Refining Logical Characterizations of Advice Complexity Classes [1]

Albert Atserias José L. Balcázar

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Jordi Girona Salgado 1-3
08034 Barcelona, Spain
{atserias, balqui}@lsi.upc.es

Abstract: Numerical relations in logics are known to characterize, via the finite models of their sentences, polynomial advice nonuniform complexity classes. These are known to coincide with reduction classes of tally sets. Our contributions here are: 1/ a refinement of that characterization that individualizes the reduction class of each tally set, and 2/ characterizing logarithmic advice classes via numerical constants, both in the (rather easy) case of C/log and in the more complex case of Full-C/log; this proof requires to extend to classes below P the technical characterizations known for the class Full-P/log.

1. Introduction

Many complexity classes are captured by appropriate logics. Here we focus on nonuniform complexity classes [KL80] originally introduced for dealing with computational models in which a different device is used for each input length; boolean circuits are an example of such a model. Nonuniform complexity classes have many characterizations in various, sometimes surprising terms, and thus become of interest in several approaches to computational complexity. See [BDG95] and the references there. (They are defined in the preliminaries below.)

In particular, [M90] proves that classes with polynomially long advice words are captured by the same logics that capture the corresponding uniform class, when they are extended by arbitrary numerical relations. Given that specific numerical relations (like $\mathrm{BIT}(x, y)$ whose interpretation is true if the x-th bit of y is 1) are frequently assumed available, it is a natural question to ask for a characterization of logics with arbitrary nu-

[1] This research is partially supported by the UE through the Esprit Long Term Research program, project 20244 (ALCOM-IT) and through the HCM network CHRX–CT93–0415 (COLORET), and by the Spanish MEC, Acción Integrada HA-201-B and DGICYT, project PB95-0787 (KOALA).

merical relations, and Molzan answers this using polynomial advice. Some cases are also described in [S94], [GL84], [I87], [HMPST93] using classes defined by nonuniform circuits.

The classes corresponding to logarithmic advice have been studied to a much lesser extent. They come in two flavors: in the most general case, each advice word can be used only at a specific length, whereas in the most restrictive case (termed "strong" in [K87], where they were introduced, and "Full" in more recent works) each advice word is also good for all smaller lengths. Complexity-theoretic studies of these classes can be found in [BS92], [HM94], [BH97], and [BBH97], and a more global perspective is given in Hermo's PhD dissertation [H96].

For example, as polynomial size circuits correspond to P/poly (polynomial time with polynomial advice), likewise circuits with easy (Turing) descriptions correspond to P/log, and if these circuits can be combined via regular expression operators then Full-P/log is obtained (see [HM94], [BBH97]).

The expressive power of logics with numerical constants seems not to have been studied, although it is an easy exercise: each constant provides $\log n$ bits of information in a finite universe of size n. Thus, advice words of length $O(\log n)$ have to be added to exactly match the power of a logic with additional numerical constants. We fully formalize this observation below.

To find logics capturing Full advice classes we end up introducing a model-theoretic variant of the constant symbols: moduled constants, that hold a tight relation between the interpretation of the symbol at each universe size. We extend a previously known technical characterization of Full advice classes to prove that logics with moduled constants capture Full logarithmic advice classes.

The notion can be lifted actually to relations: we introduce moduled relations, and prove that they give the exact classification of polynomial advice classes, depending on their characterizations via tally sets. More precisely: the reduction class of all tally sets under a reducibility defined by a relativized complexity class (e.g. polynomial time: $\cup_{T \text{ tally}} P(T)$) is the same as the class plus polynomial advice (e.g. P/poly here). Then not only arbitrary numerical relations over, say, fixpoint logic characterize $\cup_{T \text{ tally}} P(T)$, but also each moduled relation gives exactly the reduction class of an individual tally set. Other logics can be used as well, to get closure under a reducibility depending on the logic.

2. Preliminaries in computational complexity

We only consider words and languages over the alphabet $\Sigma = \{0, 1\}$. By Σ^n and $\Sigma^{\leq n}$ we denote the words of length exactly n or at most n, respectively. We define a standard pairing function as follows: given a pair of words x_1 and x_2, the pair $\langle x_1, x_2 \rangle$ is defined from x_1 and x_2 by doubling each symbol of x_1 and appending the word $01x_2$; that is, if $x_1 = a_1 ... a_n$ then $\langle x_1, x_2 \rangle = a_1^2 ... a_n^2 01 x_2$. Similarly, given a finite sequence of words $x_1, ..., x_n$, $n \geq 2$, we define $\langle x_1, ..., x_n \rangle = \langle x_1, \langle x_2, ..., x_n \rangle \rangle$. For every word x, we set $\langle x \rangle = x$. A tally set is a language over $\{0\}$. The class of tally languages is denoted TALLY.

Our model of computation will be the standard multitape oracle Turing machine with input alphabet Σ. The class of languages accepted by Turing machines with oracle A in nondeterministic polynomial time is denoted NP(A). Similarly, P(A), NL(A) and L(A) denote the class of languages accepted by Turing machines with oracle A in deterministic polynomial time, nondeterministic logarithmic space, and deterministic logarithmic space respectively. For $\mathcal{C}$ in $\{\mathrm{NP}, \mathrm{P}, \mathrm{NL}, \mathrm{L}\}$, $\mathcal{C}(\emptyset)$ is abreviated $\mathcal{C}$. By a nice complexity class we mean a reasonable complexity class (such as NP, P, NL, L, P/poly, P/log,... for some examples). Moreover, a nice complexity class has to be closed under first-order projection reductions [IL95].

Let $\mathcal{F}$ be a class of advice functions (functions from $\mathbb{N}$ to Σ^*) and let $\mathcal{C}$ be a class of languages. The class $\mathcal{C}/\mathcal{F}$ is defined as the class of languages L for which there is an advice $f \in \mathcal{F}$ and a language $L' \in \mathcal{C}$ such that, for all n and $x \in \Sigma^n$, $x \in L$ if and only if $\langle x, f(n) \rangle \in L'$. Similarly, Ko introduced full advice classes as follows: Full-$\mathcal{C}/\mathcal{F}$ is the class of languages L for which there is an advice $f \in \mathcal{F}$ and a language $L' \in \mathcal{C}$ such that, for all n and $x \in \Sigma^{\leq n}$, $x \in L$ if and only if $\langle x, f(n) \rangle \in L'$. An advice function f is called polynomial if there exists a polynomial p such that, for each n, $|f(n)| \leq p(n)$. Similarly, f is called logarithmic if there exists a real constant $c > 0$ such that, for each n, $|f(n)| \leq c \log n$. We denote by poly and log the class of polynomial and logarithmic advice functions respectively.

3. Preliminaries in descriptive complexity

Following [S94], a numerical relation of arity r is a sequence of r-ary relations $R = (R_1, R_2, ...)$ where, for all $n > 0$, $R_n \subseteq \{0, ..., n-1\}^r$. Now, given a logic $\mathcal{L}$ we define $\mathcal{L}[R]$ as an extension of $\mathcal{L}$ by a new logical relation symbol R of arity r. The interpretation of R may change for each structure; in fact for a finite structure with universe $\{0, ..., n-1\}$, the logical symbol R is interpreted as R_n (recall that $R_n \subseteq \{0, ..., n-1\}^r$). For example, if $\leq$ denotes the numerical relation which is interpreted as the natural ordering on $\{0, ..., n-1\}$ then $\mathcal{L}[\leq]$ is the extension of $\mathcal{L}$ by an ordering on the universe. In a similar fashion, we define a numerical constant as a sequence of natural numbers $c = (c_1, c_2, ...)$ where, for all $n > 0$, $c_n \in \{0, ..., n-1\}$. As before, we define $\mathcal{L}[c]$ as the extension of $\mathcal{L}$ by a new logical constant symbol c which is to be interpreted as c_n for structures with universe $\{0, ..., n-1\}$.

Given sequences of numerical relations $R_1, ..., R_n$ and numerical constants $c_1, ..., c_n$, $\mathcal{L}[R_1, ..., R_n]$ stands for $\mathcal{L}[R_1, ..., R_{n-1}][R_n]$ and $\mathcal{L}[c_1, ..., c_n]$ for $\mathcal{L}[c_1, ..., c_{n-1}][c_n]$. Finally, $\mathcal{L}[\mathbf{R}]$ and $\mathcal{L}[\mathbf{c}]$ denote the union of the logics $\mathcal{L}[R]$ and $\mathcal{L}[c]$ over every numerical relation and constant respectively; that is, $\mathcal{L}[\mathbf{R}]$ ($\mathcal{L}[\mathbf{c}]$) is the extension of $\mathcal{L}$ by arbitrarily many new logical relation (constant) symbols for numerical relations (constants).

We recall next how finite logical structures are interpreted as words and conversely.

Consider a signature $\tau = \{R_1, ..., R_r, c_1, ..., c_s\}$ where $R_1, ..., R_r$ are relational symbols and $c_1, ..., c_s$ are constant symbols (as usual in descriptive complexity our signatures do not contain functional symbols for convenience). To each finite τ-structure $M = (\{0, ..., n-1\}, R_1^M, ..., R_r^M, c_1^M, ..., c_s^M)$ we associate the word

$$w(M) = \langle w(R_1^M), ..., w(R_r^M), w(c_1^M), ..., w(c_s^M)\rangle$$

where $w(R_i^M)$ is the characteristic sequence of the relation R_i^M and $w(c_i^M)$ is the binary representation of c_i^M (with leading zeros up to length $\log n$). Conversely, to each word x we associate the τ-structure

$$M_\tau(x) = (\{0, ..., n-1\}, R_1, ..., R_r, c_1, ..., c_s)$$

such that $w(M_\tau(x)) = x$. Note that the claimed interpretation of x always exists for the signature $\tau = \{P\}$ with a unique unary relation symbol P (because $\langle x\rangle = x$) but could be not defined for any other signature. For a language L and a signature τ, $M_\tau(L)$ denotes the set $\{M_\tau(x) : x \in L\}$ if defined, and for a class of finite structures K, $w(K)$ denotes the set $\{w(M) : M \in K\}$.

Let $\mathcal{L}$ be a logic. We denote by $\models_\mathcal{L}$ the satisfaction relation of $\mathcal{L}$. Similarly, given a sentence $\varphi \in \mathcal{L}$, $\mathrm{Mod}_\mathcal{L}(\varphi)$ denotes the class of finite structures M of universe $\{0, ..., n-1\}$ for some $n \in \mathbb{N}$ such that $M \models_\mathcal{L} \varphi$. Whenever the logic is clear from the context we shall omit the subscripts in $\models_\mathcal{L}$ and $\mathrm{Mod}_\mathcal{L}$. We say that the logic $\mathcal{L}$ subsumes a class of languages $\mathcal{C}$ whenever for every language $L \in \mathcal{C}$ there is a sentence $\varphi \in \mathcal{L}$ such that $w(\mathrm{Mod}(\varphi)) = L$. Similarly, the class $\mathcal{C}$ subsumes the logic $\mathcal{L}$ if for every sentence $\varphi \in \mathcal{L}$, $w(\mathrm{Mod}(\varphi)) \in \mathcal{C}$. We say that a logic captures a class if they subsume each other. Many usual complexity classes are logically characterized in this sense. Namely, Fagin ([F74]) proved that the existential fragment of second-order logic captures NP; that is, Σ_1^1 captures NP. Later, independently, Immerman [I87] and Vardi [Vardi82] proved that P was captured by the extension of first-order logic by a least-fixed point operator and an ordering on the universe; that is FO(LFP)[$\leq$] captures P. Similarly, Immerman extends first-order logic by transitive closure and deterministic transitive closure to capture NL and L respectively; that is FO(TC)[$\leq$] captures NL and FO(dTC)[$\leq$] captures L.

We assume that all our logics can express the linear ordering and the BIT predicate, as in [BIS90], either due to their own expressive power or by having them built-in as default numerical relations. We also assume closure under standard existential quantification and the usual connectives.

4. Logical characterizations of logarithmic advice classes

Logical characterizations of some polynomial advice classes are presented by Molzan in [M90]. Namely, P/poly, L/poly and NL/poly are captured by FO(LFP)[**R**], FO(TC)[**R**] and FO(dTC)[**R**] respectively. These results are inmediately generalizable to give a logical characterization of $\mathcal{C}$/poly provided a logical characterization of $\mathcal{C}$ is known. The purpose of the present section is to give a logical characterization of logarithmic advice classes.

Roughly speaking, Molzan's characterizations are based in the following idea: a polynomial advice can be coded by a (nonuniform) sequence of relations, one for each length. Hence, a natural idea to attempt the characterization of nonuniform classes with logarithmic advice could be the following: a logarithmic advice can be coded by a (nonuniform) sequence of constants, one for each length. In the following, let $\mathcal{C}$ be a nice complexity class and let $\mathcal{L}$ be a logic.

Lemma 4.1: *If $\mathcal{C}$ subsumes $\mathcal{L}$ then $\mathcal{C}/\log$ subsumes $\mathcal{L}[\mathbf{c}]$.*

Proof: Suppose that $\mathcal{C}$ subsumes $\mathcal{L}$ and let φ be a sentence in $\mathcal{L}[\overline{c}]$, $\overline{c} = c_1, ..., c_k$. Let $f : \mathbb{N} \to \Sigma^*$ be the function defined as follows: for each n, $f(n)$ is the k-tuple $\langle w(c_{1n}), ..., w(c_{kn})\rangle$. This function will be used as an advice to decide $L = w(\mathrm{Mod}(\varphi))$. Let σ be the signature of φ and build ψ from φ by replacing each ocurrence of the numerical constant c_i in φ by a new constant symbol $d_i \notin \sigma$. Now ψ is a sentence of $\mathcal{L}$ of signature $\tau = \sigma \cup \{d_1, ..., d_k\}$. Let L' be $w(\mathrm{Mod}(\psi))$. It holds that, given a word x of length n,

$$x \in L \Leftrightarrow M_\sigma(x) \models_{\mathcal{L}[\overline{c}]} \varphi \Leftrightarrow (M_\sigma(x), c_{1n}, ..., c_{kn}) \models_{\mathcal{L}} \psi \Leftrightarrow \langle x, f(|x|)\rangle \in L'$$

Finally, since $\mathcal{C}$ subsumes $\mathcal{L}$ and $\psi \in \mathcal{L}$, $L' \in \mathcal{C}$; that is, $L \in \mathcal{C}/\log$ via the advice function f and the language L'. □

Lemma 4.2: *If $\mathcal{L}$ subsumes $\mathcal{C}$ then $\mathcal{L}[\mathbf{c}]$ subsumes $\mathcal{C}/\log$.*

Proof: Suppose that $\mathcal{L}$ subsumes $\mathcal{C}$ and let L be a language in $\mathcal{C}/\log$ via a logarithmic advice function $f : \mathbb{N} \to \Sigma^*$ and a language $L' \in \mathcal{C}$. Since $\mathcal{C}$ is a nice complexity class, we can assume without loss of generality that $|f(n)| = k \log n$, $k \in \mathbb{N}$. Define $c_i = (c_{i1}, c_{i2}, ...)$, $i = 1, ..., k$, as numerical constants where $\sum_{i=1}^{k} c_{in} n^{k-i}$ is the number represented in binary by $f(n)$. We shall prove that there is a sentence $\varphi \in \mathcal{L}[c_1, ..., c_k]$ such that $w(\mathrm{Mod}(\varphi)) = L$.

Since $\mathcal{L}$ subsumes $\mathcal{C}$ and $L' \in \mathcal{C}$, there exists a sentence $\psi \in \mathcal{L}$ such that $w(\mathrm{Mod}(\psi)) = L'$. Moreover, every word in L' is a pair $\langle x, y\rangle$ where $x \in \Sigma^*$ and $|y| = k \log |x|$. Since $\mathcal{C}$ is nice, it is closed under first-order projection reductions. Hence, we can assume, without loss of generality, that every word in L' is in fact a $(k+1)$-tuple $\langle x, y_1, ..., y_k\rangle$ where $x \in \Sigma^*$ and $|y_i| = \log |x|$, $i = 1, ..., k$. As a result, ψ is a sentence over a signature $\tau = \{P, d_1, ..., d_k\}$ where P is a unary relation symbol and $d_1, ..., d_k$ are constant symbols. Now, build φ from ψ by replacing, for $i = 1, ..., k$, each ocurrence of d_i by the numerical constant c_i. We obtain an $\mathcal{L}[c_1, ..., c_k]$-sentence such that, for every structure M of signature $\sigma = \tau - \{d_1, ..., d_k\}$ and universe $\{0, ..., n-1\}$,

$$M \models_{\mathcal{L}[\overline{c}]} \varphi \Leftrightarrow (M, c_{1n}, ..., c_{kn}) \models_{\mathcal{L}} \psi \Leftrightarrow \langle w(M), w(c_{1n}), ..., w(c_{kn})\rangle \in L' \Leftrightarrow w(M) \in L$$

Now, if $x \in L$ then $x = w(M)$ for some structure M of signature σ. Hence, $M \models_{\mathcal{L}[\overline{c}]} \varphi$ and $x \in w(\mathrm{Mod}(\varphi))$. Conversely, if $x \in w(\mathrm{Mod}(\varphi))$ then $M_\sigma(x) \models_{\mathcal{L}[\overline{c}]} \varphi$ and $x = w(M_\sigma(x)) \in L$. That is $L = w(\mathrm{Mod}(\varphi))$ as was to be proved. □

The logical characterization of $\mathcal{C}/\log$ is now quite obvious.

Theorem 4.3: *Let $\mathcal{C}$ be a nice complexity class captured by a logic $\mathcal{L}$. Then, $\mathcal{L}[\mathbf{c}]$ captures $\mathcal{C}/\log$.*

With this theorem in mind many usual logarithmic advice classes are captured. For some examples, NP/log, P/log, NL/log and L/log are captured by $\Sigma_1^1[\mathbf{c}]$, FO(LFP)[$\mathbf{c}$], FO(TC)[$\mathbf{c}$] and FO(dTC)[$\mathbf{c}$] respectively.

5. Logical characterizations of Full logarithmic advice classes

We turn now to consider full logarithmic advice classes. In [H96] it was proved that every language in Full-P/log has a prefixed advice function; that is, an advice function $f : \mathbb{N} \to \Sigma^*$ such that, for each n, $f(n)$ is a prefix of $f(n+1)$. If we denote by Pref-$\mathcal{C}/\mathcal{F}$ the class of languages that belong to Full-$\mathcal{C}/\mathcal{F}$ via a prefixed advice function then the remark above can be written Pref-P/log=Full-P/log. Note that for nice complexity classes it is always the case that Pref-$\mathcal{C}/\mathcal{F} \subseteq$ Full-$\mathcal{C}/\mathcal{F}$. Our claim is that Full-$\mathcal{C}$/log is captured by the extension of $\mathcal{L}$ with a restricted class of numerical constants; specifically, those suggested by the "prefix" property, that, in term of constants, can be defined in terms of congruence modulo powers of 2.

Formally, for a natural number $n \geq 1$, $\overline{n}$ denotes the nearest power of 2 greater than n ($\overline{n} = \min\{2^k > n : k \geq 0\}$). A numerical constant $c = (c_1, c_2, ...)$ is said to be moduled if, for each $n > 1$, $c_n \equiv c_{n-1} \pmod{\overline{c_{n-1}}}$. Given a logic $\mathcal{L}$, denote by $\mathcal{L}[\mathbf{c}_\equiv]$ the restriction of $\mathcal{L}[\mathbf{c}]$ to moduled numerical constants.

Lemma 5.1: *Let c be a moduled numerical constant. If $\mathcal{C}$ subsumes $\mathcal{L}$ then* Full-$\mathcal{C}/\log$ *subsumes $\mathcal{L}[c]$.*

Proof: Suppose that $\mathcal{C}$ subsumes $\mathcal{L}$ and let φ be a sentence in $\mathcal{L}[c]$. Let $f : \mathbb{N} \to \Sigma^*$ be the function defined as $f(n) = w(c_n)$, for each n. The same proof as in Lemma 4.1 for $k = 1$ works to prove that $w(\mathrm{Mod}(\varphi)) \in \mathcal{C}/\log$ via f. Next define g from f as follows: for each n, $g(n)$ is the reverse of $f(n)$ without leading zeros; that is, if $f(n) = 0^k 1x$ then $g(n) = x^R 1$ and if $f(n) = 0^k$ then $g(n) = \lambda$. Clearly, g is also an advice witnessing that $w(\mathrm{Mod}(\varphi)) \in \mathcal{C}/\log$ because the space constructibility of $\log n$ makes possible to add leading zeros whenever necessary and the closure of $\mathcal{C}$ under first-order projections provides the reversing. As c is a moduled numerical constant, for each n, $g(n)$ is a prefix of $g(n+1)$. It follows that $w(\mathrm{Mod}(\varphi)) \in$ Pref-$\mathcal{C}/\log \subseteq$ Full-$\mathcal{C}$/log as required. □

Lemma 5.2: Full-(Full-$\mathcal{C}/\log)/\log$ =Full-$\mathcal{C}/\log$.

Proof: The inclusion from right to left is trivial. We prove the other. Let L be a language in Full-(Full-$\mathcal{C}/\log)/\log$ via a logarithmic advice function f and a language L' in Full-$\mathcal{C}/\log$. Again, let f' and L'' be the logarithmic advice and the language witnessing that L' is in Full-$\mathcal{C}$/log. We define $g : \mathbb{N} \to \Sigma^*$ as $g(n) = \langle f(n), f'(3n)\rangle$. As $f, f' \in \log$, $g \in \log$ too. On the other hand, we define $B = \{\langle x, \langle w_1, w_2\rangle\rangle : \langle\langle x, w_1\rangle, w_2\rangle \in L''\}$. Clearly, $B \in \mathcal{C}$ because $L'' \in \mathcal{C}$ and $\mathcal{C}$ is closed under first-order projection reductions. We prove that L is in Full-$\mathcal{C}$/log via g and B.

Fix a natural number n and let x be a word of length at most n. Let $x' = \langle x, f(n)\rangle$ and $x'' = \langle x', f'(3n)\rangle$. Since $f \in \log$ and $|x| \leq n$, for large enough n, $|x'| \leq 3n$. Hence, we have,

$$x \in L \Leftrightarrow x' \in L' \Leftrightarrow \langle x', f'(|x'|)\rangle \in L'' \Leftrightarrow x'' \in L'' \Leftrightarrow \langle x, g(n)\rangle \in B$$

The third equivalence follows from $|x'| \leq 3n$ and from the fact that f' is a full advice. The last equivalence follows from the definitions of g and B. □

These two lemmas will be useful to prove the following induction step; in combination with Lemma 4.4, it will provide us with a counterpart to Lemma 4.2 for full advice classes.

Lemma 5.3: *Let $c_1, ..., c_n$, $n > 1$, be a sequence of moduled numerical constants. If* Full-$\mathcal{C}/\log$ *subsumes* $\mathcal{L}[c_1, ..., c_{n-1}]$ *then* Full-$\mathcal{C}/\log$ *subsumes* $\mathcal{L}[c_1, ..., c_n]$.

Proof: Suppose that Full-$\mathcal{C}$/log subsumes $\mathcal{L}[c_1, ..., c_{n-1}]$. Since c_n is moduled, Lemma 4.4 says that Full-(Full-$\mathcal{C}$/log)/log subsumes $\mathcal{L}[c_1, ..., c_n]$. But Full-(Full-$\mathcal{C}$/log)/log= Full-$\mathcal{C}$/log and the lemma follows. □

As claimed, the following counterpart to Lemma 4.2 follows by a straightforward induction argument.

Lemma 5.4: *If $\mathcal{C}$ subsumes $\mathcal{L}$ then* Full-$\mathcal{C}/\log$ *subsumes* $\mathcal{L}[\mathbf{c}_{\equiv}]$.

For the other direction we apply the techniques of "double exponential skip" introduced by Hermo in the study of prefixed variants of Full-P/log.

Lemma 5.5: *If $\mathcal{L}$ subsumes $\mathcal{C}$ then $\mathcal{L}[\mathbf{c}_{\equiv}]$ subsumes* Full-$\mathcal{C}/\log$.

Proof: Recall that we assume that our logic can express (or has built-in as numerical relations) the linear order and the BIT predicate. Let A be a set in Full-$\mathcal{C}/\log$ via a set $B \in \mathcal{C}$ and advice function f. For each n, let $w_n = f(n)$.

We can assume without loss of generality that the advice words have length exactly $k \log n$ by prepending enough many zeros. Additionally, we can assume that the constant k is a power of 2 (so that multiplication by k is trivial enough to be done in logic with the BIT predicate).

Change first now the set B into another set B' which expects the bits of w_n to be redistributed into a tuple of $4k$ words $\langle u_1, \ldots, u_{4k}\rangle$ of whatever (roughly similar) length is appropriate. Here the j-th word consists of the bits of w in positions $4kr + j$. The easiness in multiplying by k ensures that B' reduces to B via first-order projections, and therefore $B' \in \mathcal{C}$, which in turn is captured by $\mathcal{L}$. Let ϕ be the formula in $\mathcal{L}$ whose set of models is the set B'. Now this set has the appropriate syntax to assume that the models of ϕ consist of a relational part plus $4k$ constants.

Now, following the analogy between moduled constants and prefixed advice, we will construct, from an arbitrary sequence of logarithmically long advice strings, a right-infinite word whose logarithmically long prefixes can be used instead, and implement this nonuniform information on a fixed number of moduled constants.

These prefixed advice words will be selected as proposed in [BH97] (see [H96] for an analysis in depth of several variants of this construction, together with many consequences). In this case, the limiting right-infinite word has to be the concatenation of those advice words corresponding to lengths a double power of two:

$$\alpha = 00\cdots 0 w_{2^{2^0}} \cdots w_{2^{2^{m-1}}} w_{2^{2^m}} \cdots$$

and the prefixed advice words are to be prefixes of this word, each of the appropriate length. The head of extra zeros is of length k, for technical reasons.

Now we have to prove that we can extract in $\mathcal{L}$, from the concatenation of the selected advice words, the one that we need for each case, and then use its bits, appropriately distributed into first-order variables, as arguments to ϕ. Thus we will prove that A can be expressed in $\mathcal{L}$ with moduled constants, so that $\mathcal{L}[\mathbf{c}_\equiv]$ subsumes Full-$\mathcal{C}/\log$.

For a given n, let 2^{2^m} the smallest double power of two larger than n. Straightforward computation shows that it is quadratic in n, and that the total length of the words from $w_{2^{2^0}}$ up to $w_{2^{2^m}}$ is therefore less than $4k \log n$. Moreover, the first k symbols of α ensure that all the advice words lie in between consecutive powers of 2 (multiplied by k).

So that the first $4k \log n$ bits of α already include near the right end, in easy-to-describe places, the word $w_{2^{2^m}}$, whose bits, appropriately shuffled as we did in passing from B to B', can be used by ϕ instead of w_n due to the "full" property of our advice words. We can obtain them encoded into $4k$ existentially quantified variables, as follows.

Let β be the reverse of $\alpha_{1:4k \log n}$. Assume that we get the $4k \log n$ bits of β distributed in $4k$ moduled constants, $c_{\beta,j}$ for $0 \leq j < 4k$, each consisting of the bits in positions $4kr+j$ of β, for some r, and in the same order as indicated by r. Existentially quantify $4k$ variables x_i and use the BIT predicate to match them with the bits of $c_{\beta,i}$ between 2^{m-2} and 2^{m-1} in reverse order. The remaining bits of x_i must be all zero. There is a placeholder 0 at the end of the first k constants that situates everything into place.

The value of m can be identified after existential quantification. This is easily seen by first writing the first-order formula that checks, given x and j, that 2^j is the smallest power of two strictly larger than x. This is done by checking for a 1 in the j-th bit of x and zeroes above it. The formula has to be applied twice, first to $n-1$ and again to $j-1$ to iterate the process, yielding m. Both 2^{m-2} and 2^{m-1} are immediately identified from m with successor and BIT predicates. $\square$

Taken together, lemmas 5.4 and 5.5 prove our main result in this section:

Theorem 5.6: *Let $\mathcal{C}$ be a nice complexity class captured by a logic $\mathcal{L}$. Then, $\mathcal{L}[\mathbf{c}_{\equiv}]$ captures* Full-$\mathcal{C}/\log$.

Just for a few examples, taking $\mathcal{L} =$ FO(LFP) we have that FO(LFP)$[\mathbf{c}_{\equiv}]$ captures Full-P/log. Similarly, FO(TC)$[\mathbf{c}_{\equiv}]$ and FO(dTC)$[\mathbf{c}_{\equiv}]$ capture Full-NL/log and Full-L/log respectively. We also have a complexity-theoretic consequence, in which capturability by a logic only plays the role of a mild sufficient condition:

Corollary 5.7: *Let $\mathcal{C}$ be a nice complexity class captured by a logic $\mathcal{L}$. Then* Pref-$\mathcal{C}/\log =$ Full-$\mathcal{C}/\log$.

This generalizes the analogous fact shown for P in [BH97] and [H96]. The proof consists simply in using the advice encoding described in the proof of Lemma 5.6; the capturability by a logic is needed to extract the needed advice word.

6. Logical characterizations of reductions to tally sets

It is well known that a language is in P/poly if and only if it is polynomial-time Turing reducible to a tally set; that is, P/poly = P(TALLY). Similarly, NP/poly = NP(TALLY). ¿From the results of Molzan, P(TALLY) and NP(TALLY) are captured by FO(LFP)$[\mathbf{R}]$ and $\Sigma_1^1[\mathbf{R}]$ respectively. The purpose of the present section is to give a logical characterization of NP(A) and P(A) for a fixed tally set A.

Given a relation $R_n \subseteq \{0, ..., n-1\}^r$ let $\overline{R_n}$ be the set $\{\sum_{i=1}^{r} a_i n^{r-i} : (a_1, ..., a_r) \in R_n\}$. As with numerical constants, an r-ary numerical relation $R = (R_1, R_2, ...)$ is said to be moduled if for each $n > 0$, $\overline{R_n} \subseteq \overline{R_{n+1}}$. Intuitively, R_{n+1} is an extension of R_n only by tuples coding numbers greater that n^r.

Note that every tally set can be coded by a moduled numerical relation of arbitrary arity and conversely. Given a tally set A and a fixed arity r, define $R_r^A = (R_{r1}^A, R_{r2}^A, ...)$ as the moduled numerical relation of arity r such that, for each n, $(a_1, ..., a_r) \in R_{rn}^A$ if and only if $0^m \in A$ where $m = \sum_{i=1}^{r} a_i n^{r-i}$. Conversely, given a moduled numerical relation $R = (R_1, R_2, ...)$ of arity r define A as the tally set such that $0^m \in A$ if and only if for all but finitely many n, the n-ary expansion of m, $(a_1, ..., a_r)$, belongs to R_n. Given a tally set A, let $R_A = \{R_r^A : r \geq 1\}$. Given a logic $\mathcal{L}$, $\mathcal{L}[R_A]$ denotes the restriction of $\mathcal{L}[\mathbf{R}]$ to numerical relations in R_A.

The proof of the following theorem is similar to the proof of Fagin's theorem: we build a second-order existential sentence describing the computations of a nondeterministic oracle Turing machine with a tally oracle.

Theorem 6.1: *Let A be a tally set. Then $\Sigma_1^1[R_A]$ captures* NP*(A).*

Proof: Let $\varphi = \exists P_1 ... \exists P_r \psi$ be a sentence in $\Sigma_1^1[R_A]$ where ψ is first-order. We can decide $w(\mathrm{Mod}(\varphi))$ in nondeterministic polynomial time with oracle A by guessing interpretations for $P_1, ..., P_r$ and evaluating ψ. Whenever we need to know if $(a_1, ..., a_r)$ belongs to a moduled numerical relation of arity r, just ask the oracle for 0^m, $m = \sum_{i=1}^{r} a_i n^{r-i}$.

Suppose next that $L \in \mathrm{NP}(A)$ via a nondeterministic oracle Turing machine N time-bounded by n^c, $c \geq 1$. Assume, without loss of generality, that N has only one working-tape, that N only writes words of $\{0\}^*$ to its oracle tape, and that it writes them from left to right. Fix an input $x = w(M)$, the encoding of a structure $M = (\{0, ..., n-1\}, P^M)$ where P is a unary relation symbol (recall that for $\tau = \{P\}$, $M_\tau(x)$ is always defined). We describe a second-order existential formula φ such that M is a model of φ in $\Sigma_1^1[R_c^A]$ if and only if $x \in L(N, A)$.

The formula φ has the form $\exists S \exists T \exists T_0 ... \exists T_r \exists C_0 \exists C_1 \psi$ where ψ is a first-order sentence. The arities of S, T, T_i and C_i are $2c, 1, 2c$ and c respectively. Moreover $r = |(Q \times \Gamma) \cup \Gamma| + 1$ where Q is the set of internal states of N and Γ is its tape alphabet. Intuitively, ψ says that $T, T_0, ..., T_r, C_0, C_1$ encode an accepting computation of $N(x, A)$ and S is a successor relation over c-tuples. It is not difficult to find a first-order formula ψ_1 expressing that S is such a successor relation. We shall describe next a formula ψ_2 expressing that $T, T_0, ..., T_r, C_0, C_1$ encode an accepting computation. First, $T(x)$ means that the head of the input tape points to the x-th cell, its contents being $P(x)$. Let $(Q \times \Gamma) \cup \Gamma = \{b_0, ..., b_{r-1}\}$. Intuitively, if $b_i = b \in \Gamma$ then $T_i(\overline{x}, \overline{y})$ states that, at time $\overline{y}$, the $\overline{x}$-th cell of the working-tape contains the symbol b. On the other hand, if $b_i = (q, b) \in Q \times \Gamma$, then $T_i(\overline{x}, \overline{y})$ states that, at time $\overline{y}$, (i) the $\overline{x}$-th cell of the working-tape contains the symbol b, (ii) the tape-head is at this cell and (iii) the internal state of the machine is q. Finally, $T_r(\overline{x}, \overline{y})$ states that, at time $\overline{y}$, the oracle tape contains the word $0^{\overline{x}}$ and $C_i(\overline{y})$ states that, at time $\overline{y}$, the i-th, $i \in \{0, 1\}$, nondeterministic choice is made.

First of all, ψ_2 should say that, at time $\overline{0}$, (i) the head of the input tape is at the left end ($T(0)$), (ii) the internal state is q_0, the initial state of N and (iii) the oracle tape is empty ($T_r(\overline{0}, \overline{0})$). Next, ψ_2 should say that the contents of the $\overline{x}$-th cell of the working tape at time $\overline{y}+1$ follows from the internal state of the machine, the contents of the cells $\overline{x}-1$, $\overline{x}$ and $\overline{x}+1$ at time $\overline{y}$ and the contents of the cell pointed in the input tape. Moreover, if a symbol is written in the oracle tape then, if $0^{\overline{x}}$ were written at time $\overline{y}$ then $0^{\overline{x}+1}$ is written at time $\overline{y}+1$. The crucial point of the construction is the following: if at time $\overline{y}$ the internal state of N is $QUERY$ and the oracle tape contains the word $0^{\overline{x}}$ ($T_r(\overline{x}, \overline{y})$), then at time $\overline{y}+1$, the internal state of the machine is YES or NO according to whether $R_c^A(\overline{x})$ holds; moreover, the oracle tape is empty at time $\overline{y}+1$. Finally, two things remain to be said about the computation. Namely, that exactly one nondeterministic choice is taken at each step ($C_0(\overline{y}) \leftrightarrow \neg C_1(\overline{y})$) and that the computation halts in an accepting internal state of N.

We see next that M is a model of $\exists S \exists T \exists T_0 ... \exists T_r \exists C_0 \exists C_1 (\psi_1 \wedge \psi_2)$ in $\Sigma_1^1[R^A]$ if and only if $x \in L(N, A)$. From left to right it is clear from the construction of φ and because for all $\overline{x} \in \{0, ..., n-1\}^c$, $\overline{x} \in R_{cn}^A$ if and only if $0^{\overline{x}} \in A$. On the other hand, if $x \in L(N, A)$ then, since N is time-bounded by $|x|^c$, every accepting computation of $N(x, A)$ asks the oracle for words of length at most $|x|^c$. Thus, since $|x|^c \leq n^c$, R^A_{cn} contains enough information about A to satisfy φ. $\Box$

Looking carefully at the proofs that FO(LFP), FO(TC) and FO(dTC) capture P,NL and L, one can see that the proof above also works for them. In the following let $\mathcal{C}$ be a class in $\{\mathrm{NP}, \mathrm{P}, \mathrm{NL}, \mathrm{L}\}$ and let $\mathcal{L}$ be the corresponding logic capturing $\mathcal{C}$.

Theorem 6.2: *Let A be a tally set. Then $\mathcal{L}[R_A]$ captures $\mathcal{C}(A)$.*

Molzan's result follows from ours due to this refining on the numerical relations and to the fact that P(TALLY)=P/poly.

In [BH97] it was proved that Pref-P/log could be characterized as the closure of P under (Turing) reductions to a restricted class of tally sets. Namely, it was proved that Pref-P/log= P(TALLY2) where TALLY2 $= \{L : L \subseteq \{0^{2^k} : k \in \mathbb{N}\}\}$. Following the proof in [BH97] it is easily seen that for $\mathcal{C}$ in $\{\mathrm{NP}, \mathrm{P}, \mathrm{NL}, \mathrm{L}\}$ it also holds that Pref-$\mathcal{C}/\log = \mathcal{C}(\mathrm{TALLY2})$. Of course, every tally2 set is tally and, as a result, it can be encoded by a moduled numerical relation $R = (R_1, R_2, ...)$; moreover, every tuple in R_n encodes a power of 2. Hence, if $\mathbf{R}_{\equiv 2}$ is the class of moduled numerical relations whose tuples encode powers of 2, our results in this paper prove the following corollary.

Corollary 6.3: *$\mathcal{L}[\mathbf{R}_{\equiv 2}]$ and $\mathcal{L}[\mathbf{c}_{\equiv}]$ have the same expressive power in the finite.*

The proof is a simple combination of theorems 6.2, 5.6 and corollary 5.7. Recall however that all our logics are assumed to express the linear ordering and the BIT predicate (due to their own expressive power or by having them built-in as default numerical relations).

Acknowledgments: We are grateful to Montserrat Hermo, Ricard Gavaldà and Miguel Carrillo for helpful discussions on some of the topics of this paper.

References

[BBH97] J. Balcázar, H. Buhrman, M. Hermo. Circuit expressions of low Kolmogorov complexity. In preparation.

[BDG95] J. Balcázar, J. Díaz, J. Gabarró. *Structural Complexity I.* Second Edition. Texts in Theoretical Computer Science. Springer-Verlag 1995.

[BH97] J. Balcázar, M. Hermo. The structure of logarithmic advice complexity classes. In preparation.

[BS92] J. Balcázar, U. Schöning. Logarithmic Advice Classes. *Theoretical Computer Science*, 99:279-290, 1992.

[BIS90] D. Barrington, N. Immerman, H. Straubing. On Uniformity within NC^1. *J. of Computer and System Sciences*, 41:274-306, 1990.

[F74] R. Fagin. Generalized First-Order Spectra and Polynomial-Time Recognizable Sets. *Complexity of Computation, R. Karp, ed., SIAM-AMS Proc.*, 7:27-41, 1974.

[GL84] Y. Gurevich, H. Lewis. A logic for constant depth circuits. *Inform. and Control*, 61:65-74, 1984.

[HMPST93] A. Hajnal, W. Maass, P. Pudlák, M. Szegedy, G. Turán.rew Threshold Circuits of Bounded Depth. *J. Computer System Sciences*, 46:129-154, 1993.

[H96] M. Hermo. *Nonuniform Complexity Classes with Sub-Linear Advice Functions.* Doctoral Thesis, 1996.

[HM94] M. Hermo, E. Mayordomo. A Note on Polynomial Size Circuits with Low Resource-Bounded Kolmogorov Complexity. *Mathematical Systems Theory*, 27(4):347-356, 1994.

[I87] N. Immerman. Languages that Capture Complexity Classes. *SIAM J. Comput.*, 16(4):760-778, 1987.

[IL95] N. Immerman, S. Landau. The Complexity of Iterated Multiplication. *Information and Computation*, 116:103-116, 1995.

[KL80] R. Karp, R. Lipton. Some connections between nonuniform and uniform complexity classes. *Proc. 12th ACM Symposium on the Theory of Computing*, 302-309, 1980.

[K87] K. Ko. On Helping by Robust Oracle Machines. *Theoretical Computer Science*, 52:15-36, 1987.

[M90] B. Molzan. Expressibility and Nonuniform Complexity Classes. *SIAM J. Computing*, 19(3):411-423, 1990.

[S94] H. Straubing. *Finite Automata, Formal Logic, and Circuit Complexity.* Birkhäuser, 1994.

[V82] M. Vardi. The Complexity of Relational Query Languages. *Proc. 14th ACM Symp. on Theory of Computing*, pages 137-146, 1982.