 (1990), pp. 438-451.

Gra90b: E. Grandjean, *First-order spectra with one variable*, J. of Comput. and System Sci., 40 (1990), pp. 136-153.

Gra94: E. Grandjean, *Invariance properties of RAMs and linear time*, Computational Complexity, 4 (1994), pp. 62-106.

Gra94b: E. Grandjean, *Linear time algorithms and NP-complete problems*, SIAM J. Comput., 23 (1994), pp. 573-597.

Gra96: E. Grandjean, *Sorting, linear time and the satisfiability problem*, Annals of Math. and Artificial Intelligence, 16 (1996), pp. 183-236.

Grl90: E. Grädel, *On the notion of linear time computability*, International J. of Foundations of Computer Science, 1 (1990), pp. 295-307.

GrOl95: E. Grandjean, F.Olive, *Monadic Logical Definability of NP-Complete Problems*, CSL'94 Proc. LNCS 933 (1995), pp. 190-204.

Gur84: Y. Gurevich, *Toward logic tailored for computational complexity*, Computation and Proof Theory, (M.M.Richter et. al., eds.).Springer-Verlag Lecture Notes in Math. Vol.1104, pp. 175-216, Springer-Verlag, New York/Berlin, 1984.

Gur86: Y. Gurevich, *Logic and the challenge of computer science*, Current Trends in Theorical Computer Science, (E.Boerger Ed.), pp. 1-55, Computer science, Rockville, MD, 1986.

GuSh89: Y. Gurevich, S. Shelah, *Nearly linear time*, Lect.notes Comput.Sci. 363 (1989), Springer-Verlag, pp. 108-118.

Imm83: N.IMMERMAN, *Languages which capture complexity classes*, 15th ACM Symp. on Theory of Computing , 1983, pp.347-354; SIAM J. Comput., 16, No.4 (1987), 760-778.

Imm86: N. Immerman, *Relational queries computable in polynomial time*, 14th ACM STOC Symp., 1982, pp.147-152. Also appeared in revised form in Information and Control, 68 (1986), pp. 86-104.

Imm89: N. Immerman, *Descriptive and computational complexity*, in J. Hartmanis ed., Computational Complexity Theory, Proc. of AMS Symposia in Appl. Math. 38 (1989), pp. 75-91.

Kar72: R.M. Karp, *Reducibility among combinatorial problems*, IBM Symp. 1972, Complexity of Computers Computations, Plenum Press, New York, 1972.

KoTh90: P.G. Kolaitis, M.N. Thakur, *Logical definability of NP-optimization problems*, Technical report UCSC-CRL-90-48, Computer and Information Sciences, University of California, Santa-Cruz, 1990.

Lyn82: J.F. Lynch, *Complexity classes and theories of finite models*, Math. Systems Theory, 15 (1982), pp. 127-144.

Lyn92: J.F. Lynch, *The quantifier structure of sentences that characterize nondeterministic time complexity*, in Comput. Complexity, 2 (1992), pp. 40-66.

Oli96: F.OLIVE, *Caractérisation Logique des problèmes NP-complets : Robustesse et Normalisation*, PhD. Thesis, 1996, Université de Caen, France.

PaYa91: C. Papadimitriou, M. Yannakakis, *Optimization, Approximation and Complexity*, in J.Comput.System Sci., 43 (1991), pp. 425-440.

Pin95: J.E. Pin, *Logic, semigroups and automata on words*, Annals of Math. and Artificial Intelligence, 16 (1996), pp. 343-384.

Sch78: C.P. Schnorr, *Satisfiability is quasilinear complete in NQL*, J. ACM, 25 (1978), pp. 136-145.

Tho82: W. Thomas, *Classifying regular events in symbolic logic*, J. Comput. System Sci. 25 (1982), pp. 360-376.

Var82: M. Vardi, *Complexity of relational query languages*, 14th ACM Symp. on Theory of Computation, 1982, pp. 137-146.