A Perspective on Lindström Quantifiers and Oracles

Iain A. Stewart*

Department of Mathematics and Computer Science,
Leicester University, Leicester LE1 7RH, U.K.
i.a.stewart@mcs.le.ac.uk
WWW home page: http://www.mcs.le.ac.uk/~istewart

Abstract. This paper presents a perspective on the relationship between Lindström quantifiers in model theory and oracle computations in complexity theory. We do not study this relationship here in full generality (indeed, there is much more work to do in order to obtain a full appreciation), but instead we examine what amounts to a thread of research in this topic running from the motivating results, concerning logical characterizations of nondeterministic polynomial-time, to the consideration of Lindström quantifiers as oracles, and through to the study of some naturally arising questions (and subsequent answers). Our presentation follows the chronological progress of the thread and highlights some important techniques and results at the interface between finite model theory and computational complexity theory.

1 Introduction

Prior to about 1974, the model theory of finite structures had been deemed to be rather uninteresting, essentially because many of the fundamental results of (infinite) model theory, such as the Completeness and Compactness Theorems, fail when only finite structures are allowed (see [17]). This is not to say that there are no interesting results in the finite case: Trakhtenbrot's Theorem [38] and the 0-1 law for first-order logic [12] are two. But such results appeared sporadically and there was no real concerted research effort in this regard. However, all this changed thanks to Fagin's Theorem [10] which tied together finite model theory and computational complexity theory. Since then, finite model theory has witnessed an explosive growth.

Notwithstanding the beauty of Fagin's Theorem, finite model theory still remained relatively dormant until the late seventies and early eighties. Instrumental in those early days was the work of Immerman which provided further links between finite model theory and complexity theory; and, in particular, yielded logical characterizations of numerous complexity classes by extensions of first-order logic using "operators corresponding to problems" (other operators of a very different syntactic nature, such as fixed-point operators, were also

* Supported by EPSRC Grant GR/K 96564.

used to extend first-order logic [20, 39]). These operators are essentially infinite uniform sequences of Lindström quantifiers: Lindström [24, 25] introduced these quantifiers in model theory so as to characterize minimal logics expressing certain properties. Immerman's logical characterizations of complexity classes yielded other things, such as new notions of (complexity-theoretic) reductions and strong (complexity-theoretic) completeness results, and his general approach enabled him to solve a long-standing open question; namely, he showed that the complexity class **NL** is closed under complementation [22] (this problem was solved independently at about the same time by Szelepcsényi [37]).

Whilst Immerman logically characterized complexity classes such as **L**, **NL** and **P** using operators of the form alluded to above, **NP** remained to be so characterized. Such a characterization was explicitly established by Stewart [31]; although unbeknownst to him, it had (essentially) already been obtained by Dahlhaus [5]. It is this logical characterization of **NP** that forms the starting point for the research travels in this paper. In more detail, we intend to show how this logical characterization of **NP** leads to the consideration of Lindström quantifiers as oracles[1] (even though there are very definite differences between the two concepts), and to a logical characterization of the complexity class $\mathbf{L}^{\mathbf{NP}}$. We then examine this characterization in more general settings in two ways: first, we consider analogous logical characterizations of complexity classes of the form $\mathbf{L}^{\mathbf{CC}}$ where **CC** is not necessarily **NP** (that is, we explore the extent to which the methods and techniques involved in the logical characterization of $\mathbf{L}^{\mathbf{NP}}$ are in some sense generic); and, second, we consider whether our logical characterization of $\mathbf{L}^{\mathbf{NP}}$, which happens to be on the class of ordered structures, holds on the class of all structures.

It is not our intention here to provide a systematic and detailed framework within which different oracle access mechanisms for oracle machines of varying resource bounds can be related to the use of Lindström quantifiers in different logics: one such framework has been laid out in [26, 27]. What we aim to do is to show how logical characterizations of **NP** give rise to the study of the relationship between resource-bounded oracle computations and logics involving Lindström quantifiers. Note the wording of the previous sentence: the motivating spark for the research highlighted within this paper is essentially of the form: "*We have a logical characterization of* **NP** *as a fragment of a particular logic. So which natural complexity class, if any, is captured by the full logic*?". That is, the research arose because we wanted to find out which complexity class was captured by a certain logic, and not because we decided to model resource-bounded oracle computations in a logical fashion.

The survey presented herein highlights results in the papers [7, 15, 32, 33]. Whilst we acknowledge that survey papers generally include explicit definitions of all concepts and notions occurring, we occasionally fail to be as detailed as we might be both for reasons of space and because it is really on the results in

[1] We do not mean to imply that this logical characterization of **NP** inspired the first consideration of Lindström quantifiers as oracles: such a consideration was made by Grädel [13].

the aforementioned papers that we wish to focus. There are already a number of surveys in which any such material missing from this paper might be found (notably [28]), as well as the standard finite model theory reference [9]. We assume that the reader is acquainted with the basic notions of complexity theory (as can be found in, for example, [11]). We include sketches of important proofs and we hope that we give a flavour of how current research at the interface between finite model theory and computational complexity theory is progressing. We also pose a number of problems which we believe to be worthy of investigation.

We begin, in Section 2, by giving some basic definitions relevant to the logical characterization of complexity classes before saying more about Lindström quantifiers and how they compare with oracles in Section 3. We exhibit a logical characterization of the complexity class $\mathbf{L}^{\mathbf{NP}}$, on ordered structures, in Section 4, and in Section 5, we examine the logical characterization of other log-space oracle complexity classes. In Section 6, we return to our logical characterization of $\mathbf{L}^{\mathbf{NP}}$ and consider it on the class of all structures.

2 Some basic definitions

We begin by giving some very basic definitions relating to how to consider a complexity class as a class of problems, where by "problem" we mean a set of finite structures (see below). The reader is encouraged to consult [9] and [11] for further details regarding finite model theory and complexity theory, respectively.

In general, a *signature* $\sigma = \langle \underline{R}_1, \underline{R}_2, \ldots, \underline{R}_r, \underline{C}_1, \underline{C}_2, \ldots, \underline{C}_c \rangle$ is a tuple of *relation symbols* $\{\underline{R}_i : i = 1, 2, \ldots, r\}$, with $\underline{R}_i$ of *arity* a_i, and *constant symbols* $\{\underline{C}_i : i = 1, 2, \ldots, c\}$. A (*finite*) *structure* of *size* n over σ is a tuple $\mathcal{A} = \langle \{0, 1, \ldots, n-1\}, R_1, R_2, \ldots, R_r, C_1, C_2, \ldots, C_c \rangle$ consisting of a *universe* $|\mathcal{A}| = \{0, 1, ..., n-1\}$, *relations* $R_1, R_2, \ldots, R_r$ on the universe $|\mathcal{A}|$ of arities $a_1, a_2, \ldots, a_r$, respectively, and *constants* $C_1, C_2, \ldots, C_c$ from the universe $|\mathcal{A}|$. The size of $\mathcal{A}$ is also denoted by $|\mathcal{A}|$. We denote the set of all (finite) structures over σ by STRUCT(σ) (henceforth, we do not distinguish between relations (resp. constants) and relation (resp. constant) symbols, and we assume that all structures are finite and of size at least 2). A *problem* over σ is a subset of STRUCT(σ) which is closed under isomorphism. If Ω is some problem then the signature of Ω is denoted $\sigma(\Omega)$.

We tie together classes of problems and complexity classes as follows. Complexity classes are generally considered to be classes of languages over the alphabet $\{0, 1\}$, recognized by various resource-bounded computing devices (for example, by log-space deterministic Turing machines), whereas problems are (isomorphism closed) sets of finite structures. What we do is to encode, via some (reasonable) *encoding scheme* (see below), our structures so that they might be input to some computational device. For example, suppose that we have some structure $\mathcal{A}$ of size n in which there is a relation R of arity a. We could encode this relation as a string of n^a bits

$$R(0,0,\ldots,0,0), R(0,0,\ldots,0,1), \ldots, R(0,0,\ldots,0,n-1),$$
$$R(0,0,\ldots,1,0), \ldots, R(n-1,n-1,\ldots,n-1,n-1).$$

Any constant of $\mathcal{A}$ could be encoded as a 0-1 string representing its binary representation, and these strings, encoding relations and constants of $\mathcal{A}$, could then be concatenated, perhaps, to obtain an encoding of $\mathcal{A}$. Note that there are numerous (an infinite number, in fact) different encoding schemes we might choose to encode the structure $\mathcal{A}$. Consequently, we can now say that some problem is accepted by, for example, some Turing machine if some encoding of it, according to some fixed encoding scheme, is the language accepted by the Turing machine. By "fixed" we do not mean that the same encoding scheme should be used for every problem, just that the same encoding scheme should be used for every structure of our problem. Note also that because a problem is a set of structures closed under isomorphism, there will be a number of strings encoding the "same" structure. (This is the basic approach adopted in complexity theory when abstract decision problems are equated with languages: see [11]).

We are now in a position to identify a complexity class **CC** with a class of problems L. We identify **CC** with L, and write $\mathbf{CC} = L$ and say that L *captures* **CC**, if, and only if:

- for every problem in L, there is an encoding of this problem in **CC**; and
- for every language in **CC**, there is a problem in L of which this language is the encoding.

We reiterate again that any encoding scheme is only specific to the problem in hand.

Given any language Λ over $\{0,1\}$, we can always obtain a problem Ω of which this language is the encoding; and, moreover, this problem is over the signature $\sigma = \langle M, S \rangle$, where M is a unary relation symbol and S is a binary relation symbol. For any string $\omega \in \Lambda$ of length n, let $\mathcal{A}$ be a σ-structure of size n such that S encodes a successor relation on $\{0, 1, \ldots, n-1\}$, e.g., $\{(0,1),(1,2),(2,3),\ldots,(n-2,n-1)\}$ which yields the ordering $0, 1, 2, \ldots, n-1$, and such that $M(i)$ holds if, and only if, the ith bit of ω, with respect to the ordering given by S, is 1. Note that there are numerous such structures $\mathcal{A}$ corresponding to ω in this way. Let the problem Ω consist of every such σ-structure derived from some string $\omega \in \Lambda$ in this way (and so Ω is closed under isomorphism). As our encoding scheme for Ω, encode any σ-structure $\mathcal{A}$ for which S is a successor relation as the string obtained by reversing the above process, and encode any σ-structure $\mathcal{A}$ for which S is not a successor relation as some fixed string not in Λ (we may assume that there exists such a string). Then, via this encoding scheme, the problem Ω is encoded as the language Λ. We shall return to these "successor" structures presently.

The above discussion suffers from the fact that we have not defined exactly what we mean by an encoding scheme. Rather than becomes embroiled in this issue, let us refer the reader to the discussion in [11] on as to what constitutes a "reasonable" encoding scheme.

We remark that it is usual to say that an input to some Turing machine has size n if the input string consists of n symbols. We always work modulo an encoding scheme and consequently we talk of some structure being input to some Turing machine when strictly we mean that the encoding of this structure

is input. Furthermore, if we say that some structure of size n is input to some Turing machine then "size" refers to the size of the structure and not the length of the encoding.

3 Lindström quantifiers in descriptive complexity

We now show how Lindström quantifiers are used in the logical characterization of complexity classes defined by a model of computation not involving oracles. Prompted by these characterizations, we then examine the possibility of using oracle machines to solve problems definable in other related logics formed using Lindström quantifiers.

3.1 In the absence of oracles

First-order logic, FO, consists of all formulae formed from atomic formulae (over some signature) using the boolean connectives $\wedge$, $\vee$ and $\neg$, and the quantifiers $\forall$ and $\exists$. We write FO to denote the class of first-order formulae and also the class of problems defined by first-order sentences, and do likewise for any other logic. Whilst it is not difficult to see that any problem in FO is in the complexity class **L**, there are problems in **L** that are not in FO; one such being the problem, over the empty signature, consisting of those structures of even size. Consequently, if we are interested in capturing complexity classes such as **L**, **NL**, **P** and **NP** then first-order logic obviously needs to be extended in some fashion.

One natural extension is to *second-order logic*, SO, which is obtained by allowing the existential and universal quantification of new relation symbols (that is, relation symbols not appearing in the underlying signature). For example, if $\sigma_2 = \langle E \rangle$, where E is a binary relation symbol (and so σ_2-structures can be thought of as undirected graphs via "there is an edge from vertex u to vertex v if, and only if, $E(u,v)$ or $E(v,u)$ holds"; and also as directed graphs via "there is an edge from vertex u to vertex v if, and only if, $E(u,v)$ holds") then the problem

$$\begin{aligned}\text{3COL} = \{\mathcal{A} \in \text{STRUCT}(\sigma_2) :\ &\text{the undirected graph with vertex set}\\ &\text{given by } |\mathcal{A}| \text{ and edge set given by the relation } E \text{ can be 3-coloured}\}\end{aligned}$$

is defined by the second-order sentence

$$\begin{aligned}&\exists R \exists W \exists B (\forall x (R(x) \vee W(x) \vee B(x))\\ &\quad \wedge \forall x (\neg(R(x) \wedge W(x)) \wedge \neg(R(x) \wedge B(x)) \wedge \neg(W(x) \wedge B(x)))\\ &\quad \wedge \forall x \forall y ((E(x,y) \vee E(y,x)) \Rightarrow (\neg(R(x) \wedge R(y)) \wedge \neg(W(x) \wedge W(y))\\ &\qquad\qquad \wedge \neg(B(x) \wedge B(y)))))\end{aligned}$$

(where R, W and B are new relation symbols af arity 1).

In a seminal result, Fagin [10] proved that a problem is in **NP** if, and only if, it can be defined by a sentence of *existential second-order logic*, Σ_1^1 (that is, the fragment of SO consisting of formulae of the form

$$\exists T_1 \exists T_2 \ldots \exists T_k \phi,$$

where each T_i is a new relation symbol and ϕ is a first-order formula). Stockmeyer [36] extended Fagin's characterization to show that SO captures the Polynomial Hierarchy, **PH**.

Given the facts that FO $\subset$ **L** and that Σ_1^1 is a rather basic fragment of SO, one still has the problem of capturing complexity classes "below" **NP** such as **L**, **NL** and **P**. One method is to restrict the syntax of Σ_1^1 even further, as was done by Grädel [14], but one can also augment FO with appropriate "operators". We illustrate such an approach with the best known of these operators, TC.

Define the signature $\sigma_{2++} = \langle E, C, D \rangle$, where E is a binary relation symbol and C and D are constant symbols, and define the problem TC as

$$\begin{aligned} \mathrm{TC} = \{\mathcal{A} \in \mathrm{STRUCT}(\sigma_{2++}) :\ & \text{the digraph with vertex set } |\mathcal{A}| \text{ and} \\ & \text{edge set given by the relation } E \text{ contains a path from vertex } C \\ & \text{to vertex } D\}. \end{aligned}$$

Corresponding to the problem TC is an operator of the same name; that is, an infinite uniform, or vectorized, sequence of *Lindström quantifiers* (whilst we do not define here explicitly what a Lindström quantifier is, we hope that the essence of Lindström quantifiers is gleaned from what follows). The logic $(\pm\mathrm{TC})^*[\mathrm{FO}]$, or *transitive closure logic*, is the closure of FO under the usual first-order connectives and quantifiers, and also the operator TC, with TC applied as follows. Given a formula $\phi(\mathbf{x}, \mathbf{y}) \in (\pm\mathrm{TC})^*[\mathrm{FO}]$, where the variables of the k-tuples $\mathbf{x}$ and $\mathbf{y}$, for some k, are all distinct and free in ϕ, the formula Φ defined as $\mathrm{TC}[\lambda \mathbf{x}\mathbf{y}\phi](\mathbf{u}, \mathbf{v})$, where $\mathbf{u}$ and $\mathbf{v}$ are k-tuples of (not necessarily distinct) constant symbols and variables, is also a formula of $(\pm\mathrm{TC})^*[\mathrm{FO}]$, with the free variables of Φ being those variables in $\mathbf{u}$ and $\mathbf{v}$, as well as the free variables of ϕ different from those in the tuples $\mathbf{x}$ and $\mathbf{y}$. If Φ is a sentence then it is interpreted in a structure $\mathcal{A} \in \mathrm{STRUCT}(\sigma)$, where σ is the underlying signature, as follows. We build a digraph with vertex set $|\mathcal{A}|^k$ and edge set

$$\{(\mathbf{a}, \mathbf{b}) \in |\mathcal{A}|^k \times |\mathcal{A}|^k : \phi(\mathbf{a}, \mathbf{b}) \text{ holds in } \mathcal{A}\},$$

and say that $\mathcal{A} \models \Phi$ if, and only if, there is a path in this digraph from vertex $\mathbf{u}$ to vertex $\mathbf{v}$ (the semantics can easily be extended to formulae of $(\pm\mathrm{TC})^*[\mathrm{FO}]$: see, for example, [32] for a more detailed semantic definition). We also denote the fragments of $(\pm\mathrm{TC})^*[\mathrm{FO}]$ where applications of TC do not appear within the scope of a negation sign by $\mathrm{TC}^*[\mathrm{FO}]$, and where at most m applications of TC may be nested by $(\pm\mathrm{TC})^m[\mathrm{FO}]$: the fragment $\mathrm{TC}^m[\mathrm{FO}]$ is as expected. We reiterate that TC is essentially an infinite sequence of Lindström quantifiers $\{\mathrm{TC}_k\}$ where TC_k binds $2k$ free variables in the formula to which it is applied.

In a celebrated result, Immerman [21, 22] captured the complexity class **NL** by the logic $(\pm\mathrm{TC})^*[\mathrm{FO}]$, but only on the "successor" structures, or to give them their popular name *ordered structures*, encountered earlier. More precisely, he assumed that the logic $(\pm\mathrm{TC})^*[\mathrm{FO}]$ has at its disposal a "built-in" binary relation symbol *succ*, different from any symbol of the underlying signature, that is *always* interpreted as a successor relation (i.e., $succ(x, y)$ holds if, and only if, $y = x+1$) on the domain of any structure, and also two constant symbols, 0 and *max*, that denote the least and greatest elements with respect to the relation *succ*: we denote the resulting logic by $(\pm\mathrm{TC})^*[\mathrm{FO}_s]$. Note that sentences of a logic such as $(\pm\mathrm{TC})^*[\mathrm{FO}_s]$ might not define problems, i.e., sets of finite structures closed under isomorphism, as, for example, the successor relation might be used "explicitly" in a sentence as it is in the sentence $E(0, max)$ of $\mathrm{FO}_s(\sigma_2)$. We simply ignore all such sentences and only ever concern ourselves with sentences defining problems.

However, it is undecidable as to whether a sentence of $(\pm\mathrm{TC})^*[\mathrm{FO}_s]$ defines a problem and consequently $(\pm\mathrm{TC})^*[\mathrm{FO}_s]$ does not have a recursive syntax and should not really be called a logic (see [18]). Notwithstanding this remark, we continue to call $(\pm\mathrm{TC})^*[\mathrm{FO}_s]$ a logic on the grounds that one could regard it as the (bona fide) logic $(\pm\mathrm{TC})^*[\mathrm{FO}]$ restricted to the class of ordered structures. But let us press on with our discussion of Immerman's characterization of **NL**.

In fact, Immerman showed that $(\pm\mathrm{TC})^*[\mathrm{FO}_s] = \mathrm{TC}^1[\mathrm{FO}_s] = \mathbf{NL}$ and that every formula in $(\pm\mathrm{TC})^*[\mathrm{FO}_s]$ has a simple normal form; in particular, any problem Ω that is definable by a sentence of $(\pm\mathrm{TC})^*[\mathrm{FO}_s]$ can be defined by one of the form

$$\mathrm{TC}[\lambda \mathbf{x}\mathbf{y}\psi](\mathbf{0}, \mathbf{max}),$$

where: $\mathbf{x}$ and $\mathbf{y}$ are k-tuples of distinct variables, for some k; ψ is a quantifier-free formula of FO_s (actually, a *quantifier-free projection*: see [21]); and $\mathbf{0}$ (resp. $\mathbf{max}$) is the constant symbol 0 (resp. *max*) repeated k times. In such a circumstance we say that there is a *quantifier-free first-order translation with successor* from Ω to TC; and as Ω is an arbitrary problem in **NL** we have that TC is *complete for* **NL** *via quantifier-free first-order translations with successor* (clearly, we may have other translations such as first-order translations with, or without, successor). Such a result sharpens the well-known result that TC is complete for **NL** via log-space reductions [29]. An important point to note is that adopting such a logical viewpoint to complexity theory yields techniques for proving problems complete for some complexity class that were hitherto unavailable (see [34]) and scope for showing that traditional complete problems might not remain complete under restricted logical reductions (see [1]).

One can augment FO (or FO_s) with an operator (or operators) such as TC corresponding to *any* problem (or problems) and examine the class of problems so captured. A variety of such logics have been formed and many well-known complexity classes, including **L**, **NL**, **P** and **NP**, consequently captured (see, for example, the presentation and references in [35]). Of particular interest to us is the logic formed by extending FO_s using an operator corresponding to the

problem

$$\mathrm{HP} = \{\mathcal{A} \in \mathrm{STRUCT}(\sigma_{2++}) : \text{ the digraph with vertex set given by } |\mathcal{A}| \text{ and edge set given by the relation } E \text{ has a Hamiltonian path from vertex } C \text{ to vertex } D\}.$$

Theorem 1. [5, 31] *A problem is in* **NP** *if, and only if, it can be defined by a sentence of the logic* $HP^*[FO_s]$.

In fact, Theorem 1 holds in the absence of the built-in successor relation [5], and there is a normal form result analogous to that for $\mathrm{TC}^*[\mathrm{FO}_s]$, above [31]. Obviously a more satisfactory result would be that $\mathbf{NP} = (\pm\mathrm{HP})^*[\mathrm{FO}_s]$. However, this would yield as an immediate corollary that **NP** = **co-NP**! Consequently, we are left with the question: "*Can we characterize the class of problems* $(\pm HP)^*[FO_s]$ *in the traditional* (*Turing-machine-based*) *setting*?". Note that such a question does not arise with regard to the analogous logical characterizations of **L**, **NL** and **P** as these complexity classes are closed under complementation.

3.2 Modelling Lindström quantifiers using oracles

Before turning to the question just stated, let us consider the possibilities of modelling applications of Lindström quantifiers using oracles. For concreteness, let us consider a sentence ϕ of the logic $(\pm\mathrm{HP})^*[\mathrm{FO}_s]$. There are some points worthy of note, which we return to presently.

(a) There might be a number of applications of HP within ϕ and some of these applications might be nested.
(b) When deciding whether some appropriate structure, of size n, satisfies ϕ, every application of HP involves the building of a digraph of size n^k, for some k.
(c) In constructing ϕ, the first application of the operator HP (in the process of constructing ϕ) is to a first-order formula.

Let us consider a typical standard oracle-machine computation: for concreteness, consider a log-space deterministic oracle Turing machine (DOTM) where the oracle is the problem HP. In a computation by such a Turing machine, oracle strings are (repeatedly) written to the unique write-only oracle tape and when the oracle consultation state is entered, whether the string written on the oracle tape is in the oracle (i.e., whether the string encodes a digraph and two vertices such that there is a Hamiltonian path in the digraph from the first vertex to the second) is reflected in the new state adopted by the machine. The oracle tape is cleared, with the oracle head subsequently positioned over the leftmost cell of the oracle tape; and all this happens in one move of the machine (see, for example, [4] for more on different types of oracle machines). In comparison with the three points remarked upon above, note the following.

(*a*) Whilst there are a number of different oracle access mechanisms in the literature (see, for example, [4]), it is usually the case that oracle calls can not be nested (and this is certainly the case for the standard notion of a log-space DOTM with an **NP** oracle).
(*b*) The lengths of different oracle strings queried in a particular computation might vary dramatically.
(*c*) The resources used to build a specific oracle query in such a computation can be log-space.

So whilst there may be an intuitive link between Lindström quantifiers and oracles, there are a number of points of divergence.

4 Characterizing Hamiltonian path logic

As it happens, the points of divergence mentioned in the previous section can often be reconciled. The complexity class $\mathbf{L}^{\mathbf{CC}}$ consists of those problems accepted by a standard log-space DOTM with a **CC** oracle.

Theorem 2. [33] *A problem is in* $\mathbf{L}^{\mathbf{NP}}$ *if, and only if, it can be defined by a sentence of the logic* $(\pm HP)^*[FO_s]$. *Moreover, any problem in* $\mathbf{L}^{\mathbf{NP}}$ *can be defined by a sentence of* $(\pm HP)^*[FO_s]$ *of the form*

$$\exists z_1 \exists z_2 \ldots \exists z_k (HP[\lambda \mathbf{x}\mathbf{y}\phi](\mathbf{0}, \mathbf{max}) \wedge \neg HP[\lambda \mathbf{x}'\mathbf{y}'\phi'](\mathbf{0}, \mathbf{max}))$$

where ϕ *and* ϕ' *are quantifier-free first-order* (*actually, quantifier-free projections*) *and* $z_1, z_2, \ldots, z_k$ *are free in* ϕ *and* ϕ'.

Proof. (*Sketch*) Let Ω be some problem accepted by the log-space DOTM M with an **NP** oracle. Without loss of generality, we may assume that the oracle is HP (essentially because HP is complete for **NP** via 1-L *reductions* [19]). Also, by a result of Wagner [40], we may assume that on any input structure of size n, M makes $O(\log n)$ oracle queries.

Consider a typical computation of M on some input structure of size n. There are some answers (0 or 1) to queries associated with this computation, and as there are $O(\log n)$ queries, we can encode these answers using the binary representations of k values $v_1, v_2, \ldots, v_k \in \{0, 1, \ldots, n-1\}$, for some k. If σ is the underlying signature of Ω, denote by σ' the signature σ with k additional constant symbols $C_1, C_2, \ldots, C_k$.

We shall define a log-space DOTM M' that takes σ'-structures as input and which accepts some σ'-structure $\mathcal{A}'$ if, and only if, M accepts $\mathcal{A} = \mathcal{A}'|_\sigma$ (i.e., the σ-structure obtained from $\mathcal{A}'$ by removing the constants $C_1^{\mathcal{A}'}, C_2^{\mathcal{A}'}, \ldots, C_k^{\mathcal{A}'}$) with the oracle answers as given by the constants $C_1^{\mathcal{A}'}, C_2^{\mathcal{A}'}, \ldots, C_k^{\mathcal{A}'}$.

The DOTM M' proceeds as follows.

> Simulate the action of M on $\mathcal{A}$ except that instead of writing on the oracle tape, use the bits of $C_1^{\mathcal{A}'}, C_2^{\mathcal{A}'}, \ldots, C_k^{\mathcal{A}'}$ as the answers to (non-existent) queries.

If the simulation is rejecting then reject else:
build, and query, a triple (G, a, b), where G is a digraph and a and b are distinct vertices of G, with the property that: if
$(G_1, a_1, b_1), (G_2, a_2, b_2), \ldots, (G_m, a_m, b_m)$
are the queries made by M on input $\mathcal{A}$ for which the answer (according to $C_1^{\mathcal{A}'}, C_2^{\mathcal{A}'}, \ldots, C_k^{\mathcal{A}'}$) is "yes" then $(G, a, b) \in$ HP if, and only if, $(G_1, a_1, b_1), (G_2, a_2, b_2), \ldots, (G_m, a_m, b_m) \in$ HP;
build, and query, a triple (G', a', b') with the property that: if
$(G'_1, a'_1, b'_1), (G'_2, a'_2, b'_2), \ldots, (G'_{m'}, a'_{m'}, b'_{m'})$
are the queries made by M on input $\mathcal{A}$ for which the answer (according to $C_1^{\mathcal{A}'}, C_2^{\mathcal{A}'}, \ldots, C_k^{\mathcal{A}'}$) is "no" then $(G', a', b') \notin$ HP if, and only if, $(G'_1, a'_1, b'_1), (G'_2, a'_2, b'_2), \ldots, (G'_{m'}, a'_{m'}, b'_{m'}) \notin$ HP.
If the answers to these two queries are "yes" and "no", respectively, then accept otherwise reject.

The triples (G, a, b) and (G', a', b') can actually be built from $(G_1, a_1, b_1), (G_2, a_2, b_2), \ldots, (G_m, a_m, b_m)$ and $(G'_1, a'_1, b'_1), (G'_2, a'_2, b'_2), \ldots, (G'_{m'}, a'_{m'}, b'_{m'})$ using 1-L reductions [31], and so M' is indeed a log-space DOTM.

We can now encode the computation of M' so that there are two first-order quantifier-free (actually, quantifier-free projections) σ'-formulae $\phi(\mathbf{x}, \mathbf{y})$ and $\phi'(\mathbf{x}', \mathbf{y}')$ with the property that

$$\mathcal{A}' \models \mathrm{HP}[\lambda \mathbf{x}\mathbf{y}\phi](\mathbf{0}, \mathbf{max}) \text{ if, and only if, there is a Hamiltonian path in the digraph } G \text{ from } a \text{ to } b$$

and

$$\mathcal{A}' \models \mathrm{HP}[\lambda \mathbf{x}'\mathbf{y}'\phi'](\mathbf{0}, \mathbf{max}) \text{ if, and only if, there is a Hamiltonian path in the digraph } G' \text{ from } a' \text{ to } b'.$$

Consequently, Ω can be defined by a sentence of the form

$$\exists z_1 \exists z_2 \ldots \exists z_k (\mathrm{HP}[\lambda \mathbf{x}\mathbf{y}\phi](\mathbf{0}, \mathbf{max}) \wedge \neg \mathrm{HP}[\lambda \mathbf{x}'\mathbf{y}'\phi'](\mathbf{0}, \mathbf{max})).$$

Our only remark with regard to the encoding of the computation of M is that $\phi(\mathbf{x}, \mathbf{y})$ and $\phi'(\mathbf{x}', \mathbf{y}')$ do not describe the digraphs G and G' directly. We actually describe these digraphs using formulae of $(\pm\mathrm{DTC})^*[\mathrm{FO}_s]$ where the problem DTC is defined as

$$\mathrm{DTC} = \{\mathcal{A} \in \mathrm{STRUCT}(\sigma_{2++}) : \text{ the digraph with vertex set } |\mathcal{A}| \text{ and edge set given by the relation } E \text{ contains a deterministic path from vertex } C \text{ to vertex } D\},$$

with a *deterministic path* between two vertices in a digraph being one where every vertex on the path, except perhaps the last, has out-degree 1 in the digraph (Immerman captured **L** by the logic $(\pm\mathrm{DTC})^*[\mathrm{FO}_s]$ [21]). We then apply the

operator HP to these formulae and manipulate the resulting formulae to get our normal form.

The fact that any problem definable in $(\pm\mathrm{HP})^*[\mathrm{FO}_s]$ can be accepted by a log-space DOTM with an **NP** oracle is relatively straightforward to prove. □

The proof of Theorem 2 yields a number of corollaries regarding the relationship between fragments of $(\pm\mathrm{HP})^*[\mathrm{FO}_s]$ and complexity classes defined by bounding the number of queries made by log-space and polynomial-time oracle machines to an **NP** oracle. We mention one of these corollaries below as an illustration (for others see [32, 33]).

The Boolean Hierarchy, **BH**, is defined as follows.

$\mathbf{NP}(0) = \mathbf{P}$;
$\mathbf{NP}(1) = \mathbf{NP}$;
$\mathbf{NP}(2i) = \{X \cap Y : X \in \mathbf{NP}(2i-1), Y \in \mathbf{co\text{-}NP}\}$;
$\mathbf{NP}(2i+1) = \{X \cup Y : X \in \mathbf{NP}(2i), Y \in \mathbf{NP}\}$,

for $i \geq 0$, and $\mathbf{BH} = \cup\{\mathbf{NP}(i) : i = 0, 1, \ldots\}$. For any function $f(n)$, $\mathbf{L}^{\mathbf{NP}}[O(f(n))]$ consists of those problems accepted by a log-space DOTM which makes $O(f(n))$ queries to its **NP** oracle on every input of size n. For any logic L, the logic $Bool(L)$ consists of the closure of L under $\wedge$, $\vee$ and $\neg$. Theorem 2, a simple modification of its proof and a result in [40] (stating that if $\mathbf{L}^{\mathbf{NP}}[O(1)] = \mathbf{L}^{\mathbf{NP}}$ then the Polynomial Hierarchy collapses to $\mathbf{L}^{\mathbf{NP}^{\mathbf{NP}}}$) yield the following corollary.

Corollary 1. [33] *$Bool(HP^*[FO_s]) = \mathbf{L}^{\mathbf{NP}}[O(1)]$; and so $\mathbf{BH} = \mathbf{L}^{\mathbf{NP}}[O(1)]$. If $Bool(HP^*[FO_s]) = (\pm HP)^*[FO_s]$ then the Polynomial Hierarchy collapses to $\mathbf{L}^{\mathbf{NP}^{\mathbf{NP}}}$.* □

The fact that $\mathbf{BH} = \mathbf{L}^{\mathbf{NP}}[O(1)]$ was proved in [23].

Write $\mathbf{L}_{||}^{\mathbf{NP}}[O(f(n))]$ to denote the complexity class consisting of all those problems accepted by log-space DOTMs which make, on an input of size n, $O(f(n))$ queries to an **NP** oracle such that all queries are made "in parallel", i.e., intuitively, *all* queries are computed before any is made. As established in [40], if either $\mathbf{L}^{\mathbf{NP}} = \mathbf{L}_{||}^{\mathbf{NP}}[O(\log n)]$ or $\mathbf{L}_{||}^{\mathbf{NP}}[O(\log n)] = \mathbf{L}^{\mathbf{NP}}[O(1)]$ then the Polynomial Hierarchy collapses. Thus, adhering to the current beliefs in complexity theory, it is unlikely that either of these two events happens. This prompts the question as to whether there is a logical characterization of the complexity class $\mathbf{L}_{||}^{\mathbf{NP}}[O(\log n)]$ as a fragment of $(\pm\mathrm{HP})^*[\mathrm{FO}_s]$ (neither $(\pm\mathrm{HP})^*[\mathrm{FO}_s]$ nor $Bool(\mathrm{HP}^*[\mathrm{FO}_s])$ seem likely to suffice).

5 Characterizations of other log-space oracle classes

Whilst Theorem 2 resolves the question posed earlier regarding identifying the logic $(\pm\mathrm{HP})^*[\mathrm{FO}_s]$ with a traditional complexity class, it is rather specific in that it deals only with the operator HP; and there is nothing particularly special about HP beyond the fact that it was involved in the first characterization of

NP by what one might call a "Lindström logic". However, from Theorem 2, an analogous result holds for any operator for which the corresponding problem is complete for **NP** via quantifier-free first-order translations with successor (such as 3COL: see [30]).

Such an observation led Gottlob to consider the more general question: "*If Ω is complete for some complexity class* **CC** *via first-order translations with successor, under what circumstances can we deduce that* $(\pm\Omega)^*[FO_s] = \mathbf{L}^{\mathbf{CC}}$?", and in [15], Gottlob provided a precise answer which revolves around the concept of smoothness; where a complexity class **CC** is *smooth* if, and only if, $\mathbf{L}^{\mathbf{CC}}(\mathbf{CC}) = \mathbf{L}^{\mathbf{CC}}$, i.e., the class of problems that are $\mathbf{L}^{\mathbf{CC}}$ many-one reducible to a problem in **CC** coincides with $\mathbf{L}^{\mathbf{CC}}$.

Theorem 3. [15] *Let* **CC** *be some complexity class that is closed under log-space reductions, and let Ω be some problem that is complete for* **CC** *via first-order translations with successor. Then* $(\pm\Omega)^*[FO_s] = \mathbf{L}^{\mathbf{CC}}$ *if, and only if,* **CC** *is smooth.*

Proof. (*Sketch*) For simplicity, assume that Ω is over the signature σ_2 (the general case is similar). We begin by showing that if **CC** is smooth then any problem Γ in $(\pm\Omega)^*[\mathrm{FO}_s]$ is in $\mathbf{L}^{\mathbf{CC}}$. We do this by induction on the length of the sentence defining Γ. The only non-trivial case is when this sentence Φ is of the form

$$\Omega[\lambda \mathbf{x}\mathbf{y}\phi],$$

where $|\mathbf{x}| = |\mathbf{y}| = k$, for some k, and where the predicate defined by $\phi(\mathbf{x}, \mathbf{y})$ can be checked in $\mathbf{L}^{\mathbf{CC}}$.

Let $\mathcal{A}$ be some $\sigma(\Gamma)$-structure of size n. In order to ascertain whether $\mathcal{A} \models \Phi$, we "build" a σ_2-structure $\mathcal{B}$ whose universe is $|\mathcal{A}|^k$ and whose "binary relation" on $|\mathcal{A}|^k$ contains the pair $(\mathbf{u}, \mathbf{v})$ if, and only if, $\phi(\mathbf{u}, \mathbf{v})$ holds in $\mathcal{A}$. We then check to see whether this structure $\mathcal{B}$ is such that $\mathcal{B} \in \Omega$. Thus, the problem Γ is $\mathbf{L}^{\mathbf{CC}}$ many-one reducible to a problem in **CC**, i.e., $\Gamma \in \mathbf{L}^{\mathbf{CC}}$ because **CC** is smooth.

Now we show that if **CC** is smooth then any problem in $\mathbf{L}^{\mathbf{CC}}$ can be defined by a sentence of $(\pm\Omega)^*[\mathrm{FO}_s]$. In the proof of Theorem 2, we used specific properties of the problem HP with regard to 1-L reductions in order to replace a number of oracle queries with just two oracle queries. However, we are now working in a much more general context and, consequently, such problem-specific properties are not at our disposal.

Let M be some log-space DOTM with a **CC** oracle that accepts the problem Γ. Consider a computation of M on the $\sigma(\Gamma)$-structure $\mathcal{A}$ of size n. We may clearly assume that no instantaneous description (ID) occurring in a computation of M is ever repeated in this computation (for us, an ID of M consists of the state, the contents of the work-tape, the work-head position and the input-head position at any particular instant of a computation of M) and that whenever M terminates (which it does on every input), the oracle tape is empty. Note that because M is a log-space DOTM we can:

- encode an ID of M as k values $v_1, v_2, \ldots, v_k \in \{0, 1, \ldots, n-1\}$, for some k;

- "decode" a given k-tuple $\mathbf{u} \in \{0, 1, \ldots, n-1\}^k$ so as to decide in log-space whether it really encodes a legitimate ID of M on some input structure of size n.

Henceforth, we assume that all k-tuples over $\{0, 1, \ldots, n-1\}$ encode potential IDs in some computation of M on an input structure of size n; as, by above, this property can be checked in log-space.

Let $\mathbf{u}$ and $\mathbf{v}$ be IDs of M on input $\mathcal{A}$ such that when M is started in ID $\mathbf{u}$, the next ID it reaches where the oracle tape is empty is $\mathbf{v}$. Then we say that $\mathbf{v}$ is the *direct successor* of $\mathbf{u}$. Additionally, if there is no oracle computation between $\mathbf{u}$ and $\mathbf{v}$, we say that $\mathbf{v}$ is the *strict successor* of $\mathbf{u}$.

Consider the following two problems.

> Problem A Given two IDs $\mathbf{u}$ and $\mathbf{v}$ of M and the input structure $\mathcal{A}$, $\mathbf{u}$ and $\mathbf{v}$ are such that either: $\mathbf{v}$ is the strict successor of $\mathbf{u}$; or $\mathbf{v}$ is the direct successor of $\mathbf{u}$, there is an oracle query between $\mathbf{u}$ and $\mathbf{v}$ and the answer to this oracle query is "yes".
> Problem B Given two IDs $\mathbf{u}$ and $\mathbf{v}$ of M and the input structure $\mathcal{A}$, $\mathbf{v}$ is the direct successor of $\mathbf{u}$, there is an oracle query between $\mathbf{u}$ and $\mathbf{v}$ and the answer to this oracle query is "no".

Then Problem A is log-space reducible to a problem in **CC**, and Problem B is log-space reducible to a problem in **co-CC**. Thus, by hypothesis, Problems A and B can be defined by $\sigma(\Gamma)$-formulae of $(\pm\Omega)^*[\mathrm{FO}_s]$ of the form

$$\Omega[\lambda \mathbf{x}\mathbf{y}\phi(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{w})] \text{ and } \neg\Omega[\lambda \mathbf{x}'\mathbf{y}'\phi'(\mathbf{x}', \mathbf{y}', \mathbf{z}, \mathbf{w})],$$

respectively, where ϕ and ϕ' are formulae of FO_s (the tuples of free variables $\mathbf{z}$ and $\mathbf{w}$ encode the IDs). Consequently, there is a $\sigma(\Gamma)$-formula $\Phi(\mathbf{z}, \mathbf{w})$ of $(\pm\Omega)^*[\mathrm{FO}_s]$ of the form

$$\Omega[\lambda \mathbf{x}\mathbf{y}\phi] \wedge \neg\Omega[\lambda \mathbf{x}'\mathbf{y}'\phi']$$

which holds in some structure $\mathcal{A}$ for some IDs $\mathbf{u}$ and $\mathbf{v}$ if, and only if, $\mathbf{v}$ is the direct successor of $\mathbf{u}$ in the computation of M on input $\mathcal{A}$.

We can now use the operator DTC and our formula $\Phi(\mathbf{z}, \mathbf{w})$ to define the problem consisting of all those $\sigma(\Gamma)$-structures $\mathcal{A}$ such that on input $\mathcal{A}$, there exists an accepting ID that can be reached from the initial ID in the digraph defined by $\Phi(\mathbf{z}, \mathbf{w})$; in other words, we can define the problem Γ. However, as $\mathbf{L} \subseteq \mathbf{CC}$, we can eliminate any applications of the operator DTC at the expense of introducing applications of the operator Ω. Hence, Γ can be defined by a sentence of the logic $(\pm\Omega)^*[\mathrm{FO}_s]$ as required (and the defining sentence just constructed is actually in $(\pm\Omega)^2[\mathrm{FO}_s]$).

In order to prove our theorem, we need to show that if $(\pm\Omega)^*[\mathrm{FO}_s] = \mathbf{L}^{\mathbf{CC}}$ then **CC** is smooth. Suppose that $(\pm\Omega)^*[\mathrm{FO}_s] = \mathbf{L}^{\mathbf{CC}}$. Let the problem Γ be $\mathbf{L}^{\mathbf{CC}}$ many-one reducible to the problem $\Theta \in \mathbf{CC}$. By hypothesis, this $\mathbf{L}^{\mathbf{CC}}$ many-one reduction can be defined by a formula of $(\pm\Omega)^*[\mathrm{FO}_s]$, and the problem Θ can be defined by a sentence of $(\pm\Omega)^*[\mathrm{FO}_s]$. Hence Γ can be defined by a sentence of $(\pm\Omega)^*[\mathrm{FO}_s]$, and so **CC** is smooth. □

Whilst Theorem 3 gives an answer to the question posed at the beginning of this section, it also provokes the question: "*How can we tell whether a complexity class is smooth or not*?". Indeed, for all we know *all* complexity classes might be smooth. The most transparently smooth complexity class is **L** as it is well known that the composition of two log-space machines can be replaced with a log-space machine. Essentially, whenever the second machine (in the concatenation) needs some input bit, it computes the input bit required, using log-space, and proceeds accordingly. The important point to note is that it need not "remember" the whole of the output from the first machine (of the concatenation), which might be of polynomial size.

If we were to naively attempt to replicate this construction with the first machine, M_1, a $\mathbf{L}^{\mathbf{CC}}$ machine and the second machine, M_2, a **CC** machine, so as to replace the two machines with one $\mathbf{L}^{\mathbf{CC}}$ machine, M, then (intuitively) we run into problems. Suppose that M is in the process of building some oracle string to query, but half way through this building process it discovers that it needs to compute some input bit to its simulation of M_2, i.e., an output bit from M_1. Naively, it would perform a simulation of M_1; but this might not be possible as M_1 might wish to query some oracle strings in this computation, and the oracle tape of M is blocked with a half-built query already! In fact, there *do* exist complexity classes which are closed under log-space reductions and which are not smooth although we do not prove this fact here (but simply refer the reader to [15]).

Having satisfied ourselves that not all (reasonable) complexity classes are smooth, let us look at some complexity classes which are smooth; and, in particular, at some criteria for smoothness. Many complexity classes (especially deterministic ones) can easily shown to be smooth simply by working through the definition of smoothness with the particular complexity class: such classes include **L**, **POLYLOGSPACE** and **P**. Other complexity classes can easily be shown to be smooth by applying the following lemma; but first two definitions. Reverting back to the traditional definition of a complexity class as a class of languages, we say that a complexity class **CC** is closed under *marked union* if for every two languages Λ_1 and Λ_2 in **CC**, the marked union of Λ_1 and Λ_2, namely the language obtained by inserting a 0 before every string of Λ_1 and a 1 before every string of Λ_2, and then taking the union of the resulting two languages, is also in **CC**. The complexity class $\mathbf{P}^{\mathbf{CC}}$ consists of those problems accepted by a standard polynomial-time DOTM with a **CC** oracle.

Lemma 1. [15] *If* $\mathbf{P}^{\mathbf{CC}} = \mathbf{CC}$ *and* **CC** *is closed under marked union then* **CC** *is smooth.* □

The complexity classes Δ_i^p, for $i \geq 1$, of the Polynomial Hierarchy, and the complexity classes **PSPACE** and k-**EXPTIME**, for $k \geq 1$, can be shown to be smooth by applying Lemma 1.

Other sufficient criteria for smoothness exist. A complexity class is *closed under conjunctions* if, and only if, for any finite set of instances of some problem $\Gamma \in \mathbf{CC}$, the problem of deciding whether every one of these instances is in Γ

is also in **CC**. A problem Ω is **NP**-*reducible* to some problem Γ if, and only if, there is a non-deterministic polynomial-time transducer M such that on input some $\sigma(\Omega)$-structure $\mathcal{A}$, there is at least one computation of M on $\mathcal{A}$ which outputs a $\sigma(\Gamma)$-structure $\mathcal{B}$ such that $\mathcal{B} \in \Gamma$.

Lemma 2. [15] *If the complexity class* **CC** *is closed under* **NP**-*reductions, under conjunctions and under marked union then* **CC** *is smooth.* □

The complexity classes Σ_i^p and Π_i^p, for $i \geq 1$, of the Polynomial Hierarchy, and the complexity classes **NEXPTIME** and **co-NEXPTIME** can be shown to be smooth by applying Lemma 2.

The criteria for smoothness above, whilst widely applicable, do not seem to be strong enough to show that every smooth complexity class is in fact smooth. The most notable example is the complexity class **NL** for which smoothness follows from results due to Immerman [21], and for which neither of Lemmas 1 nor 2 seem to be of any use. It would be interesting to derive other criteria from which the smoothness of (complexity classes such as) **NL** could be deduced.

Let us end this section by remarking that the proof of Theorem 2 also yielded a normal form result, whereas no analogous normal form result has been forthcoming from the proof of Theorem 3. Perusal of the proof of Theorem 2 yields that the completeness of HP via 1-L reductions played a role; and this might imply that normal form results could be specific to particular problems. However, Gottlob has established general criteria for determining when a normal form result, analogous to that in Theorem 2, exists.

Theorem 4. [15] *Let* **CC** *be some complexity class that is closed under* **NP**-*reductions, conjunctions and marked union, and let Ω be some problem that is complete for* **CC** *via first-order translations with successor. Then every problem in* $\mathbf{L}^{\mathbf{CC}} = (\pm\Omega)^*[FO_s]$ *can be defined by a sentence of the form*

$$\exists z_1 \exists z_2 \ldots \exists z_k (\Phi \wedge \neg \Phi'),$$

where Φ and Φ' are formulae of $\Omega^1[FO_s]$ formed by applying the operator Ω to appropriate first-order formulae and where $z_1, z_2, \ldots, z_k$ are free in Φ and Φ'. □

The proof of Theorem 4 is similar to the proof of Theorem 2 except that it uses the fact that a complexity class closed under conjunctions and **NP**-reductions is necessarily closed under conjunctions and disjunctions (defined similarly) to obviate the need for the oracle to be complete for some complexity class via 1-L reductions. It would be interesting to investigate whether there is any sort of relationship between a complexity class having a complete problem via 1-L reductions and a complexity class being closed under operations such as those in Theorem 4.

6 In the absence of successor

As remarked earlier, the motivating theorem for the thread of research presented in this paper, Theorem 1, holds in the absence of the built-in successor relation.

Consequently, it is natural to ask whether the same can be said of the subsequent results; in particular, Theorem 2. Dawar, Gottlob and Hella have given partial answers in this regard, as we now explain; but first we require some definitions.

We denote by:

- L^k the fragment of FO which consists of those formulae whose variables, both bound and free, are among $x_1, x_2, \ldots, x_k$;
- $L^k_{\infty\omega}$ the closure of L^k under the operations of conjunction and disjunction applied to arbitrary (finite and infinite) sets of formulae (with obvious semantics);
- $L^\omega_{\infty\omega}$ the union $\bigcup_{k=0}^{\infty} L^k_{\infty\omega}$.

The logic $L^\omega_{\infty\omega}$ was introduced by Barwise [3] and is known as *bounded-variable infinitary logic*. It plays an important role in finite model theory (see, for example, [9]).

However, we introduce $L^\omega_{\infty\omega}$ here for a specific reason. Bounded-variable infinitary logic can be extended by a set of Lindström quantifiers Ω (possibly infinite and not necessarily in the form of uniform sequences corresponding to problems), just as FO was (and with similar semantics), to yield the logic $(\pm\Omega)^*[L^\omega_{\infty\omega}]$. Dawar and Hella [8] have established certain properties of logics of the form $(\pm\Omega)^*[L^\omega_{\infty\omega}]$, or more specifically, of fragments of such logics where the number of quantifiers of Ω appearing in any formula is finite, which are of direct relevance to the investigations presented here.

Let $\mathcal{A}$ be some structure and let $\mathbf{u} = (u_1, u_2, \ldots, u_k) \in |\mathcal{A}|^k$. The *basic equality type* of $\mathbf{u}$ is the formula

$$\bigwedge_{(i,j)\in S} (x_i = x_j) \wedge \bigwedge_{(i,j)\in T} (\neg(x_i = x_j)),$$

where $S = \{(i,j) : i < j \text{ and } u_i = u_j\}$ and $T = \{(i,j) : i < j \text{ and } u_i \neq u_j\}$.

Let ϕ be a formula of $(\pm\Omega)^*[L^\omega_{\infty\omega}]$, where Ω is a set of Lindström quantifiers. Then ϕ is a *basic flat formula* if it is atomic or if it is formed by applying a quantifier $Q \in \Omega$ to some first-order quantifier-free formulae. Also, ϕ is in *flat normal form* if it is obtained from basic flat formulae by successive applications of the connectives $\vee$, $\wedge$ and $\neg$ and the first-order quantifiers $\forall$ and $\exists$. For example, the formula $\mathrm{HP}[\lambda\mathbf{xy}\phi](\mathbf{0}, \mathbf{max})$ of Theorem 2 is a basic flat formula, whereas the formula $\exists z_1 \exists z_2 \ldots \exists z_k (\mathrm{HP}[\lambda\mathbf{xy}\phi](\mathbf{0}, \mathbf{max}) \wedge \neg \mathrm{HP}[\lambda\mathbf{x'y'}\phi'](\mathbf{0}, \mathbf{max}))$ of Theorem 2 is in flat normal form.

We are now in a position to state Dawar and Hella's result (which although stated in [7] is proved in [8]).

Theorem 5. [7, 8] *Let σ_ϵ be the empty signature, and denote the unique structure of size n over σ_ϵ by $\mathcal{A}_n$. Let Ω_0 be a finite set of Lindström quantifiers.*

(*a*) *For every n, there exists a sentence $\eta_n \in (\pm\Omega_0)^*[L^k]$ such that for every n', $\mathcal{A}_{n'} \models \eta_n$ if, and only if, $\mathcal{A}_n$ and $\mathcal{A}_{n'}$ satisfy the same sentences of $(\pm\Omega_0)^*[L^k_{\infty\omega}]$.*

(*b*) *The sentence* η_n *is of the form*

$$\bigwedge_{1\le i\le m} \exists x_1 \ldots \exists x_k \psi_i \wedge \forall x_1 \ldots \forall x_k \bigvee_{1\le i\le m} \psi_i \wedge \bigwedge_{1\le j\le r} \forall x_1 \ldots \forall x_k (\phi_j \leftrightarrow \gamma_j),$$

where $\psi_1, \psi_2, \ldots, \psi_m$ *are the basic equality types of* k*-tuples of* $\mathcal{A}_n$*, each formula* γ_j *is a disjunction of some basic equality types and each formula* ϕ_j *is a basic flat formula.*

(*c*) *Given some* $\mathbf{u} \in |\mathcal{A}_n|^l$*, there is a formula* $\eta_{n,\mathbf{u}}$ *of the form*

$$\eta_n \wedge \forall x_{l+1} \ldots \forall x_k \gamma,$$

where γ *is a disjunction of some basic equality types, such that for every* $\mathcal{A}_{n'}$ *and* $\mathbf{u}' \in |\mathcal{A}_{n'}|^l$*,*

$$\mathcal{A}_{n'} \models \eta_{n,\mathbf{u}}(\mathbf{u}') \text{ if, and only if, the expansions } \langle \mathcal{A}_n, \mathbf{u} \rangle \text{ and } \langle \mathcal{A}_{n'}, \mathbf{u}' \rangle \text{ satisfy exactly the same sentences of } (\pm\Omega_0)^*[L^k_{\infty\omega}].$$

□

Corollary 2. [7] *Let* Ω_0 *be a finite set of Lindström quantifiers and let* σ_ϵ *be the empty signature. For every formula* $\phi(x_1, x_2, \ldots, x_l) \in (\pm\Omega_0)^*[L^\omega_{\infty\omega}(\sigma_\epsilon)]$*, there exists a formula* $\psi(x_1, x_2, \ldots, x_l) \in (\pm\Omega_0)^1[FO(\sigma_\epsilon)]$ *such that* ϕ *and* ψ *are equivalent.*

Proof. There is clearly some k such that $\phi \in (\pm\Omega_0)^*[L^k_{\infty\omega}(\sigma_\epsilon)]$. Fix n and let $\psi_1, \psi_2, \ldots, \psi_m$ be the basic equality types of k-tuples of elements of $\mathcal{A}_n$. Let η_n be the sentence as in the statement of Theorem 5, and for every $\mathbf{u} \in |\mathcal{A}_n|^l$, let $\eta_{n,\mathbf{u}}$ be as in Theorem 5. Define $F = \{(n, \mathbf{u}) : \mathbf{u} \in |\mathcal{A}_n|^l \text{ such that } \mathcal{A}_n \models \phi(\mathbf{u})\}$ and define ψ as

$$\bigvee_{(n,\mathbf{u})\in F} \eta_{n,\mathbf{u}}.$$

Consider any n and $\mathbf{u} \in |\mathcal{A}_n|^l$. Suppose that $\mathcal{A}_n \models \phi(\mathbf{u})$. So, $(n, \mathbf{u}) \in F$, $\eta_{n,\mathbf{u}}$ is a disjunct of ψ and $\mathcal{A}_n \models \eta_{n,\mathbf{u}}(\mathbf{u})$. Thus, $\mathcal{A}_n \models \psi(\mathbf{u})$. Conversely, suppose that $\mathcal{A}_n \models \psi(\mathbf{u})$. So, $\mathcal{A}_n \models \eta_{n',\mathbf{u}'}(\mathbf{u})$, for some $(n', \mathbf{u}') \in F$, with the result that $\langle \mathcal{A}_n, \mathbf{u} \rangle$ and $\langle \mathcal{A}_{n'}, \mathbf{u}' \rangle$ satisfy the same sentences of $(\pm\Omega_0)^*[L^k_{\infty\omega}]$. In particular, $\mathcal{A}_n \models \phi(\mathbf{u})$. The result follows. □

The following is immediate from Corollary 2.

Corollary 3. [7] *If* Ω *is a set of Lindström quantifiers such that every quantifier can be evaluated in* **NP** *then every sentence of* $(\pm\Omega)^*[FO(\sigma_\epsilon)]$ *is equivalent to a Boolean combination of* **NP** *properties.* □

Given a language A over $\{0, 1\}$,

$$tally(A) = \{1^n : \text{ the binary representation of } n \in 1A\}.$$

The Linear Exponential Boolean Hierarchy, **EBH**, is defined as follows.

$\mathbf{EBH}(0) = \mathbf{DETIME} = \mathbf{DTIME}(2^{O(n)})$;
$\mathbf{EBH}(1) = \mathbf{NETIME} = \mathbf{NTIME}(2^{O(n)})$;
$\mathbf{EBH}(2i) = \{X \cap Y : X \in \mathbf{EBH}(2i-1), Y \in \mathbf{co\text{-}NETIME}\}$;
$\mathbf{EBH}(2i+1) = \{X \cup Y : X \in \mathbf{EBH}(2i), Y \in \mathbf{NETIME}\}$,

for $i \geq 0$, and $\mathbf{EBH} = \cup\{\mathbf{EBH}(i) : i = 0, 1, \ldots\}$. The Full Exponential Boolean Hierarchy, **EXPBH**, is defined as follows.

$\mathbf{EXPBH}(0) = \mathbf{DEXPTIME} = \mathbf{DTIME}(2^{O(poly(n))})$;
$\mathbf{EXPBH}(1) = \mathbf{NEXPTIME} = \mathbf{NTIME}(2^{O(poly(n))})$;
$\mathbf{EXPBH}(2i) = \{X \cap Y : X \in \mathbf{EXPBH}(2i-1), Y \in \mathbf{co\text{-}NEXPTIME}\}$;
$\mathbf{EXPBH}(2i+1) = \{X \cup Y : X \in \mathbf{EXPBH}(2i), Y \in \mathbf{NEXPTIME}\}$,

for $i \geq 0$, and $\mathbf{EXPBH} = \cup\{\mathbf{EXPBH}(i) : i = 0, 1, \ldots\}$.

The hierarchies **EBH** and **EXPBH** can be regarded as exponential versions of the Boolean Hierarchy. Indeed, we can go further and associate with many complexity classes **CC** contained in the Polynomial Hierarchy, a *linear exponential version* $E(\mathbf{CC})$ and a *full exponential version* $EXP(\mathbf{CC})$. Such an association is detailed in Fig. 1 below, with the motivation behind such an association given in the subsequent lemma. The definitions of the complexity classes have either been given or are obvious, except that $\{\Sigma_i^e, \Pi_i^e : i = 0, 1, \ldots\}$ (resp. $\{\Sigma_i^{exp}\Pi_i^{exp} : i = 0, 1, \ldots\}$) are the classes of the *Linear* (resp. *Full*) *Exponential Hierarchy* **EH** (resp. **EXPH**), built around **ETIME** and **NETIME** (resp. **EXPTIME** and **NEXPTIME**) just as the Polynomial Hierarchy is built around **P** and **NP**.

Basic complexity class **CC**	Linear exponential version $E(\mathbf{CC})$	Full exponential version $EXP(\mathbf{CC})$
P	**ETIME**	**EXPTIME**
NP	**NETIME**	**NEXPTIME**
Σ_i^p	Σ_i^e	Σ_i^{exp}
Π_i^p	Π_i^e	Π_i^{exp}
PH	**EH**	**EXPH**
$\mathbf{BH}(i)$	$\mathbf{EBH}(i)$	$\mathbf{EXPBH}(i)$
BH	**EBH**	**EXPBH**
L	**LINSPACE**	**PSPACE**
NL	**NLINSPACE**	**PSPACE**
$\mathbf{L}^{\mathbf{NP}}$	$\mathbf{LINSPACE}^{\mathbf{NP}}$	$\mathbf{PSPACE}^{\mathbf{NP}}$

Figure 1. Linear and full exponential versions of complexity classes.

Lemma 3. [16] *For each basic complexity class* **CC** *in the first column of Figure* 1:

- *the closure under polynomial-time many-one reductions of* $E(\mathbf{CC})$ *is the complexity class* $EXP(\mathbf{CC})$; *and*
- *for every language* $\Lambda \in \mathbf{CC}$, $tally(\Lambda) \in \mathbf{CC}$ *if, and only if,* $\Lambda \in E(\mathbf{CC})$.

□

We are now ready to establish the results of Dawar, Gottlob and Hella.

Theorem 6. [7] *If there exists a family Ω of Lindström quantifiers where each quantifier in Ω can be evaluated in* **NP** *and where* $(\pm\Omega)^*[FO] = \mathbf{L}^{\mathbf{NP}}$ *then*:

- $\mathbf{EBH} = \mathbf{LINSPACE}^{\mathbf{NP}}$ *and* **EBH** *collapses*; *and*
- $\mathbf{EXPBH} = \mathbf{PSPACE}^{\mathbf{NP}}$ *and* **EXPBH** *collapses.*

Proof. Let Λ be a language in $\mathbf{LINSPACE}^{\mathbf{NP}}$. By Lemma 3, $tally(\Lambda) \in \mathbf{L}^{\mathbf{NP}}$. Define the problem Γ over σ_ϵ to be such that $tally(\Lambda)$ is the natural encoding of Γ. By hypothesis, $\Gamma \in (\pm\Omega)^*[\mathrm{FO}] = \mathbf{L}^{\mathbf{NP}}$; and so by Corollary 3,

$$tally(\Lambda) \in \mathbf{EBH}_k, \text{ for some } k.$$

But $\mathbf{LINSPACE}^{\mathbf{NP}}$ has complete problems, and consequently $\mathbf{LINSPACE}^{\mathbf{NP}} \subseteq \mathbf{EBH}_k$. As $\mathbf{EBH} \subseteq \mathbf{LINSPACE}^{\mathbf{NP}}$, the first part of the theorem follows.

However, the second part of the theorem also follows as, by Lemma 3, the complexity classes $\mathbf{EXPBH}_k$, **EXPBH** and $\mathbf{PSPACE}^{\mathbf{NP}}$ are the closures of the complexity classes $\mathbf{EBH}_k$, **EBH** and $\mathbf{LINSPACE}^{\mathbf{NP}}$, respectively. □

Any collapses of the form detailed in Theorem 6 would cause not inconsiderable surprise amongst many complexity theorists. Therefore, we are led to speculate that the complexity class $\mathbf{L}^{\mathbf{NP}}$ can not be captured on the class of all structures by a "Lindström logic". It would be extremely interesting if this result could be established. However, a great improvement of Theorem 6 would be if the premise in the theorem yielded a complexity-theoretic collapse "lower down", say of the Polynomial Hierarchy.

We close by noting that things are different for some complexity classes "beyond" $\mathbf{P}^{\mathbf{NP}}$. Amalgamating results due to Gurevich [17] and Blass and Gurevich [2], Dawar [6] observed that any recursively presented complexity class containing $\mathbf{P}^{\mathbf{NP}}$ and closed under compositions is recursively indexable, and so there is a logic capturing this class. Other results in [6] then yield that if the complexity class is *bounded* (see [6]), then it can be captured by a "Lindström logic". So, for example, $\mathbf{L}^{\Sigma_2^p}$ can be captured by a "Lindström logic" (as it is bounded).

Acknowledgements. I am indebted to Argimiro Arratia-Quesada, Anuj Dawar, Richard Gault and Juha Nurmonen for their comments on a first draft of this paper.

References

1. E. Allender, J. Balcázar and N. Immerman, A first-order isomorphism theorem, *Lecture Notes in Computer Science Vol.* 665 (1993) 163–174.
2. A. Blass and Y. Gurevich, Equivalence relations, invariants and normal forms, *SIAM J. Comput.* **13** (1984) 682–689.
3. J. Barwise, On Moschovakis closure ordinals, *J. Symbolic Logic* **42** (1997) 292–296.
4. J.F. Buss, Alternations and space-bounded computations, *J. Comput. Systems Sci.* **36** (1988) 351–378.

5. E. Dahlhaus, Reduction to NP-complete problems by interpretations, *Lecture Notes in Computer Science Vol.* 171, Springer-Verlag (1984) 357–365.
6. A. Dawar, Generalized quantifiers and logical reducibilities, *J. Logic Computat.* **5** (1995) 213–226.
7. A. Dawar, G. Gottlob and L. Hella, Capturing relativized complexity classes without order, *Mathematical Logic Quarterly*, to appear.
8. A. Dawar and L. Hella, The expressive power of finitely many generalized quantifiers, *Inform. and Computat.* **123** (1995) 172–184.
9. H.D. Ebbinghaus and J. Flum, *Finite Model Theory*, Springer-Verlag (1995).
10. R. Fagin, Generalized first-order spectra and polynomial-time recognizable sets, in: *Complexity of Computation* (ed. R.M. Karp), *SIAM-AMS Proceedings* **7** (1974) 43–73.
11. M.R. Garey and D.S. Johnson, *Computers and Intractability: a Guide to the Theory of NP-Completeness*, W.H. Freeman and Co. (1979).
12. Y.V. Glebskii, D.I. Kogan, M.I. Liogonki and V.A. Talanov, Range and degree of realizability of formulas in the restricted predicate calculus, *Cybernetics* **5** (1969) 142–154.
13. E. Grädel, On logical descriptions of some concepts in structural complexity theory, *Lecture Notes in Computer Science Vol.* 440 (1990) 163–175.
14. E. Grädel, Capturing complexity classes by fragments of second-order logic, *Theoret. Comput. Sci.* **101** (1992) 35–57.
15. G. Gottlob, Relativized logspace and generalized quantifiers over finite ordered structures, *J. Symbolic Logic* **62** (1997) 545–574.
16. G. Gottlob, N. Leone and H. Veith, Second order logic and the weak exponential hierarchies, *Lecture Notes in Computer Science Vol.* 969, Springer-Verlag (1995) 66–81.
17. Y. Gurevich, Toward logic tailored for computational complexity, *Lecture Notes in Mathematics Vol.* 1104, Springer-Verlag (1984) 175–216.
18. Y. Gurevich, Logic and the challenge of computer science, in: *Trends in Theoretical Computer Science* (ed. E. Börger), Computer Science Press (1988) 1–57.
19. J. Hartmanis, N. Immerman and S. Mahaney, One-way log-tape reductions, *Proc. 19th Symp. Foundations of Computer Science*, IEEE Press (1978) 65–71.
20. N. Immerman, Relational queries computable in polynomial time, *Inform. Control* **68** (1986) 86–104.
21. N. Immerman, Languages that capture complexity classes, *SIAM J. Comput.* **16** (1987) 760–778.
22. N. Immerman, Nondeterministic space is closed under complementation, *SIAM J. Comput.* **17** (1988) 935–938.
23. J. Köbler, U. Schöning and K.W. Wagner, The difference and the truth-table hierarchies for NP, *RAIRO Inform. Theory* **21** (1987) 419–435.
24. P. Lindström, First order predicate logic with generalized quantifiers, *Theoria* **32** (1966) 186–195.
25. P. Lindström, On extensions of elementary logic, *Theoria* **35** (1969) 1–11.
26. J.A. Makowsky and Y.B. Pnueli, Oracles and quantifiers, *Lecture Notes in Computer Science Vol.* 832 (1994) 189–222.
27. J.A. Makowsky and Y. Pnueli, Logics capturing oracle complexity classes uniformly, *Lecture Notes in Computer Science Vol.* 960 (1995) 463–479.
28. J.A. Makowsky and Y. Pnueli, Computable quantifiers and logics over finite structures, in: *Quantifiers: Logics, Models and Computation, Volume* 1 (ed. M. Krynicki et al.), Kluwer (1995) 313–357.

29. W. Savitch, Maze recognizing automata and nondeterministic tape complexity, *J. Comput. System Sci.* **7** (1973) 389–403.
30. I.A. Stewart, Comparing the expressibility of languages formed using NP-complete operators, *J. Logic Computat.* **1** (1991) 305–330.
31. I.A. Stewart, Using the Hamiltonian path operator to capture NP, *J. Comput. System Sci.* **45** (1992) 127–151.
32. I.A. Stewart, Logical characterizations of bounded query classes I: logspace oracle machines, *Fund. Informat.* **18** (1993) 65–92.
33. I.A. Stewart, Logical characterizations of bounded query classes II: polynomial-time oracle machines, *Fund. Informat.* **18** (1993) 93–105.
34. I.A. Stewart, Methods for proving completeness via logical reductions, *Theoret. Comput. Sci.* **118** (1993) 193–229.
35. I.A. Stewart, Logical description of monotone NP problems, *J. Logic Computat.* **4** (1994) 337–357.
36. L. Stockmeyer, The polynomial hierarchy, *Theoret. Comput. Sci.* **3** (1976) 1–22.
37. R. Szelepcsényi, The method of forced enumeration for nondeterministic automata, *Acta Informat.* **26** (1988) 279–284.
38. B.A. Trakhtenbrot, Impossibility of an algorithm for the decision problem on finite classes, *Doklady* **70** (1950) 569–572.
39. M. Vardi, Complexity of relational query languages, *Proc. 14th ACM Ann. Symp. on Theory of Computing* (1982) 137–146.
40. K.W. Wagner, Bounded query classes, *SIAM J. Comput.* **19** (1990) 833–846.